BUDAI ATTILA

POV-R

VRML

A SZÁMÍTÓGÉPES GRAFIKA

Breeze

CD MELLÉKLETTEL

LSI INFORMATIKAI OKTATÓKÖZPONT

BUDAI ATTILA

A SZÁMÍTÓGÉPES GRAFIKA

Nyitott rendszerű képzés – Távoktatás –
Oktatási segédlete
Felsőoktatási Tankönyv

GÁBOR DÉNES FŐISKOLA

Budapest, 2002

Lektorálta: **Dr. Kovács Magda, PhD**
főiskolai tanár

Dr. Hosszú Gábor
a műszaki tudomány kondidátusa
egyetemi docens

Dr. Kun István, PhD
főiskolai tanár

Korrektor: **Márkus Orsolya**

Szerkesztés, tördelés: **OFFICE-CAR Bt.**
2040 Budaörs Csata u. 17.

A jegyzet megrendelhető, illetve megvásárolható az
LSI Informatikai Oktatóközpontban
1037 Budapest, Bécsi út 324.
Telefon: 436-6520
Fax: 436-6521

ISBN 963 577 243 2

Kiadó: **LSI Informatikai Oktatóközpont**
Felelős vezető: **Dr. Kovács Magda**
Témafelelős: **Flier István**

TARTALOMJEGYZÉK

ELŐSZÓ

A számítógépes képfeldolgozás és grafika fejlődése az elmúlt években rendkívül felgyorsult. Ennek néhány jellemző vonása:

a) A grafikus felület (GUI) használata világszabvánnyá vált. Ezt követően a számítógépes grafika többi területén is az egyre erősödő szabványosítás volt a jellemző.

b) Az elképzelt vagy méretük/távolságuk miatt láthatatlan, illetve veszélyességük miatt az ember számára megközelíthetetlen világok, az ún. „virtuális valóságok" (VR) valósághű, háromdimenziós (3D) modellezését és interaktív megjelenítését a mai technológia már lehetővé teszi. Ezt például egyre intenzívebben kihasználja a szórakoztatóipar is a filmgyártásban.

c) Az ún. fotorealisztikus képábrázolást lehetővé tevő algoritmusok (raytracing, radiosity) hatékonysága rohamosan javul, felhasználásuk különösen a film- és reklámszakmában terjed, de ezeket egyre több más szakterületen is kezdik alkalmazni a várostervezéstől a belsőépítészetig.

d) A valósághű grafikus képek megjelenése oda vezet, hogy a korábban lényegesen elkülönülő két szakterület, a számítógépes grafika és a multimédia közötti határ elmosódik. Terjed a grafikus animáció a WEB-en, a grafikus képek lassan megkülönböztethetetlenek a fototechnikával előállított és digitalizált képektől.

e) A grafikus programcsomagok előállítói (COREL, AUTODESK stb.) között rendkívül éles a piaci verseny, rendkívül felgyorsult a programcsomagok új verzióinak megjelenése.

A szoftvergyártóknál érezhető a teljességre való törekvés, azaz szolgáltatásaikkal szeretné lefedni a képfeldolgozás minden funkcionális területét, az Internetes csoportmunkától kezdve a multimédiás feldolgozások és a 3D grafikákon keresztül, a nyomdai kapcsolat biztosításáig.

f) A fontosabb tervezőprogramok (pl. CAD) szállítói már mindenütt biztosítják az Internetre kapcsolódás lehetőségét, sőt többségük az Internetes csoportmunkát is.

Emellett a különböző képeket, grafikus (pl. CAD) objektumokat tartalmazó adatbázisok száma és tartalma a WEB-en nagyon gyorsan bővül.

Mindezekre is figyelemmel levonhatjuk a következtetést, hogy napjainkban már a számítógépes grafika alapeljárásainak ismerete az informatikában nélkülözhetetlen.

A „Számítógépes grafika" című könyv a számítógépes grafika és képfeldolgozás legfontosabb fogalmait, algoritmusait és szabványait mutatja be az olvasónak. A szerző arra törekedett, hogy e viszonylag szerteágazó szakterület ismeretanyagából a lényeges tudnivalókat kiemelje és összefoglalja, és a számítógépes grafika összes fontosabb részterületéről adjon egy átfogó képet.

Terjedelmi korlátok miatt sincs lehetőségünk, hogy az egyes algoritmusokat teljes részletességgel bemutassuk. Ebben az is gondot okoz, hogy az egyes eljárásokhoz

szükséges matematikai apparátus (funkcionálanalízis, kombinatorikus topológia, projektív geometria stb.) messze túlmutat a műszaki felsőoktatás matematika tananyagán. Ennek ellenére a szerző – ahol csak ez lehetséges – megpróbál szemléletesen utalni az algoritmusok fontosabb elemeire.

A könyv ismeretanyagát három kategóriába soroltuk:

- a kurzívval szedett részek tartalmazzák a könyv mondanivalójának lényegét,
- a normál betűtípussal szedett részek magyarázzák, példákon keresztül mutatják be az egyes fogalmakat, eljárásokat,
- a kisebb betűmérettel szedett részek nélkül a könyv tartalma teljes mértékben érthető, ezek jórészt kiegészítő információkat tartalmaznak

A könyv a Gábor Dénes Főiskola „Számítógépes grafika" című tantárgyának tankönyve. Ezzel összefüggésben a könyv megírása során az volt a legfontosabb célkitűzés, hogy a hallgatók e témakörben olyan ismeretanyagot, fogalomrendszert és szemléletmódot sajátítsanak el, mely e gyorsan fejlődő szakterületen is időtálló és felkészít a konkrét grafikus programcsomagok gyakorlati alkalmazására.

Ennek megfelelően került összeállításra a könyv CD melléklete, mely az összes olyan oktatóanyagot, programokat, objektumkönyvtárakat tartalmazza, amivel bárki PC-jén egy „mini" grafikus munkahelyet alakíthat ki és az összes fontosabb algoritmust a gyakorlatban is kipróbálhatja.

A könyv érdemi részét egy viszonylag bőséges irodalomjegyzék zárja le. Ezek közül különösen a megadott Internet címek önálló felkeresését és tanulmányozását ajánlom az olvasónak.

Bízom abban, hogy a Gábor Dénes Főiskola hallgatóin túlmenően a könyv segíthet bárkinek, aki a számítógépes képfeldolgozás és grafika alapjait szeretné megismerni és egyes eljárásait a gyakorlatban is kipróbálni.

A könyv tartalmilag és formailag a távoktatás követelményrendszerének kíván megfelelni. Ezt szolgálják többek között az egyes fejezetek végén található ellenőrző kérdések, melyekkel bárki ellenőrizheti az ismeretanyag megértését és elsajátítását, felidézheti az olvasottakat.

A könyv készítése során megfelelő gondossággal igyekeztünk eljárni, vélhetően ez azonban nem óvott meg a tévedésektől. A szerző előzetesen is köszönetet mond mindenkinek, aki az alábbi E-mail címen az esetleges hibákra felhívja a figyelmét, vagy a könyvvel kapcsolatos észrevételeiről tájékoztatja.

Végezetül a szerző köszönetet mond mindenkinek, aki segített a jegyzet megírásában és kiadásában, különösen a Gábor Dénes Főiskola vezetői, oktatói és hallgatói közül azoknak, akik észrevételeikkel hozzásegítettek a könyv tartalmának kidolgozásához.

Budapest, 1999. augusztus

A Szerző
budai@gdf-ri.hu

A KÖNYV CD MELLÉKLETÉNEK TARTALMA ÉS HASZNÁLATA

A CD első része a virtuális valóság modellező nyelv 1 és 2 verziójának (VRML1 és VRML2), ezek lejátszó programjainak (Live 3D, Cosmo Player) és a Povray fotorealisztikus képeket előállító renderelőprogramnak a magyar nyelvű oktatóprogramjait tartalmazza. Ezek e szoftverekkel kapcsolatos legfontosabb ismereteket foglalják össze, melyek elsajátításával az olvasó képes lesz ezekkel a programcsomagokkal egyszerűbb feladatokat megoldani. A CD e programokra vonatkozó teljes dokumentációt is tartalmazza angol nyelven. Ezt több ezer grafikus építőelemet (vektorgrafikus objektumok, textúrák, raszteres képek pl. clipartok, fraktálok stb.) tartalmazó objektumkönyvtárak egészítik ki. A különböző grafikai eljárásokkal készített 3D tárgyakat, „világokat", képeket több ezres elemszámú galériákban tanulmányozhatjuk. A CD-n található 48 freeware és shareware programból álló készlettel az összes fontosabb grafikus feladat (például objektummodellezés, speciális effektek, képmódosítás, szerkesztés, konvertálás) megoldható. Összességében a CD tartalma alapján egy komplett vektor és rasztergrafikus munkahelyet rendezhetünk be számítógépünkön.

Ajánlott konfiguráció:

Pentium II. min. 266 MHz
64 Mbyte RAM
Grafikus kártya min. 4 Mbyte video RAM-al, 2D és 3D gyorsítással
Monitor min. 800x600 felbontással, 75 Hz vagy nagyobb képfrissítési frekvenciával
Min. 1 Gbyte harddisk
WIN 95 operációs rendszer

A programok jelentős része ennél kisebb teljesítményű konfiguráción (pl. 486 DX, 8 Mbyte RAM) is futtatható, de egyes programoknál – pl. bonyolultabb objektumok renderelése – ez esetben jelentős lassulásra számíthatunk.

A CD tartalmát HTML formátumban dolgoztuk ki, ezért használatba vétele feltételezi, hogy gépünkön telepítve legyen valamilyen, ezt a formátumot kezelni tudó WEB Browser. Ha ilyennel nem rendelkezünk, akkor első lépésben telepítsük fel a CD-ről a NETSCAPE Navigátort vagy Communicator-t (utóbbi nagyobb teljesítményű gép-konfiguráció esetén ajánlott). Ezt követően az INDIT oldalt betöltve megkapjuk a CD tartalomjegyzékét, melynek linkjeire kattintva tölthetjük be az oktatóprogramokat, léphetünk be az egyes galériákba stb. A CD-n található programok döntő része feltételezi, hogy az installáláskor gépünkön csak a SETUP fut és az installációt követően gépünket újraindítjuk. Felhívjuk a figyelmet arra, hogy a SETUP-ok egy része erre nem figyelmezteti a felhasználót.

SZÁMÍTÓGÉPES GRAFIKA CD MELLÉKLETÉNEK TARTALOMJEGYZÉKE

KÉSZÍTETTÉK: BUDAI ATTILA, SÁVOS TAMÁS, PHAM VU KIEN CUONG

I. A VIRTUÁLIS VALÓSÁG MODELLEZÉSE

1.1 A VIRTUÁLIS VALÓSÁG MODELLEZŐ NYELVE 1. VERZIO /VRML1/

Oktatóprogram

Készítette Dr. Szilassi Lajos, Szeged, E-mail:szilassi@jgytf.u-szeged.hu
Az oktatóprogram a LIVE 3D lejátszóval mutatja be a VRML 1 nyelv lehetőségeit. Foglalkozik a különböző geometriai objektumok definiálásával, kamerakezeléssel, bemutatja a felhasználói felület lehetőségeit (forgatás, séta a 3D térben stb). Az oktatási anyagot számos mintapéldát (VRML-fájlt) tartalmazó könyvtár egészíti ki.

1.2 A VIRTUÁLIS VALÓSÁG MODELLEZŐ NYELVE 2. VERZIO /VRML2/

Oktatóprogram

Készítette Neithammer Zoltán, Szentes, E-mail:neithammer@mail.datanet.hu
Az oktatóprogram a VRML 2 alapismeretek tárgyalását követően, többek között az objektumok generálásával, transzformációkkal, lámpák és kamerák elhelyezésével, animációkészítéssel, a 3D térben szenzorok elhelyezésével, hangdefiniálással foglalkozik. Az oktatóprogramot egy jelentős mennyiségű VRML 2 prototípust tartalmazó objektumkönyvtár egészíti ki.

1.3 COSMO PLAYER VIRTUÁLIS VALÓSÁG LEJÁTSZÓ PROGRAM

A Silicon Graphics cég oktatóprogramja

A kezelőfelület magyar nyelvű leírását készítette Neithammer Zoltán, Szentes, E-mail:neithammer@mail.datanet.hu
Az oktatóprogram egy 3D VRML2 tenger alatti virtuális világban ismertet meg a COSMO PLAYER felhasználói felületének lehetőségeivel, a virtuális világban tett séta irányításával.

Cosmo Player 2.1 angol nyelvű dokumentáció

Cosmo Player 2.1 Releas Notes for Windows 95 and Windows NT

1.4, VRML 2 KOMPLEX VILÁGOK GALÉRIÁJA

A galéria VRML2-vel felépített, esetenként filmszerű hatást keltő 3D virtuális világokat tartalmaz (pl.: űrhajó belseje).

II. A POVRAY RAYTRACING PROGRAM

2.1, Povray oktatóprogram magyarul
Az oktatóprogram ismerteti a POVRAY fotorealisztikus renderelőprogram objektumainak definiálását (primitívek, színkezelés, kamera, textúrák stb.) és egy példa feladaton bemutatja a program kezelését.

2.2, Teljes oktatóprogram angol nyelven
Az oktatóanyag a POVRAY lehetőségeit teljes körűen ismerteti.

2.3, Povray művészi képek galléria
A galériában a Povray-el készített fotorealisztikus képek és művészi alkotások találhatók.

2.4, Povray textúra gyűjtemény
A textúra könyvtár a Povray-hez több száz textúra definíciót tartalmaz (fa, márvány, kő stb.).

2.5, Modellezők a Povray-hez, Povray pluginok
A modellezők általános jellemzése és közülük a legismertebbek (pl.: Breeze Designer) bemutatása, a Povray pluginokra néhány példa.

III. SZÁMÍTÓGÉPES GRAFIKÁVAL KÉSZÍTETT KÉPEK ÉS FILMEK GALÉRIÁJA

3.1, Clipartok
3.2, Textúrák
3.3, Fraktálok
3.4, AnimaltGif
3.5, Stereogrammok
3.6, Fotórealisztikus képek (raytracing, radiosity eljárás)
3.7, Számítógépes grafikával készített filmek

IV. GRAFIKUS FREEWARE, SHAREWARE PROGRAMOK, SZÍNKÓDOLÓ, KÉPANYAG A SZÁMÍTÓGÉPES GRAFIKA KÖNYVHÖZ

!!!A SHAREWARE PROGRAMOK JOGSZERŰ FELHASZNÁLÁSÁRA VONATKOZÓ INFORMÁCIÓK A MELLÉKELT DOKUMENTÁCIÓKBAN TALÁLHATÓK. AZ EBBEN FOGLALT ELŐÍRÁSOK BETARTÁSÁÉRT A FELHASZNÁLÓ FELELŐS!!!

4.1. 3D grafikus modellezők a Povray-hez, objektumgenerátorok, textúrakészítők

- ***Breeze Designer (freeware)***

Megtalálható: /programok/Prog1/Breeze
Modellező a POVRAY-hez.

- ***AC3D (freeware)***
Megtalálható: /programok/Prog1/AC3D
Modellező a POVRAY-hez.
- ***Povlab 4.0 (freeware)***
Megtalálható: /programok/Prog1/Povlab4.0
Modellező a POVRAY-hez.
- ***Moray for Windows (shareware)***
Megtalálható: /programok/Prog1/Morayforwin
Modellező a POVRAY-hez.
- ***TexturaPov (freeware)***
Megtalálható: /programok/Prog1/TexturaPov
Textúragenerátor a POVRAY-hez.
- ***Spatch V2.0 (freeware)***
Megtalálható: /programok/Prog1/sPatch
Spline alapú 3D felületgenerátor POV és DXF exporttal.
- ***TreeDesigner (freeware)***
Megtalálható: /programok/Prog1/treedesigner
Fa és növényzet modellező a Moray és a Povray számára.
- ***LParser (freeware)***
Megtalálható: /programok/Prog1/LParser
Szövegvezérelt objektumgenerátor (pl.: növényzet).
- ***LParser Frontend (freeware)***
Megtalálható: /programok/Prog1/LParserFrontend
32 bites Windows-Frontend az LParser számára.
- ***Terragen (freeware)***
Megtalálható: /programok/Prog1/terragen
3D tájképgenerátor.
- ***Universe (freeware)***
Megtalálható: /programok/Prog1/universe
3D világűr képek generátora.

4.2. 3D vektorgrafikus programok

- ***Behemot Graphics Editor V.0.8.1. (freeware)***
Megtalálható: /programok/Prog2/BehemotGraphicsEditorV0.81
Grafikus 3D objektumgenerátor (b-rep, NURBS) és renderelő. Ismeri a CSG műveleteket, DXF és VRML és MPEG formában exportálhatók a fájlok.
- ***M3D (freeware)***
Megtalálható: /programok/Prog2/M3D
Függvény egyenlet alapján 3D felületet kirajzoló program .

- ***Metamorpher 98 (freeware)***
Megtalálható: /programok/Prog2/Metamorpher
Kiinduló és befejező kép közötti morphing-ot készítő program.
- ***SGRAM (Custom Stereograms) (freeware)***
Megtalálható: /programok/Prog2/CustomStereogram
Sztereogram készítő program.
- ***Alice (freeware)***
Megtalálható: /programok/Prog2/Alice
Szövegvezérelt, interaktív 3D modellező.
- ***WcvT2Pov (freeware)***
Megtalálható: /programok/Prog2/WcvT2Pov
3D fájlformátum konverter (DXF, 3DS és POV) .
- ***Street Graphics V.1.0.6 (shareware)***
Megtalálható: /programok/Prog2/StreetGraphics
Vektoros rajzoló program.

4.3. Grafikus képfeldolgozók, konverterek

- ***Adobe Acrobat (freeware)***
Megtalálható: /programok/Prog3/AdobeAcrobat
*.PDF fájlnéző.
- ***B_Spline (freeware)***
Megtalálható: /programok/Prog3/B_Spline
B_Spline generátor.
- ***IrfanView 32 (freeware)***
Megtalálható: /programok/Prog3/IrfanView32
Képnéző és konvertáló (JPG, GIF, BMP, DIB, RLE, PCX, DCX, PNG, TIFF, TGA, RAS/SUN, ICO, CUR, ANI, AVI, WAV, MID, RMI, WMF, EMF, PBM, PPM, IFF/LBM, PSD, CPT, MPG, MOV, PCD).
- ***AnmanieSmp V2.0 (freeware)***
Megtalálható: /programok/Prog3/AnmanieSmpV2.0
Grafikus eszköztár képrészletek nagyítására, összevonására, egérrel való vontatására.
- ***Ico2Bmp V.1.0 (freeware)***
Megtalálható: /programok/Prog3/Icon2BmpV1.0
A *.ico file-okat *.bmp-be konvertálja.
- ***GuckMal 2000 V0.1 (shareware)***
Megtalálható: /programok/Prog3/Guckmal2000V0.1
A program közel 40 képformátumot dolgoz fel, pl.: BMP, PCX, GIF, TIF, JPG, PCD, PNG, CLP, CUT, ICO, IFF, IMG, MAC, MSP, PCT, PSD, RLE, TGA, WMF, WPG stb.
FUNKCIÓK: Konvertálás, szkennelés TWAIN-el, kivágás, skálázás, világosság, kontraszt és fényerő változtatás, forgatás, tükrözés, színkezelés stb.

- *Carricature V1.8.1 (freeware)*
Megtalálható: /programok/Prog3/CarricatureV1.81
Képfeldolgozó program, mellyel karikatúrákat lehet készíteni.
- *Microangelo 98 V4.72 (shareware)*
Megtalálható: /programok/Prog3/Microangelo98V4.72
Ikonkezelő, készítő program.
- *Microangelo Gif animator V.1.0 (shareware)*
Megtalálható: /programok/Prog3/MicroangeloGifAnimatorV1.0
Gif animáció készítő.
- *PCD-Viewer (shareware)*
Megtalálható: /programok/Prog3/PcdViewerV2.1
Photo-CD néző és konvertáló, dia előállító funkció, hang és szövegbeágyazás.
- *Kamera V1.1 (shareware)*
Megtalálható: /programok/Prog3/KameraV1.1
Eszközkészlet, mellyel képernyő fotókat lehet készíteni és BMP formátumba letárolni

4.4. Fraktál generátor
- *Fractindos (freeware)*
Megtalálható: /programok/Prog4/fractindos
DOS-os fraktálgenerátor.
- *Winfractint (freeware)*
Megtalálható: /programok/Prog4/winfractint
Fraktálgenerátor (Windows, Win95).

4.5. RGB automatikus színkodoló
Egy HTML oldalon kiválaszthatjuk a kívánt színárnyalatot, melyet az oldal háttérszíne automatikusan felvesz, megmutat. A lap alsó sorában olvasható a megfelelő RGB kód.

4.6. Képanyag a számítógépes grafika könyvhöz
A számítógépes grafika könyv egyes fejezeteihez illusztrációként színes képek és ábrák találhatók itt.

4.7. A CD használatához szükséges programok
- *Netscape Navigator Gold 3.01*
Megtalálható: /programok/Prog9/3.01/nsg301.exe
Web browser, kisebb erőforrásigénnyel.
- *Netscape Communicator 4.51 Relase Notes-al*
Megtalálható: /programok/Prog9/NetscapeCommunicator4.51
Web browser, nagyobb erőforrásigénnyel.

- ***Internet Explorer 4.0 angol (16 bites)***
Megtalálható /programok/Prog9/ie4setup/setup.exe
Web browser, nagyobb erőforrásigénnyel.
- ***Internet Explorer 4.0 magyar (32 bites)***
Megtalálható: /programok/Internet Explorer4.0magyar/ie4setup.exe
Web browser, nagyobb erőforrásigénnyel.
- ***Cosmo Player 2.1 (freeware)***
Megtalálható: /programok/Prog9/cosmo/cosmo.exe
VRML1 és 2 lejátszó plugin.
- ***Media Player (shareware)***
Megtalálható: /programok/Prog9/MediaPlayer/mpfull.exe
*.mpg, *.avi, *.mov filmlejátszó.
- ***WinZip (shareware)***
Megtalálható: /programok/Prog9/Winzip/Winzip95.exe
Zip fájl tömörítő és kifejtő.
- ***Povray (freeware)***
Megtalálható: /programok/Prog9/Povray/Povwin3.exe
Fotorealisztikus raytracing renderelő program.

JELÖLÉSEK MAGYARÁZATA

! – Kiemelt jelentőségű kérdésekre, tananyag lényegére hívja fel a figyelmet. Ellenőrző kérdésre adja meg a választ. *Ezeket a részeket kurzívan szedtük, így elkülönülnek a tananyagban.*

☝ – Fontos összefüggésekre hívja fel a figyelmet, ellenőrző kérdésre adja meg a választ.

 – Magyarázat, indoklás, példa.

 – Kiegészítő ismeretek a könyvhöz, melyek betűformájukkal is elkülönülnek a tananyagtól. Nem része a főiskolai tananyagnak.

? – Ellenőrző kérdés(ek)

(1)2.3 – (Al)Fejezet azonosító

I.

A SZÁMÍTÓGÉPES GRAFIKA FOGALMA, A GRAFIKUS RENDSZEREK TÍPUSAI ÉS FELHASZNÁLÁSI TERÜLETEIK

E fejezetben először a számítógépes grafika és digitális képfeldolgozás alapfogalmait tárgyaljuk, definiáljuk a raszter és a vektorgrafikát. Foglalkozunk a grafikus rendszerek architektúrájával és bemutatjuk a számítógépes grafika felhasználásának legfontosabb területeit.

1.1. A SZÁMÍTÓGÉPES GRAFIKA ÉS A DIGITÁLIS KÉPFELDOLGOZÁS FOGALMA

A számítógépes képfeldolgozás az elmúlt harminc évben az informatika egyik legjelentősebb szakterületévé vált, több milliárd dolláros üzletággá fejlődött. Ennek okát a képi információk kifejezőerejében kereshetjük: egy grafikával előállított diagramm pl. egy cég bevételeinek növekedéséről sokkal gyorsabban átlátható és informatívabb, mint egy számtáblázat.

Ugyanakkor az elmúlt években a hardver teljesítmény/ár mutatójának rohamos növekedése és az egyre inkább szabványosított szoftverek alkalmazása a számítógépes képfeldolgozó rendszereket gazdaságossá is tette.

A vizuális információk jelentőségével már az ókorban is tisztában voltak.

A grafika szó a görög vésni szóból származik, aminek az a magyarázata, hogy az ókorban a képeket leggyakrabban így állították elő. Ma már a grafika a rajzművészet összefoglaló fogalmát jelenti.

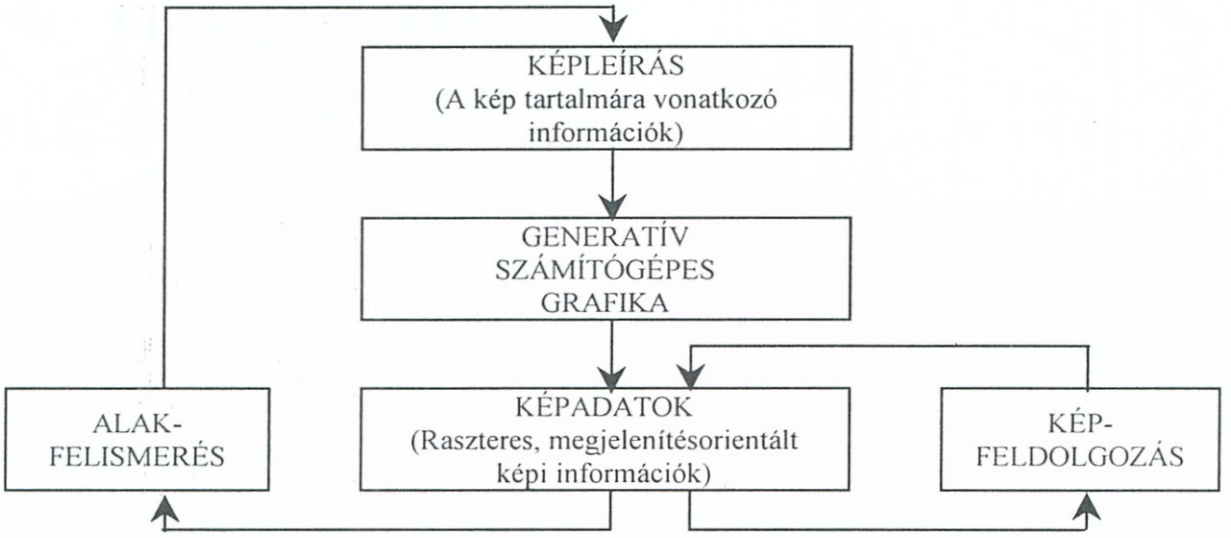
Az ISO (International Standards Organization) *nemzetközi szabványügyi szervezet adatfeldolgozási fogalomgyűjteménye* (Data Processing

Vocabulary) *szerint a számítógépes grafika* (computer graphics) *fogalma alatt az adatkonvertálási módszereket és eljárásokat értjük a számítógép és a grafikus perifériák között:*

„Methods and techniques for converting data to and from graphics displays via computer" (ISO82).Ez alapján három, relatíve önálló részterületet különböztethetünk meg:

- a generatív számítógépes grafikát (interactive computer graphics),
- a képi információk számítógépes feldolgozását (image processing),
- a számítógépes képelemzést, alakfelismerést (picture analysis).

Ezek viszonyát mutatja az 1. sz. ábra.



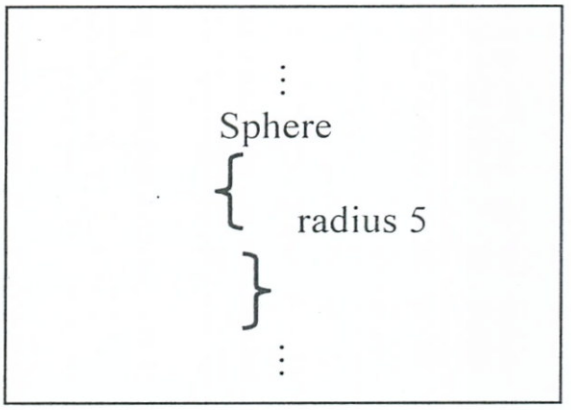
1. sz. ábra
A generatív számítógépes grafika, az alakfelismerés és a képfeldolgozás összefüggései

A magyar informatikai gyakorlat terminológiája részben eltér az ISO meghatározásától. Számítógépes grafika alatt általában a generatív számítógépes grafikát értik, a számítógépes képfeldolgozást, képelemzést és alakfelismerést pedig digitális képfeldolgozásnak nevezik.

1.1.1. Generatív számítógépes grafika

A generatív számítógépes grafika ezek szerint a képi információ tartalmára vonatkozó képleírási adatok (és nem a kép) *alapján, algoritmusokkal állít elő pl. a monitor képernyőjén megjeleníthető képeket.*

Ehhez a képleírások valamely szabályrendszer (szintaxis) szerint határozzák meg a grafikus objektumokat. Egy 5 sugarú kör definiálására mutat példát a 2. sz. ábra.



2. sz. ábra
Gömb definiálása képleírási adatokkal

A számítógépes grafika tárgyát – az előbbiekre is figyelemmel – a következőképpen határozhatjuk meg.

A számítógépes grafika alatt a két- (2D) és három- (3D) dimenziós grafikus objektumok számítógépes generálását, tárolását, feldolgozását és megjelenítését értjük.

1.1.2. Digitális képfeldolgozás

A képfeldolgozás mindazon számítógépes eljárások és módszerek összességét jelenti, melyekkel a számítógépen tárolt képek minőségét valamilyen szempont szerint javítani lehet (pl. élek kiemelése) és ezáltal továbbfeldolgozásra alkalmasabbá válnak.

A képfeldolgozó rendszerek a képeket nem generálják, hanem inputként kapják meg (például digitalizált fotók, mesterséges hold felvételek stb. formájában).

A képfeldolgozás alapegysége a raszteres kép, mely n x m képpontból (pixelből) áll. A képpontokat szürkeség, vagy színinformációkkal jellemezzük, ezek együttese alkotja a képadatokat.

A számítógépes alakfelismeréssel a raszteres képeken lévő grafikus objektumok azonosítását végezzük el. (Például egy város mesterséges hold által felvett képén az egyes utcák és ezek kereszteződéseinek megállapítása.) Ezekkel – a legtöbb esetben bonyolult matematikai eljárásokkal – a raszteres képből kinyerjük a képleíráshoz szükséges lényegi információkat.

Az alakfelismerés és a képfeldolgozás módszereinek és eljárásainak együttesét nevezzük digitális képfeldolgozásnak.

1.2. VEKTORGRAFIKUS ÉS RASZTERGRAFIKUS RENDSZEREK

A grafikus képek számítógépes feldolgozása a kezdetekben csak a különböző rajzoló programok alkalmazását jelentette. Ezek rasztergrafikus alapon működtek, azaz képpontokból (pixelekből) álló képek feldolgozását végezték.

A számítógépes grafikában azokat a raszteres, azaz képpontokból (pixelekből = picture element) *álló képet generáló és feldolgozó rendszereket, melyeknél a képi információ csak képenként kereshető vissza, és a kép tartalma csak a teljes kép felülírásával módosítható, rasztergrafikus rendszereknek nevezzük.*

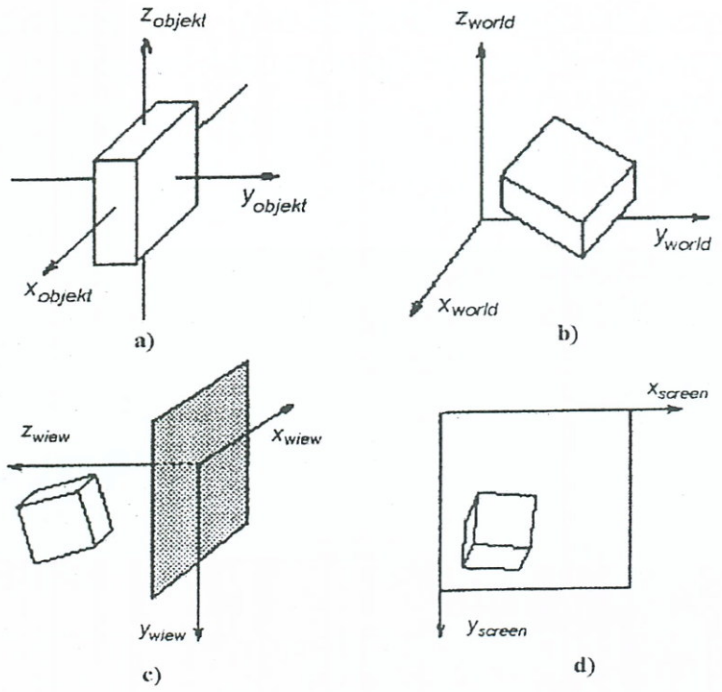
Ezekkel viszonylag egyszerű objektumokat (négyzet, kör stb.) meg lehetett jeleníteni képen, és „szabadkézi" rajzokat is előállíthattunk. A fő gondot a rasztergrafikus programok esetében az jelentette, hogy a rasztergrafikus objektumokat csak képként lehetett kezelni, ezekkel nem lehetett egyszerű módon műveleteket végezni (például szerkesztések, nagyítás stb.).

Egyre nagyobb igény mutatkozott a grafikus objektumok adatbázisban történő tárolására. Ez vezetett a vektorgrafika kialakulásához, mely az objektumokat már egy lebegőpontos világkoordináta-rendszerben írja le.

Ebben az „absztrakt" világkoordináta-rendszerben egy pontot a hozzávezető helyvektorral lehet azonosítani, ezért nevezik ezeket a rendszereket vektorgrafikusnak. A vektorgrafikában az objektumok jellemzőit adatbázisban tárolják. Ez lehetővé teszi az objektumok önálló visszakeresését és a köztük lévő strukturális kapcsolatok (például hierarchiák) számítógépes feldolgozását. A felhasználó a vektorgrafikus rendszerben új objektumokat hozhat létre, ezeket transzformálhatja a világkoordináta-rendszerben (pl. forgatás, zoom stb.).

A számítógépes grafikában azokat a rendszereket, melyek a grafikus objektumokat egy lebegőpontos világkoordináta-rendszerben modellezik, vektorgrafikus rendszereknek nevezzük.

A vektorgrafikus rendszerekben objektum, világ, nézet és megjelenítési koordináta-rendszerekben írjuk le a grafikus objektumokat (3. sz. ábra).



3. sz. ábra
A vektorgrafikus rendszerek a) objektum, b) világ, c) nézet és d) megjelenítési koordináta-rendszerei

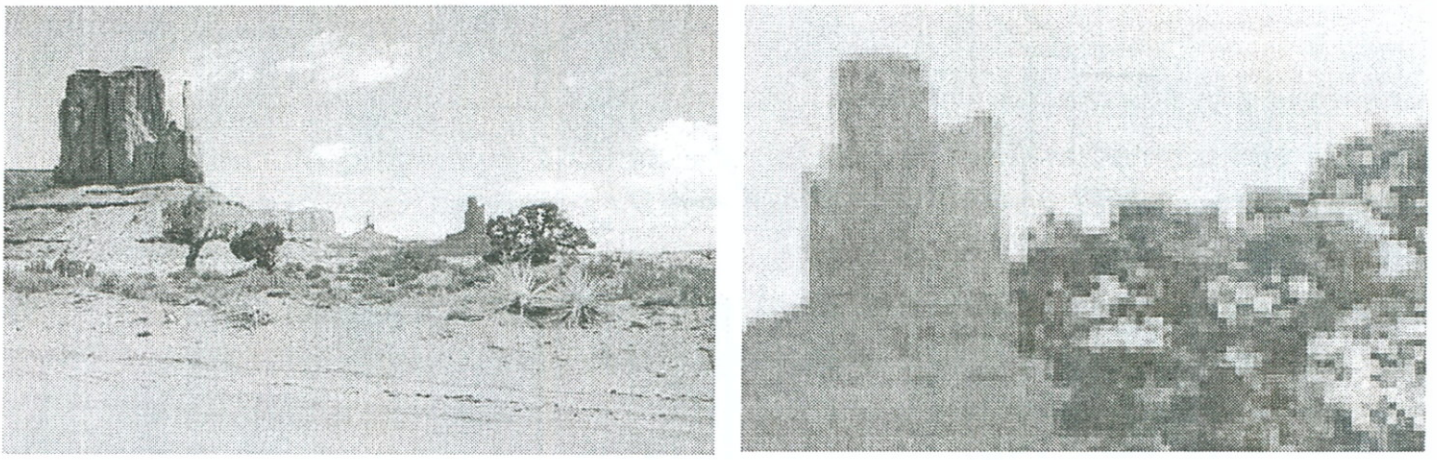
A rasztergrafika és vektorgrafika közötti legfontosabb különbség a grafikus objektumok képenkénti, illetve adatbázisszerű kezelése (4. sz. ábra).

RASZTERGRAFIKA	VEKTORGRAFIKA
pixelekből álló képet tároljuk, csak a kép kereshető vissza, a képen látható objektumok nem	az objektumokat önállóan tároljuk, ezek egyedileg is visszakereshetők

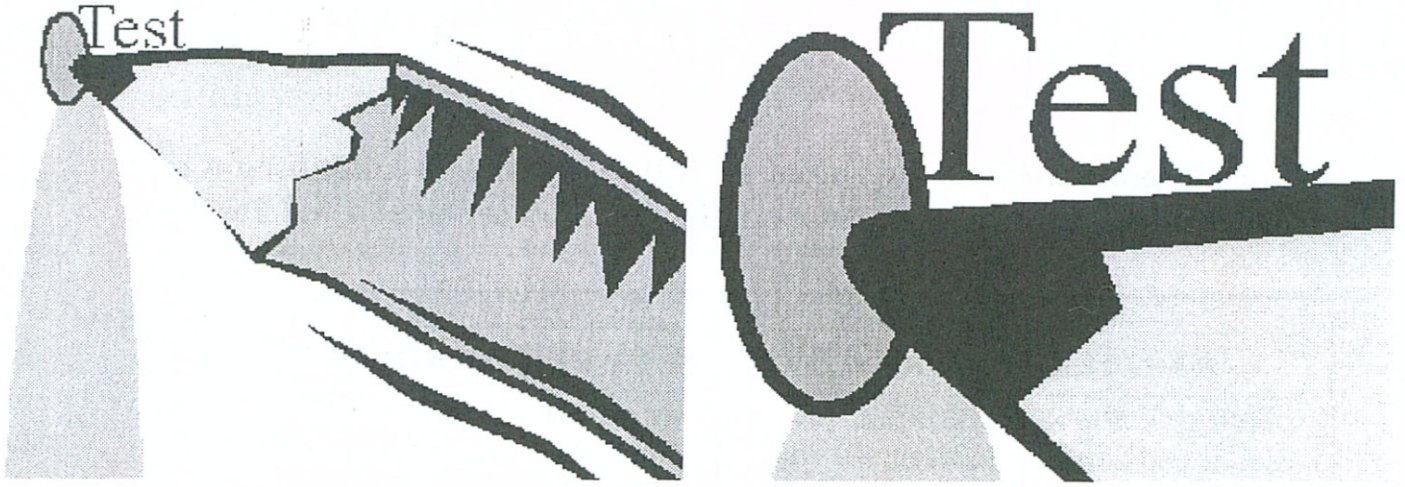
4. sz. ábra
A vektorgrafika és rasztergrafika különbsége

A képpontos és vektoros modelleken alapuló feldolgozás különbségét jól érzékelhetjük, ha a képen transzformációkat alkalmazunk. Nagyításkor a raszteres kép eltorzul, a vektorgrafikus kép viszont nem (mert a generálási szabályok azonosak, bármilyen mértékegységet is alkalmazunk) (lásd az 5. sz. ábrát). A raszteres kép torzulásának az az oka, hogy a rasztergrafikus rendszerek nagyítása az eredeti kép egy pixelje helyett ennek többszörösét jeleníti meg (például 10x nagyításnál 1 pixel helyett 100 képpontot, de az eredeti képpont formájában).

RASZTERGRAFIKA
(A nagyításnál a kép eltorzul.)



VEKTORGRAFIKA
(A nagyításnál nem torzul a kép.)



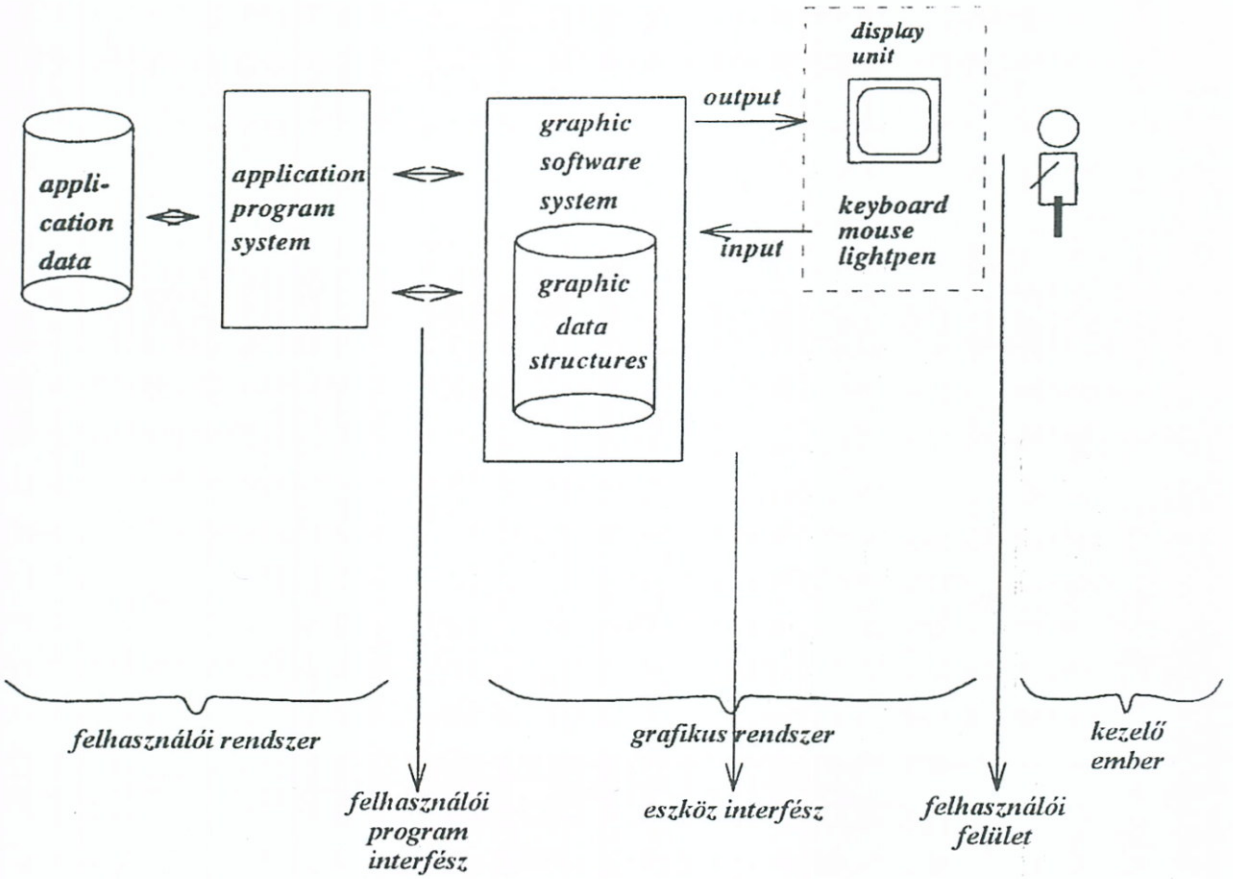
5. sz. ábra
A vektor- és rasztergrafikus kép különbsége nagyításkor

1.3. GRAFIKUS RENDSZEREK ARCHITEKTÚRÁJA, GRAFIKUS ADATSTRUKTÚRÁK ÉS ADATTÍPUSOK

Az első géptervezést segítő piaci CAD/CAM (számítógéppel segített tervezés és gyártás) rendszerek az autó és repülőgépgyártás (General Motors, Lockhead) igényei szerint kerültek kifejlesztésre a 70-es években. Ezek a

rendszerek – amellett, hogy igen drága képmegjelenítő eszközöket és a nagy számításigényű feldolgozást elvégezni képes mainframe gépeket igényeltek – készülékspecifikusak voltak, azaz egy új grafikus periféria esetén a szoftvert részben vagy teljes egészében át kellett írni. A szoftvergyártó cégek hamar felismerték, hogy a különböző hardver környezetre való implementációk költségének csökkentése miatt szabványosítani kell a programcsomagoknak az általános, a konkrét hardvertől függetlenné tehető részét. Így jöttek létre a szabványosított grafikus programcsomagok (graphic software system), melyek meghatározott nemzetközi vagy ipari szabványnak (pl. GKS, PHIGS, OpenGL stb. lásd VII. fejezet) felelnek meg.

A grafikus rendszerek ennek megfelelően kialakult architektúráját mutatja a 6. sz. ábra.



6. sz. ábra
A grafikus rendszer részei és a köztük lévő csatolók

Látható, hogy *egy grafikus rendszer három fő részből*

- *az alkalmazásspecifikus felhasználói programrendszerből* (CAD rendszer, térképészeti rendszer stb.),
- *a szabványosított grafikus programcsomagból* és
- *a grafikus hardverből* (pl. monitorvezérlőkártya és monitor)

tevődik össze.

!

Az egyes rendszerrészeket *szabványos interfészek vagy csatolók* kapcsolják össze, ami *által az egyes rendszeregységeket* (pl. újabb verziójú szoftver, vagy fejlettebb grafikus periféria megjelenése esetén) *úgy tudjuk lecserélni, hogy a többi rendszerkomponenst nem kell megváltoztatni.* Ezek a csatolók a következők:

- *a grafikus programcsomag és a felhasználói programrendszer között helyezkedik el a felhasználói programcsatoló vagy API* (application program interface). *Ez általában azt is biztosítja, hogy a grafikus programcsomaggal különböző programnyelven tarthassanak kapcsolatot a felhasználói programok* (language binding);
- *a grafikus programcsomag egy-egy konkrét grafikus perifériával a hardvercsatolón* (device interface) *keresztül tartja a kapcsolatot. Ide tartoznak a különféle eszközmeghajtók* (device driver) *és a BIOS megfelelő része* (video ROM-BIOS);
- *a grafikus rendszerrel a felhasználó egy szabványos grafikus felületen tarthat kapcsolatot. Ez a GUI* (Graphical User Interface), *mely az 1980-as évek második felében vált szabvánnyá* (1985 WINDOWS, X11 és XWINDOWS 1987).

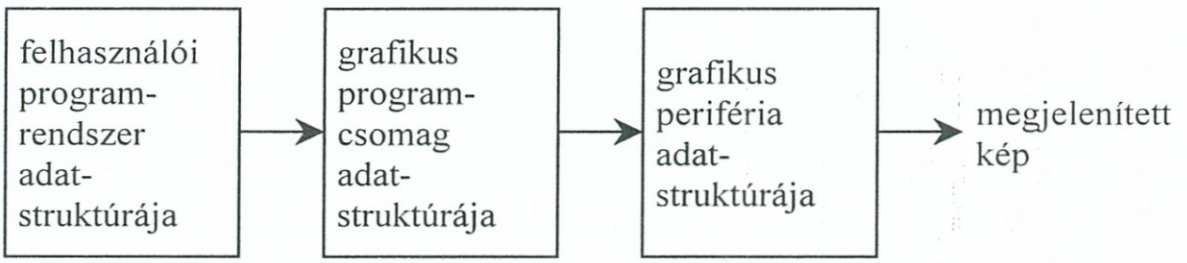
A grafikus rendszer egyes részegységei feladatkörüknek megfelelő típusú és struktúrájú adatokat kezelnek:

- A felhasználói programrendszer ún. modelladatokat dolgoz fel. Ilyenek a grafikus objektum típusa (pl. kör), a grafikus objektum nagysága (sugara 5 cm), a grafikus objektumok helye (középpontja az x=0, y=0, z=0 pontban) és egyéb jellemzői. Emellett a modelladatok strukturális jellegűek is lehetnek (pl. egy objektum más objektumokkal egy csoportot képez, egy objektum része egy másiknak stb.) és nemcsak geometriai jellemzőket tartalmazhatnak (pl. szín, vonaltípus stb.).
- A grafikus programcsomag a képet saját rendszerében kezelt grafikus objektumokból állítja elő. *A legkisebb, a grafikus programcsomag által már további részekre nem bontható elemi grafikus objektumokat primitíveknek nevezzük.* A grafikus programcsomag ezekből a primitívekből állítja elő a komplexebb objektumokat. Példák primitívekre:
  - 3D vektorgrafikában egy térbeli egyenes szakasz,
  - rasztergrafikában egy kör.

A geometriai primitívek mellett a grafikus programcsomagok szöveges primitíveket is tartalmazhatnak.
A grafikus programcsomag az általa kezelt objektumokkal meghatározott műveleteket is képes elvégezni. Ilyenek például: egy új objektum előállítása, objektum transzformálása (például forgatás), objektumok átstrukturálása stb.

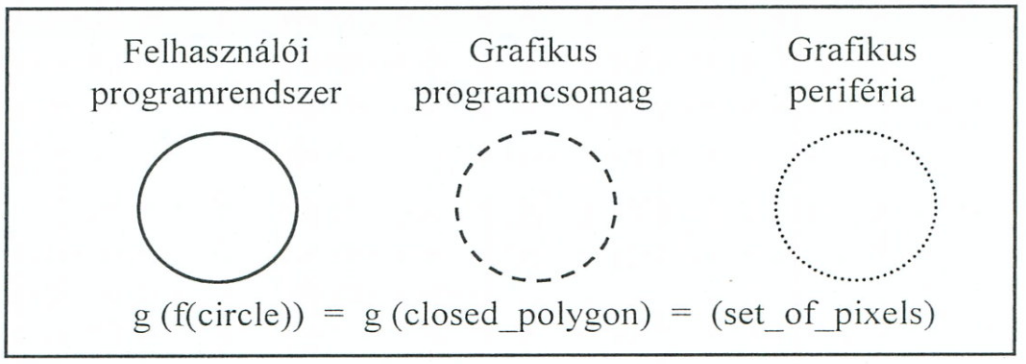
- A grafikus perifériák a raszterpontokra vonatkozó kétdimenziós memóriaterület adatait (frame buffer) jelenítik meg, melynek méretét a képernyő felbontása határozza meg.
  (A raszterpontokat egy kétdimenziós egészértékű koordinátarendszerben lehet kezelni, például egy „move and draw" utasítás fixpontos koordinátaértékek listájával kerül értelmezésre.)

Az előbbiekből nyilvánvaló, hogy *a képernyőn történő megjelenítéshez a felhasználói programrendszer modelladatait először a grafikus programcsomag primitívjeiből felépített objektumokká, majd raszteres képpé kell transzformálni* (lásd 7. sz. ábra).



7. sz. ábra
A modelladatok megjelenítéséhez szükséges transzformációk

A transzformációk során a grafikus struktúra információk egy része elveszik, mint ezt a 8. sz. ábrán is láthatjuk. A felhasználói programrendszer pontos kördefiníciójából a grafikus programcsomag már egy egyenes szakaszokból álló poligont készít, ebből a megjelenítéskor már csak egy pixelhalmaz marad.



8. sz. ábra
Strukturális információveszteség a megjelenítéskor

1.4. A SZÁMÍTÓGÉPES GRAFIKA TIPIKUS FELHASZNÁLÁSI TERÜLETEI

1.4.1. Számítógéppel segített tervezés és gyártás (Computer Aided Design and Manufacturing)

Ezen a szakterületen a számítógépes grafika már több évtizedes múltra tekinthet vissza. A számítógépes tervezőrendszerek alkalmazása egyes felmérések szerint 80% költségmegtakarítást eredményez a hagyományos eljárásokhoz képest.

A CAD/CAM rendszerek (Computer Aided Design and Manufacturing) különösen jelentős szerephez jutnak például az autó- és repülőgépgyártásban, az elektronikus áramkörök (VLSI technológia) és a számítógépek tervezésében és gyártásában. (Gondoljunk például arra, hogy egy 7,5 millió áramköri egységet tartalmazó Pentium-II processzort manuális módon megtervezni gyakorlatilag kivitelezhetetlen lenne.)

A CAD rendszereket természetesen nemcsak a gépgyártásban hasznosítják, legalább ennyire jellemző például felhasználásuk az építészetben.

E rendszerek elterjedése, hatékonyságuk főképpen a következőkkel magyarázható:

- a minőségi szellemi munkát jelentő tervezést mentesítik az automatizálható rutinfeladatoktól,
- a tervek módosítása e rendszerekkel jóval kisebb ráfordítással, kevesebb hibával elvégezhető,
- nem kell költséges prototípusokat megépíteni, a tervek szimulációval jól tesztelhetők,

- az újabb CAD rendszerek lehetővé teszik a megtervezett objektumok valósághű, foto minőségű megjelenítését is. Ez javítja a megrendelő és a tervező kommunikációját. Például egy megtervezett házat a megrendelő 3D szimulációval „bejárhat", megnézhet, az épület valós természeti környezetben is elhelyezhető (lásd 9. sz. ábra).

9. sz. ábra
CAD rendszerrel megtervezett épület képe valósághű környezetbe behelyezve (3D Studio MAX)

1.4.2. Térképészeti információs rendszerek (Geographical Information System)

A számítógépes grafika gyorsan fejlődő területe a térképek számítógépes feldolgozáson alapuló információs rendszerei (GIS = Geographical Information System). Ezek raszteres, vagy fejlettebb változataik (általában digitális képfeldolgozással előállított) vektoros térképadatokat dolgoznak fel, és ezeket összekapcsolják más adatbázisokkal. Néhány példa:

- Egy város elektromos, gáz, csatorna, közlekedési hálózatának nyilvántartása egy grafikával előállított térkép segítségével.

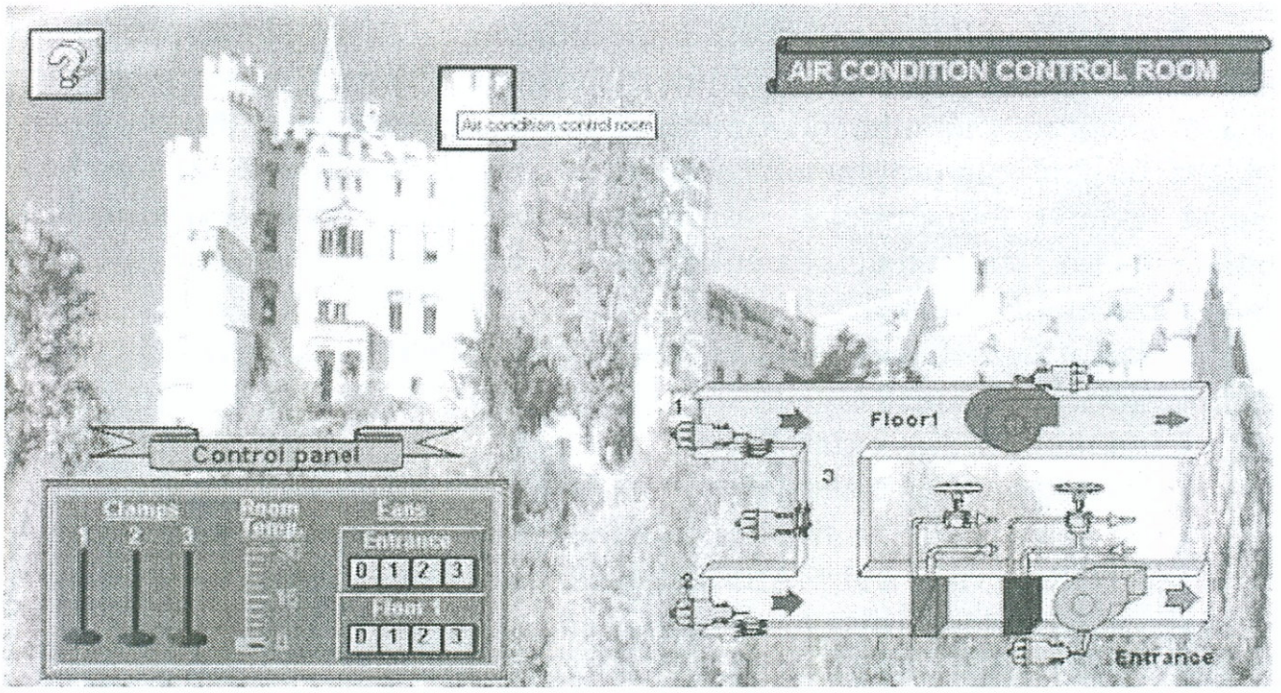
- A rendőrség ún. „bevetés" irányítási rendszere, mely többek között biztosítja például a rendőrjárőr kocsik helyzetének real-time nyilvántartását. Ezt egy vektoros térképpel összekapcsolva lehetővé teszi minden védendő objektumhoz egy bevetési terv hozzárendelését és ezzel például egy bűncselekmény elkövetésekor a menekülési útvonalak lezárását.

1.4.3. Grafikus prezentáció (Business Graphics)

Az üzleti életben, a tudományban és a közigazgatásban sokszor célszerű olyan grafikus rendszereket alkalmazni, melyek képesek meghatározott tendenciákat, összefüggéseket grafikus ábrákon is bemutatni (hisztogramok, oszlopdiagramok stb.). Ezek fokozódó jelentősége arra vezethető vissza, hogy a vizuális információkat az ember sokkal gyorsabban képes felfogni és így a döntéshozatal felgyorsulhat. A számítógépes grafikának ez az alkalmazási területe ezért fontos részét képzi az ún. Vezetési Információs Rendszereknek (Management Information System) is.

1.4.4. Folyamatok felügyeletét segítő számítógépes grafikus rendszerek

A számítógéppel vezérelt folyamatokról az érzékelők a különböző mérési eredmények igen nagy számát továbbítják a felügyelő központba (erőművek vezérlőtermei, közlekedésirányítás, komplex távadatfeldolgozó hálózat menedzselő központja stb.). Az emberi észlelés gyorsasága e rendszereknél egyes esetekben kritikus lehet. Ezért fontos szempont, hogy az eseményeket a kezelőszemélyzet megfelelő grafikus kijelzéssel vizuálisan is érzékelje (villogás, piros szín, hangjelzés stb.) és ezáltal gyors beavatkozásra legyen képes.



10. sz. ábra
Számítógéppel irányított „intelligens" épület diszpécserközpontja

1.4.5. Szimuláció számítógépes grafikával

Számítógépes grafikát szimulációs kiképzésre már kb. 30 éve használnak. Az első ilyen alkalmazások a repülőgép és űrhajószimulátorok voltak, melyekkel a pilóták oktatását hajtották végre. A valósághű jelenetek grafikával történő valós idejű (real-time) megjelenítése igen komoly számításigényt jelent, ezért e rendszerek széles körű elterjedése csak az elmúlt években kezdődött meg. Néhány példa a grafikus szimuláció alkalmazására:

- autóvezetők felkészítése szimulátorban a jeges útviszonyokra,
- nagyon gyors folyamatok modellezése a tudományos kutatásban (ilyen lehet például egy kémiai reakció szimulációja és ennek során kiszámított képek sorozatából egy film előállítása),
- katonai törzsvezetési gyakorlatok, amikor a harci eseményeket számítógéppel szimulálják és grafikusan jelenítik meg,
- katasztrófahelyzetek számítógépes szimulációja, az elhárításban részt vevő egységek kiképzése céljából,
- időjárás előrejelzés készítése számítógépes szimulációval és a légköri mozgások megjelenítése számítógépes térképen.

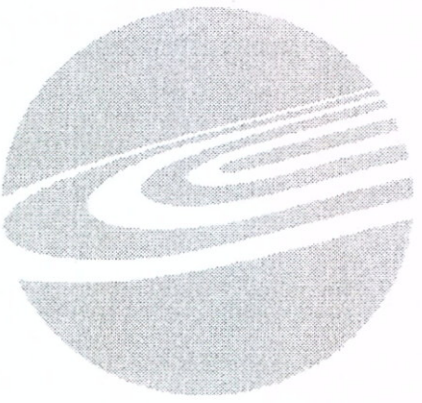
A szimulációhoz készített számítógépes animáció tulajdonképpen a modernebb változata a korábbi rajzfilm trükköknek. Ezeknek egyszerűbb esetei felhasználói szempontból kvázi rajzolt filmkockáknak tekinthetők. A legfejlettebb grafikus programokkal készített animációk viszont ma már hatásukat tekintve teljes értékű, számítógéppel készített szintetikus filmeknek minősíthetők, melyekben például a mozgás megjelenítése azonos minőségű a kamerával felvettekkel.

A számítógépes animációk felhasználási területeit hosszasan lehetne sorolni. Ilyenek például:

- a multimédiás oktatófilmek,
- a reklámfilmkészítés,
- a science-fiction filmkészítés (ennek tipikus példája a TERMINATOR II. vagy a JURASSIC PARK),
- a WEB-lapok díszítése,
- prezentációkészítés stb.

1.4.6. Szöveg- és kiadványszerkesztés (Desk Top Publishing)

A számítógéppel segített nyomdai kiadványszerkesztésben (DTP = Desk Top Publishing) is jelentős szerephez jut a számítógépes grafika. Alkalmazására különösen a speciális képek előállítása, különleges betűtípusok, emblémák, logok és reklámgrafikák elkészítése a jellemző. Egy grafikával készített logot mutat a 11. sz. ábra.



GÁBOR DÉNES
FŐISKOLA

11. sz. ábra
A Gábor Dénes Főiskola logoja

1.4.7. Virtuális valóság és felhasználása

Virtuális valóság alatt elképzelt, vagy méretük, illetve távolságuk miatt láthatatlan, veszélyességük miatt megközelíthetetlen világok valósághű, interaktív modellezését és megjelenítését értjük számítógépen.

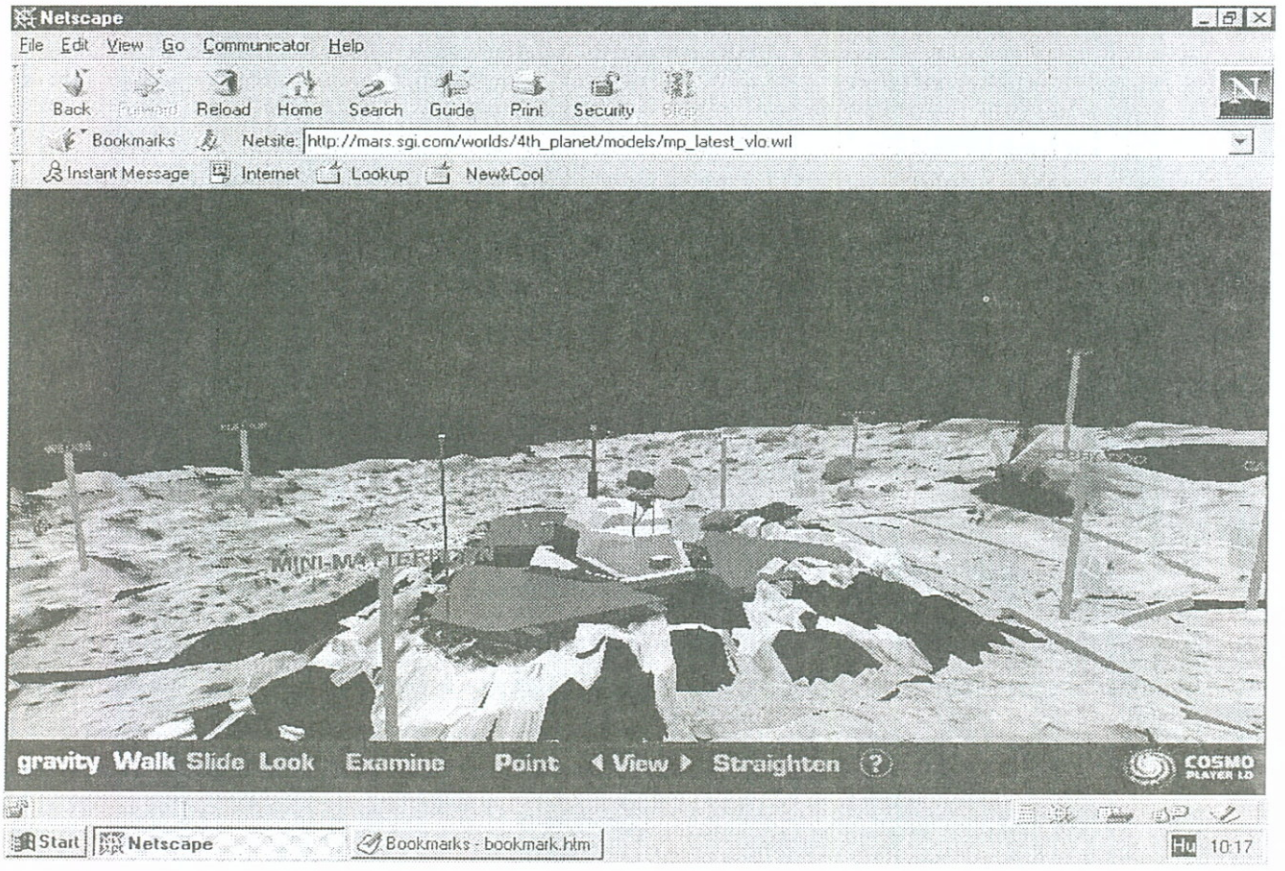
Ezeket a mesterséges, háromdimenziós hatást keltő világokat a rendszert használó ember bejárhatja, felfedezheti. Így például a Mars szonda képei alapján készített virtuális Mars felszínt mutatja be a 12/a., /b sz. ábra.

A virtuális valóságok gyors elterjedését kiváltó fontosabb felhasználói igények a következők:

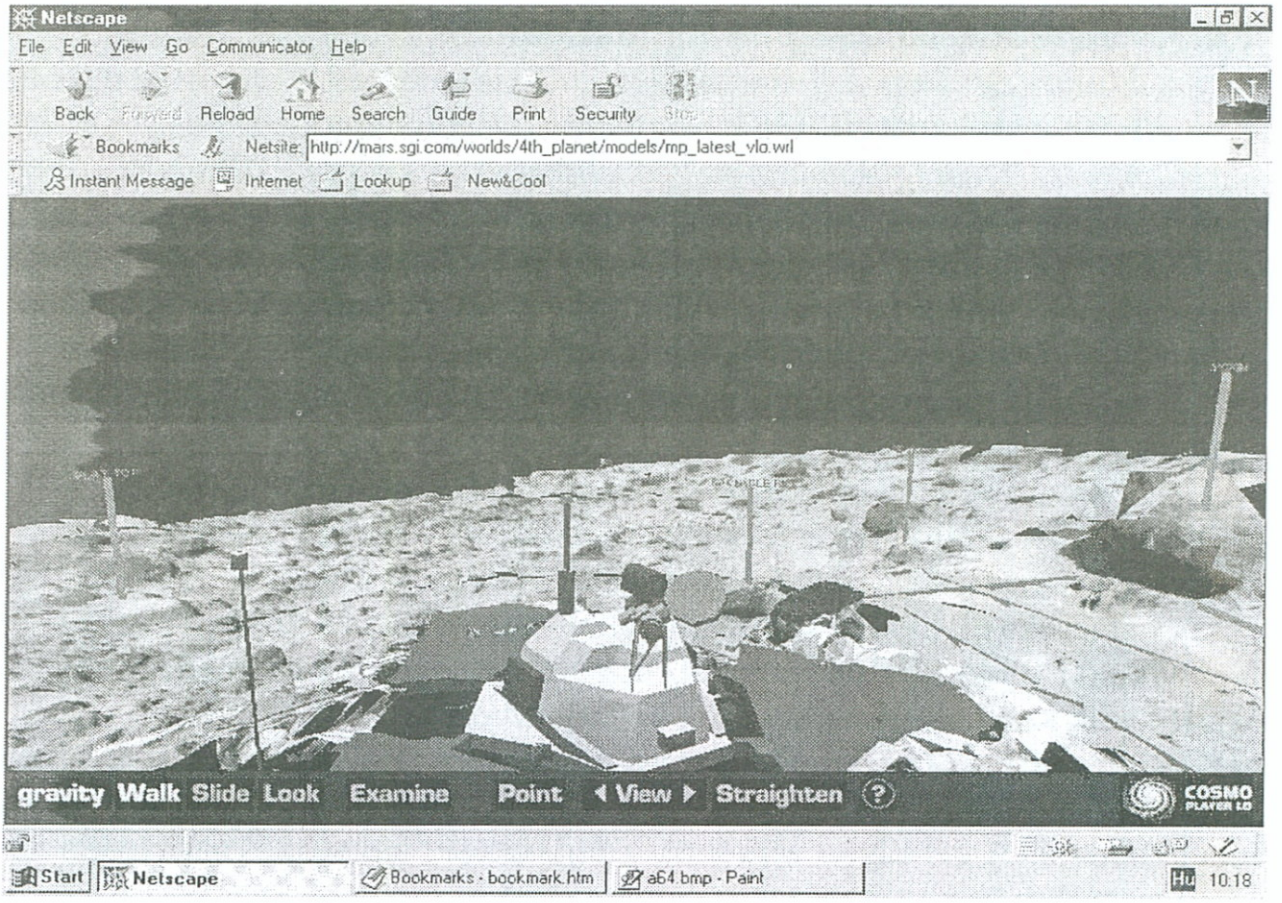
- veszélyes szituációkra való kiképzés (pl. repülőgép baleset),
- valósághű számítógépes játékok,
- filmgyártásban való felhasználás,
- tervek (pl. házak) valósághű bemutatása 3D virtuális bejárással.

Ma már elmondhatjuk például, hogy a virtuális valósággal kapcsolatos WEB lapok száma lassan közelít az egymillióhoz.

A virtuális valóság modellezésének felhasználási területeit a jövőben várhatóan az ember–gép kapcsolat perifériáinak továbbfejlesztése fogja meghatározni. A cél olyan eszközök kifejlesztése, melyekkel a virtuális valóság az ember összes érzékszervére hatni képes.



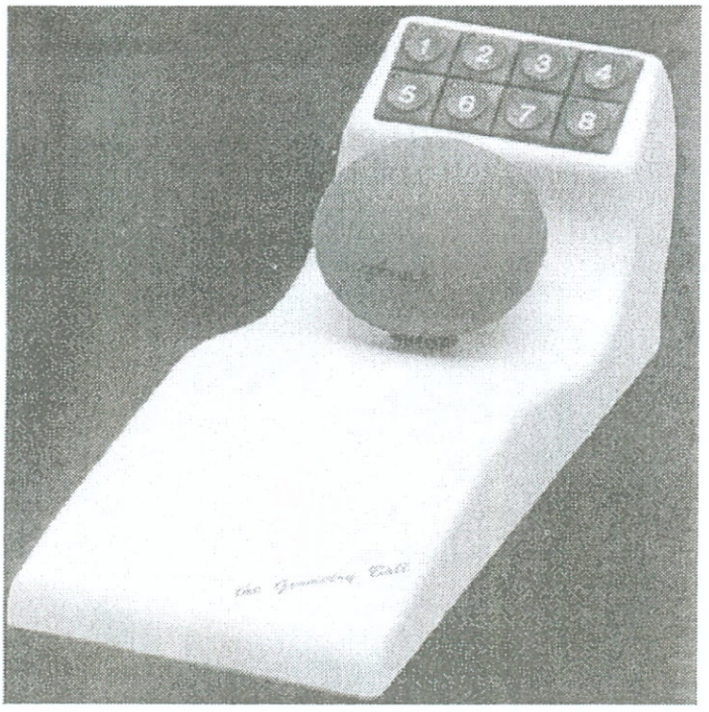
12/a. sz. ábra



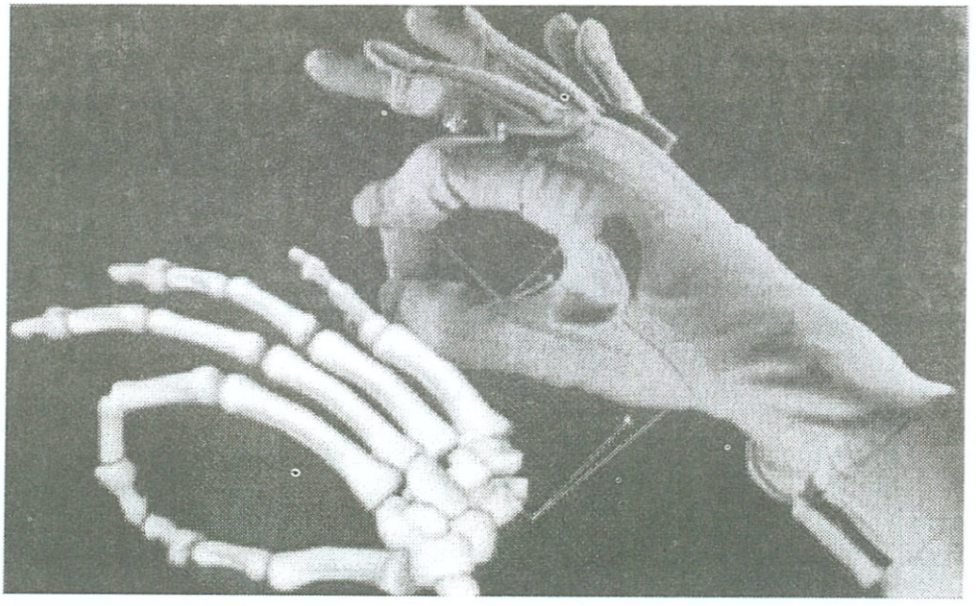
12/b. sz. ábra
A Mars felszíne a virtuális valóságban

Ezek közül az input eszközök közé sorolható a 3D pozíció érzékelő GEOBALL, mely lényegét tekintve az egér 3D megfelelője (lásd a 13. sz. ábra). A GEOBALL működése során fényérzékelők leképezik a golyó 3D koordinátáit a virtuális valóságba.

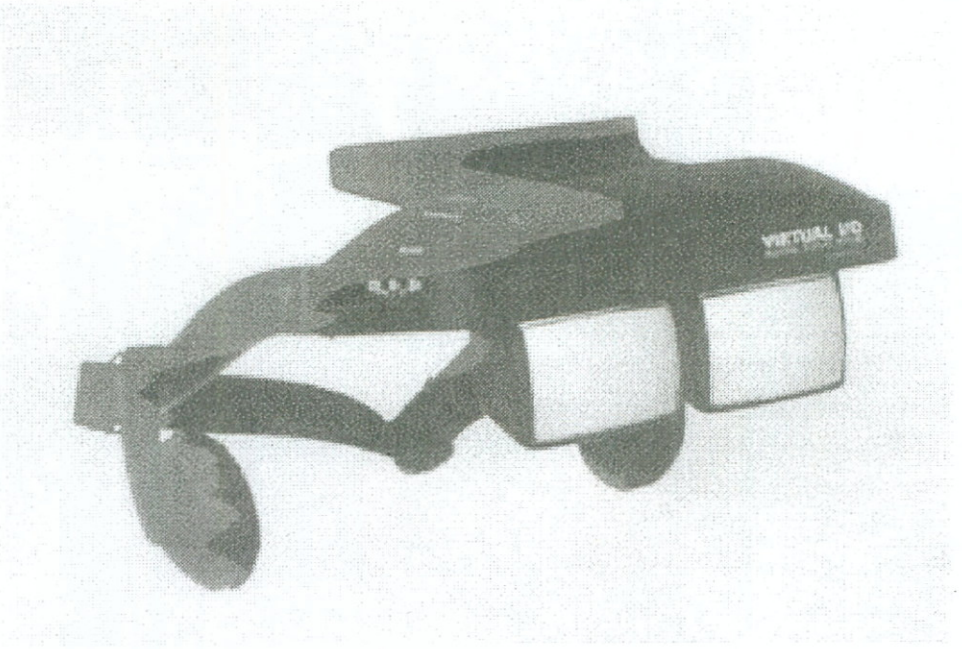
A 3D virtuális valóság manipulációs eszközei közé tartozik a CYBERGLOVE (lásd a 14. sz. ábra). A kesztyűben érzékelők vannak elhelyezve, melyek a kézmozgást közvetítik a számítógépes rendszer számára.



13. sz. ábra
GEOBALL a 3D virtuális valóság „egere"

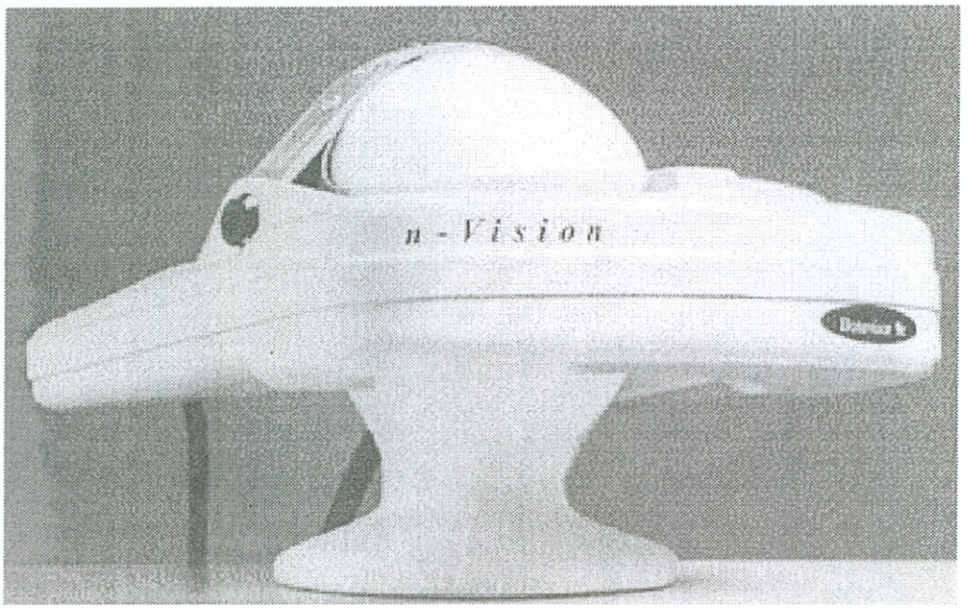


14. sz. ábra
Cyberkesztyű a virtuális valósággal való kommunikációhoz



15. sz. ábra
Virtuális valóság szemüveg

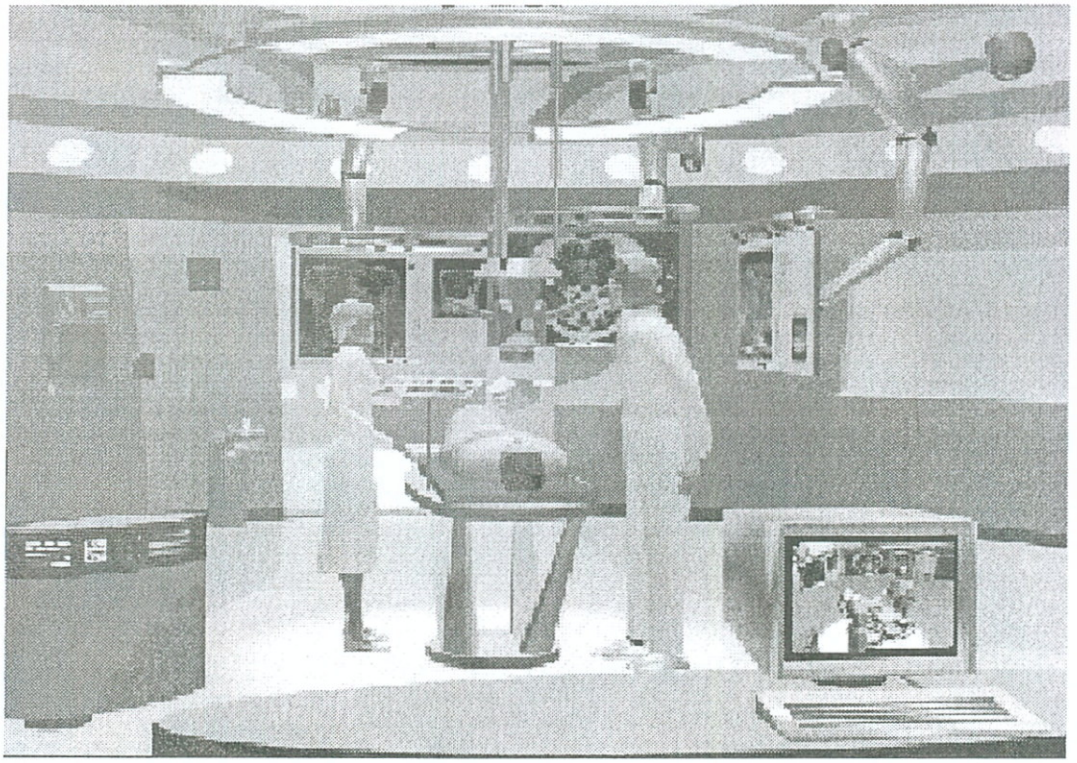
A VIRTUAL I/O két képes szemüveggel vagy a két képet egyidejűleg megjelenítő Head Mounted Display-vel „közvetlenül" érzékelhetjük szemünkkel a virtuális valóságot. A két kép egyidejű nézése miatt a térlátás illúziója tökéletes lehet. Emellett mindkét eszköz lehetővé teszi, hogy a látott képet összekapcsolja 3D (sztereo) hanghatásokkal.



16. sz. ábra

A HEAD MOUNTED DISPLAY két kép egyidejű megjelenítésével éri el a térhatást.

A jövő lehetőségeire jó példát mutat az a tudományos kutatás, mely a sebészek kiképzéséhez virtuális emberi test felhasználását célozza meg (lásd a 17. sz. ábra).



17. sz. ábra
Virtuális emberi test és műtő sebészek kiképzéséhez

1.4.8. Fotorealisztikus képábrázolás

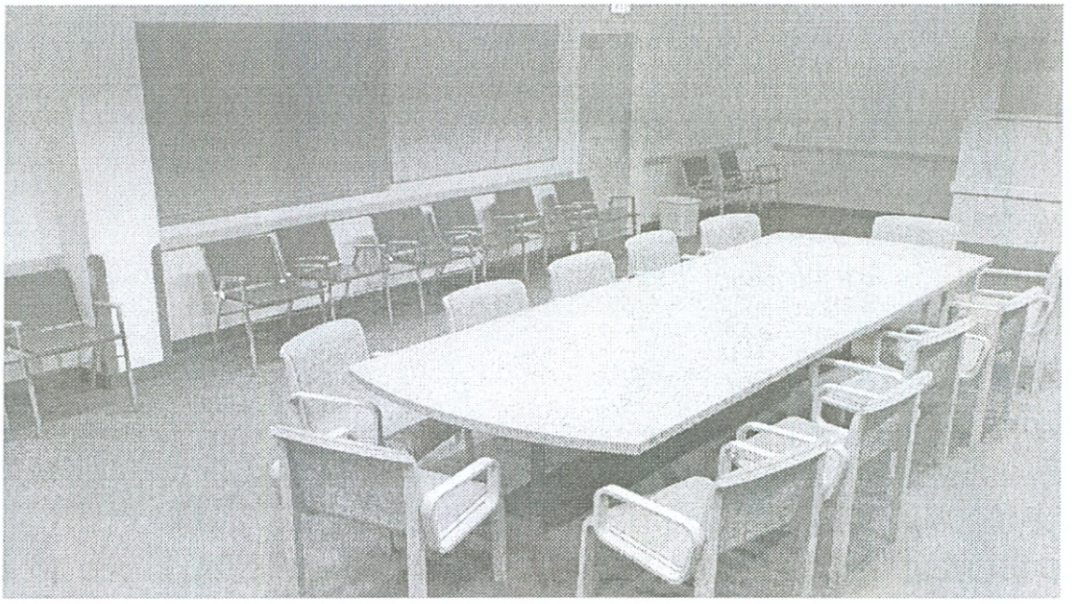
A számítógépes grafika fejlődésében fontos állomást jelentett a képek fotorealisztikus ábrázolása, mely a felhasználói területeket jelentősen kiszélesítette.

Fotorealisztikus képábrázolásról akkor beszélünk, ha a számítógépes grafikával generált képeket gyakorlatilag nem lehet megkülönböztetni a fénykép vagy video-felvételtől. Ennek egyik következménye az is, hogy a számítógépes grafika és multimédia közötti határ elmosódik.

A fotorealisztikus, számítógépes grafikával generált képre jó példa a COREL cég 1996-os rajzolóversenyének győztes képe, melyet a 18. sz. ábrán láthatunk.

18. sz. ábra
Corel Draw rajzolóprogrammal készített kép

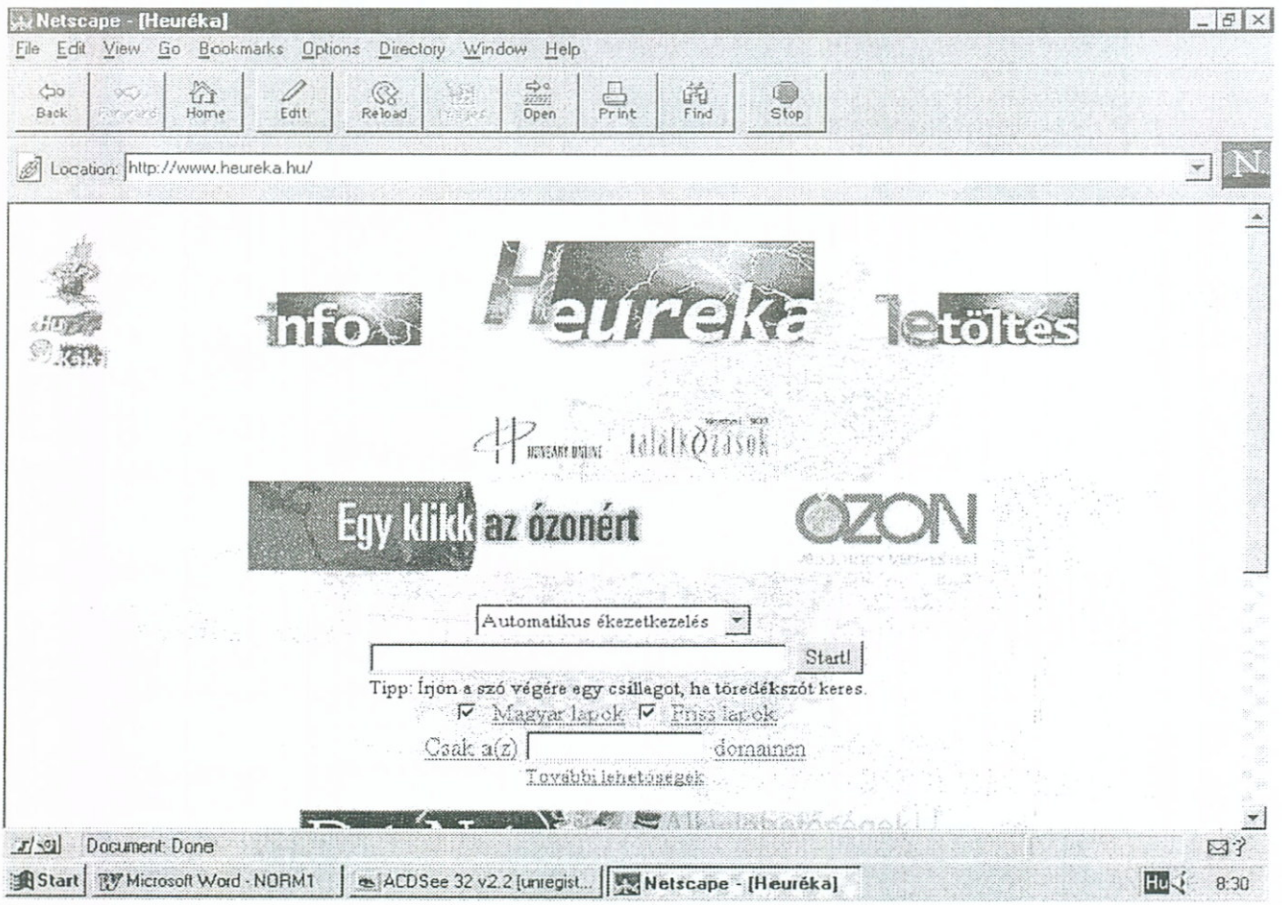
Általában elmondható, hogy a leghatékonyabb grafikus megjelenítő eljárásokkal készített képek minősége sok esetben eléri a professzionális fényképek szintjét. Erre az az ún. radiosity eljárással készített kép szolgálhat példaként, melyet a 19. sz. ábrán láthatunk.



19. sz. ábra
A számítógépes grafika radiosity algoritmusával készített
foto minőségű kép

1.5. AZ INTERNET SZEREPE A SZÁMÍTÓGÉPES GRAFIKA FEJLŐDÉSÉBEN

A számítógépes grafika fejlődésében fontos szerepet tölt be az Internet. A hipermédia tömeges terjedésével a hálózaton a szöveges megjelenítés integrálódik a vizuálissal. Ma már gyakorlatilag nem találunk olyan honlapot a WEB-en, mely ne tartalmazna grafikus és képi linkeket. Ezt mutatja például a magyar keresőnek, a Heurékának a lapja is (20. sz. ábra).



20. sz. ábra
A HEURÉKA magyar kereső lapja a WEB-en
grafikus linkekkel

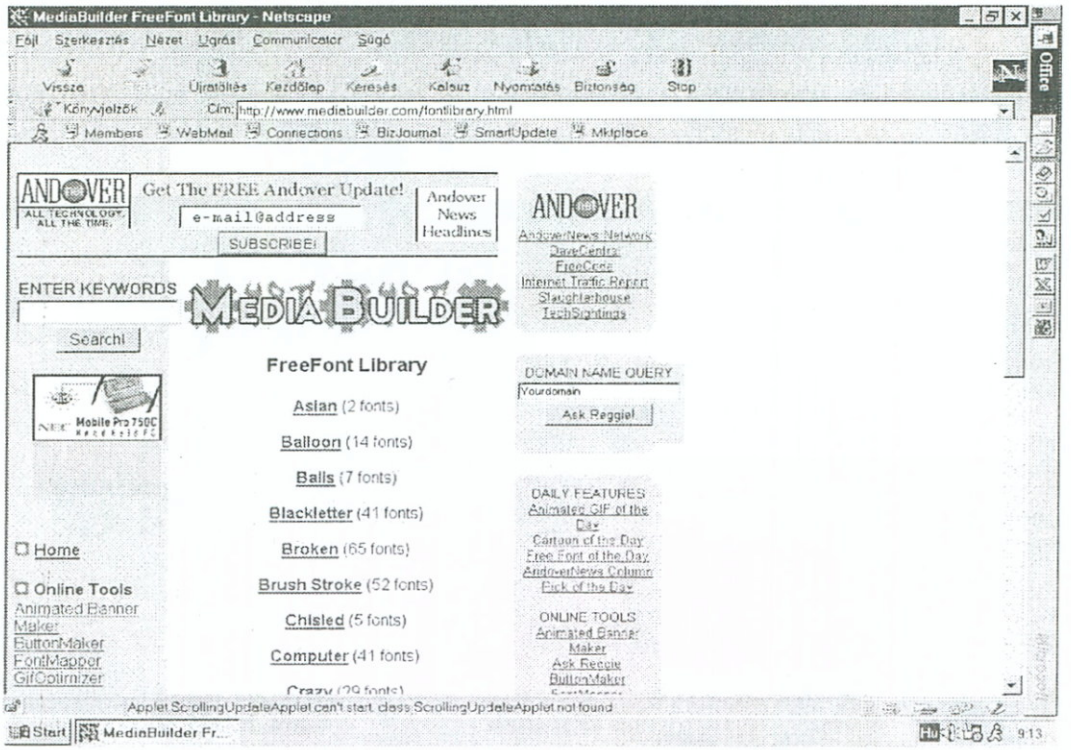
A világhálózat növekvő szerepére a számítógépes grafika fejlődésében további példákat is találhatunk:

- *A jelentősebb grafikus programcsomagokkal ma már kivétel nélkül lehetséges a hálózati csoportmunka, azaz a fizikailag elkülönült helyen dolgozó szakemberek például közös építési terveket készíthetnek.*

!

!

- *Az INTERNET hozzájárul a grafikus adatbázisokban lévő szellemi értékek felhalmozódásához. Példák erre a CLIPART könyvtárak, a grafikus adatbázis elemeket tartalmazó szimbólumkönyvtárak, font-könyvtárak* (lásd 21. sz. ábra).



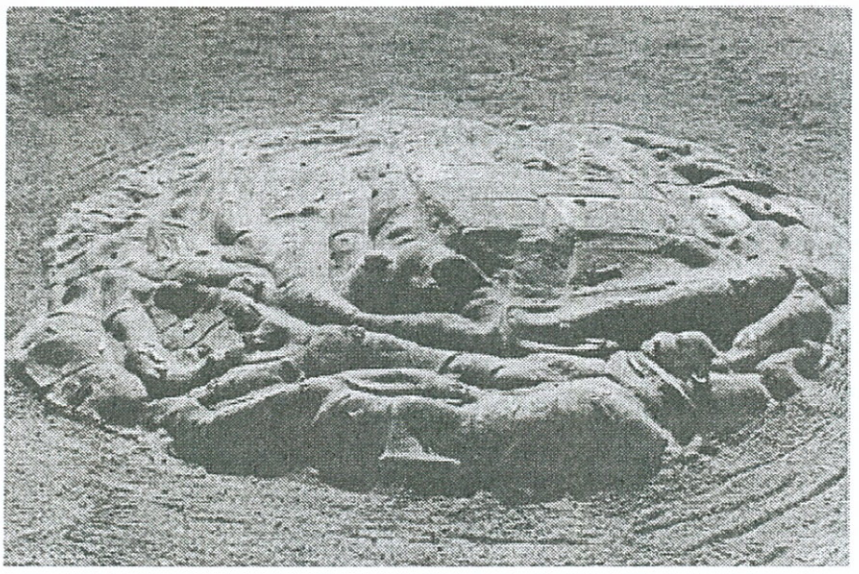
21. sz. ábra
Font könyvtár WEB lapja

!

- *Terjed a Computer Based Training a hálózaton, azaz egyre több a grafikus, multimédiás interaktív oktatóprogram.*
- *A különböző programcsomagok között a képfájl szabványok általános elterjedésével az adatcsere is egyre inkább lehetővé válik a világhálózaton.*
- *A grafikus felhasználói felület szabványossá vált. Emiatt a különböző fontosabb grafikus programcsomagok lényegében azonos kezelő felületet kínálnak a felhasználó számára, ezért egy programcsomag szolgáltatásai könnyen megtanulhatóak.*

A grafikusok, művészek az Interneten is egyre gyakrabban publikálják alkotásaikat. Erre jó példa egy szobrászművész "kiállítása" a Sztaki honlapján (megtalálható a http://www.sztaki.hu/providers/weber/ címen).

A kiállítás egy „szobrát" mutatja be a 22. sz. ábra.



22. sz. ábra

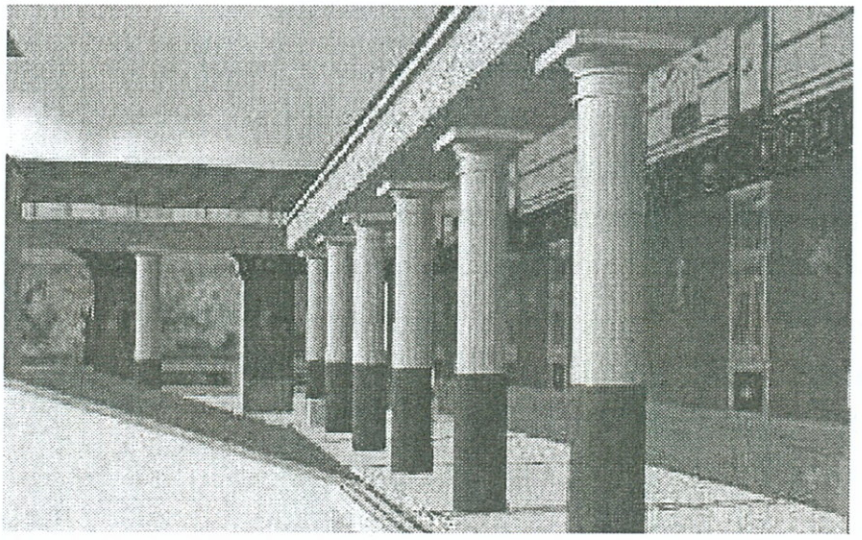
A világháló, a WEB jelentős információforrás is a számítógépes grafikában. Többek között grafikus oktatóanyagokat és dokumentációkat is letölthetünk.

A legfontosabb grafikus hardver és szoftver gyártók szervereit is érdemes tanulmányozni.

A COREL honlapja a http://www.corel.com címen érhető el.

A Silicon Graphics cég honlapját pedig a http://www.sgi.com címen találhatjuk meg.

Az INTERNET virtuális világai néha meglepő információforrások is lehetnek. Erre mutat példát a 23. sz. ábra, mely a lávával elöntött ókori városnak, Pompejinek virtuális rekonstrukciója, „múzeuma".



23. sz. ábra
Pompeji városa virtuális valóságban

Természetesen ahhoz, hogy ez a fejlődés kiteljesedjen, *meghatározott műszaki követelményeknek is meg kell felelnie a hálózatnak.* Így például:

- *a nagy sávszélességű adatátvitelre való képesség;*
- *olyan média szerverek elterjedése, melyek több száz video-stream egyidejű kiszolgálására is képesek;*
- *szabványos, multimédiás és grafikus üzemmódra is alkalmas kliens hardverek és operációs rendszerek használatának általánossá válása.*

1.6. VIZUÁLIS KULTÚRA ÉS AZ INFORMÁCIÓS TÁRSADALOM

Fontos kérdés, hogy az információs társadalomba való átmenet során mi lesz a számítógépes képfeldolgozás és grafika szerepe. Igaz-e, hogy a képi kultúra általánossá válása a szöveges (pl. könyv) információrögzítés, azaz a Gutenberg galaxis végét jelenti?

Néhány lényeges tendencia már ma is előre látható:

- az ember–gép kommunikációs kapcsolat (TV, telefon, számítógép stb.) technikai eszközeinek integrációja várhatóan felgyorsul (eszközintegráció);
- a WEB-en a multimédiás hálózati kommunikáció gyorsan terjed;
- a szabványosítás tendenciája erősödik.

Egyes jövőkutatók szerint évtizedeken belül kialakul a dinamikus információs közeg, mely egységes felhasználói környezetet biztosít a háztartásoktól az irodákig a különböző információk megjelenítésére, tárolására, továbbítására, és helyettesíti az információk papíron megjelenő vagy sugárzott változatait.

Az utóbbi évek felgyorsult hardver teljesítménynövekedése lehetővé tette a számítógépes grafikában olyan rendszerek kialakítását is, melyek a grafikus alakzatokat térbeli „képpontokkal" ábrázolják. Ezt a pixel analógiájára ezt voxel-nek (volume element) nevezték el. Ezáltal egyre tökéletesebbé válik a térbeli alakzatok számítógépes modellezése, mely feltehetően a számítógépes grafikában is korszakváltást fog eredményezni.

? ELLENŐRZŐ KÉRDÉSEK

1. Mi a grafika szó eredeti jelentése és mit értünk e fogalom alatt napjainkban?
2. *Hogyan határozza meg a számítógépes grafika fogalmát a Nemzetközi Szabványügyi Szervezet (ISO)?*
3. Milyen részterületeket különböztethetünk meg a számítógépes grafikában az ISO definíciója szerint?
4. *Mit értünk generatív számítógépes grafika alatt?*
5. *Mi a számítógépes grafika tárgya?*
6. Mit értünk képfeldolgozás alatt?
7. Mit jelent az alakfelismerés?
8. *Mit nevezünk digitális képfeldolgozásnak?*
9. *Definiálja a rasztergrafikus rendszert!*
10. Milyen grafikus rendszer teszi lehetővé a képeken lévő grafikus objektumok önálló visszakeresését?
11. *Mit nevezünk vektorgrafikus rendszernek?*
12. Milyen koordináta-rendszereket használ egy vektorgrafikus rendszer?
13. Milyen különbségeket vehetünk észre egy raszteres és egy vektoros kép nagyításakor?
14. *Sorolja fel milyen elemekből épül fel egy grafikus rendszer!*
15. *Mi a szerepe a szabványos csatolóknak (interface) a grafikus rendszerekben?*
16. *Mit jelent az API, és mi a feladata a grafikus rendszerekben?*
17. *Mi a feladata a hardver csatolónak (device interface), és milyen rendszerelemek tartoznak e csatolóhoz?*
18. *Mit jelent a GUI, és mikor vált szabvánnyá?*
19. Mutasson néhány példát a grafikus felhasználói programrendszer modelladataira!
20. *Mit nevezünk primitívnek?*
21. Mondjon példákat a grafikus programcsomag primitívjeire!
22. Milyen adatformátumot kezelnek a grafikus perifériák?
23. *Milyen transzformációk szükségesek a modelladatok képernyőn történő megjelenítéséhez?*
24. Mi az oka a grafikus információk megjelenítése során fellépő strukturális információveszteségnek?
25. Ismertessen néhány példát a CAD rendszerek alkalmazási területeire!
26. Miért hatékonyabb a mérnöki tervezőmunka egy CAD rendszerrel?

27. Mit jelent a GIS, és milyen két fő típusa van?
28. Miért célszerű az üzleti életben prezentációs szoftvert alkalmazni?
29. Mi a szerepe a számítógépes grafikának a folyamatirányításban?
30. Ismertesse néhány felhasználási területét a grafikus szimulációnak!
31. Milyen területeken hasznosítják a számítógépes animációt?
32. Mire használjuk a számítógépes grafikát a nyomdai kiadványszerkesztésben?
33. *Határozza meg a virtuális valóság fogalmát!*
34. Milyen területeken alkalmazzák a virtuális valóság modellezését?
35. Mi a GEOBALL?
36. Mi a CYBERGLOVE?
37. Hogy működik a VIRTUAL I/O szemüveg?
38. *Mit értünk fotorealisztikus ábrázolás alatt a számítógépes grafikában?*
39. *Mutasson be néhány jellemző tendenciát, mely az Internet és a számítógépes grafika kapcsolatára jellemző!*
40. Mondjon néhány példát a legfontosabb hardver és szoftver gyártók honlapjára!
41. *Milyen műszaki követelményeknek kell megfelelni a világhálózatnak a számítógépes képfeldolgozás tömeges elterjedéséhez?*
42. Mit értünk az eszközintegráció tendenciája alatt az információs társadalomba való átmenet során?
43. Mit jelent a dinamikus információs közeg?
44. Mi a voxel?

II.

MODELLEZÉS, KOORDINÁTA-RENDSZEREK, AFFIN ÉS PERSPEKTÍV TRANSZFORMÁCIÓK, FRAKTÁLGEOMETRIA

E fejezetben először a számítógépes grafika modelltereit és ezeknek a raszteres képen történő megjelenítéssel való összefüggéseit vizsgáljuk meg. Ezt követően a számítógépes grafikában alkalmazott pont és koordináta transzformációkkal foglalkozunk. Végül a fraktálgeometriának azokat a legfontosabb fogalmakat tekintjük át, amelyek a grafikában a valósághű képek generálásához, illetve a fraktálalapú képtömörítéshez nélkülözhetetlenek.

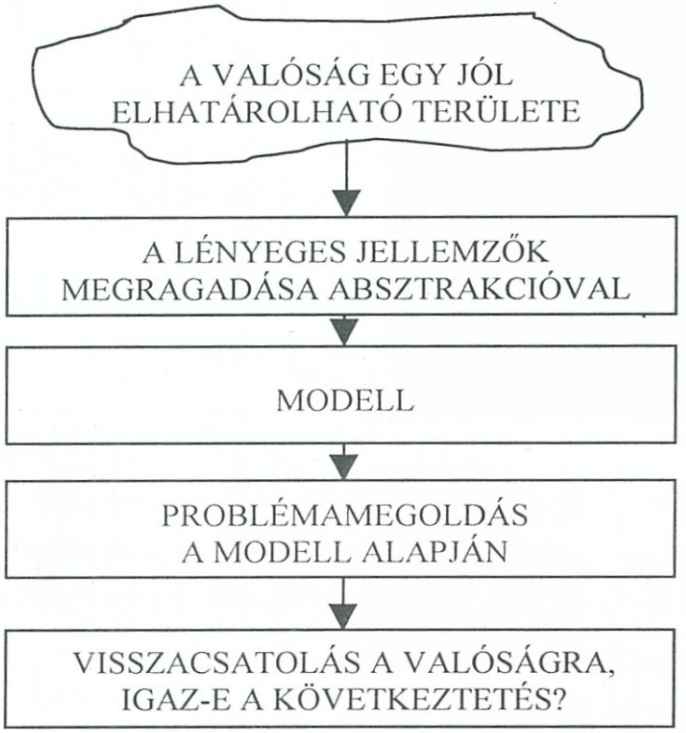
2.1. A SZÁMÍTÓGÉPES GRAFIKA MODELLJEI

2.1.1. A modellezés elve, a modellterek fajtái

A számítógépes grafikában és a képfeldolgozás során nem a valódi objektumokat, hanem azok egy modelljét dolgozzuk fel. *A modellalkotás során megpróbáljuk a grafikus objektum lényegi jellemzőit megragadni, és az így absztrakcióval képzett számítógépes modellt algoritmusokkal dolgozzuk fel.* Ennek összefüggéseit mutatja a 25. sz. ábra.

Látható, hogy a modellnek viszonylagosan önálló „élete" van a számítógépes feldolgozás során. Ezért ügyelnünk kell arra, hogy a modell alapján kapott eredményeket állandóan szembesítsük a valódi objektumokról szerzett tapasztalatainkkal.

Így például, ha matematikai módszerekkel megalkotjuk egy üvegpohár számítógépes modelljét (pl. a pohár felületeit jellemző egyenletekkel), az is fontos követelmény, hogy a „pohár" különböző fényforrásokból történő megvilágítása esetén reális fény- és árnyékhatásokat mutasson a számítógéppel megjelenített kép.



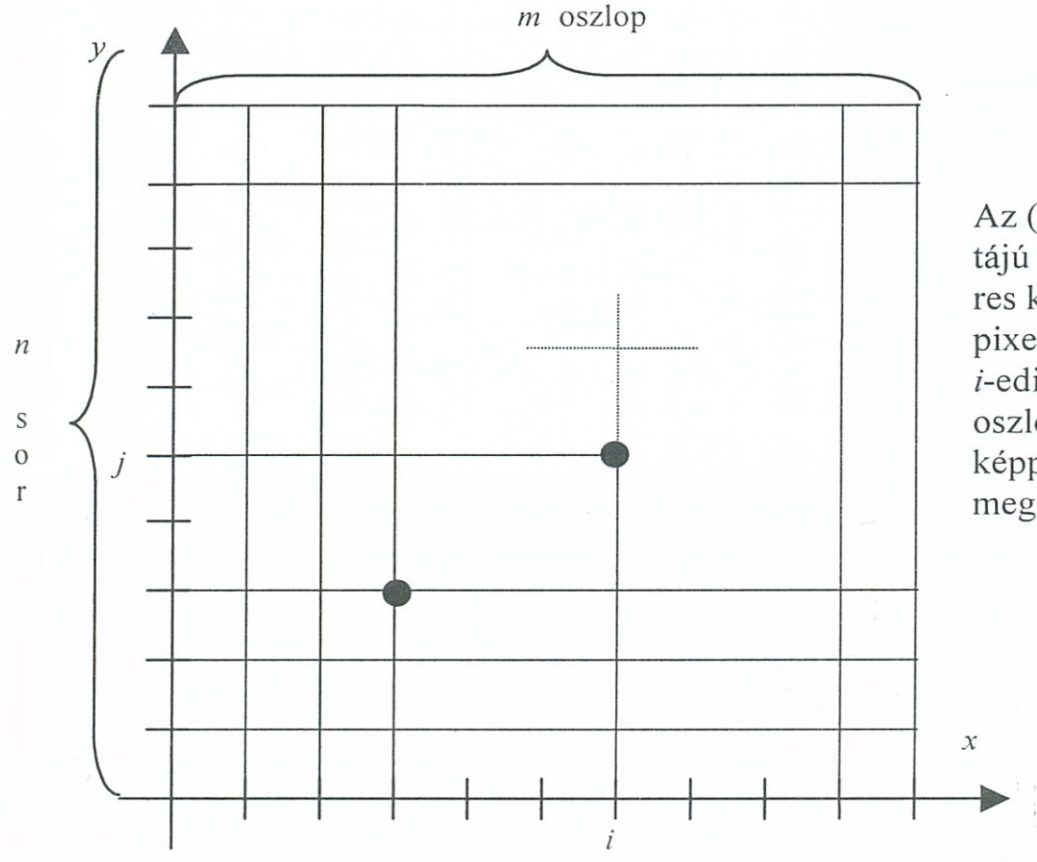
25. sz. ábra
A modellezés elve

! *A számítógépes grafikában feldolgozott grafikus objektumokat* (testek, felületek stb.) *matematikai eljárásokkal modell vagy objektumterekben írjuk le. Ezek általában két- vagy háromdimenziós koordináta-rendszerek, melyekben a grafikus objektumokat matematikai függvények* (pl. egy gömb egyenlete) *és geometriai jellemzők* (pl. egy háromszög három csúcsához mutató vektorok koordinátái) *határozzák meg. A matematikai törvényszerűségekből algoritmusok vezethetők le és paraméterállományok következnek. Ezek számítógépes megfelelői a grafikus szoftverek programrutinjai, fájljai és egyes esetekben a hardver áramkörei* (például a 3D-s gyorsítókártya chipjei).

! *A rasztergrafika modelltere egy kétdimenziós egész koordináta-rendszer, melyben a képpontoknak egészértékű koordinátapontok felelnek meg* (lásd a 26. sz. ábrát).

! *A vektorgrafika modelltere egy két- vagy háromdimenziós euklideszi tér. A grafikus objektumokat egy „végtelen" két- vagy háromdimenziós le-*

begőpontos koordináta-rendszerben ábrázoljuk, kezelésük matematikai eszköze a vektoralgebra. (A „végtelen" természetesen a számítógépen történő lebegőpontos számábrázolás véges határain belül értendő.)



Az (i, j) koordinátájú pont a raszteres kép j-edik pixelsorában és i-edik pixeloszlopában lévő képpontnak felel meg.

26. sz. ábra
A rasztergrafikus képek modellezése egész-koordináta-rendszerben

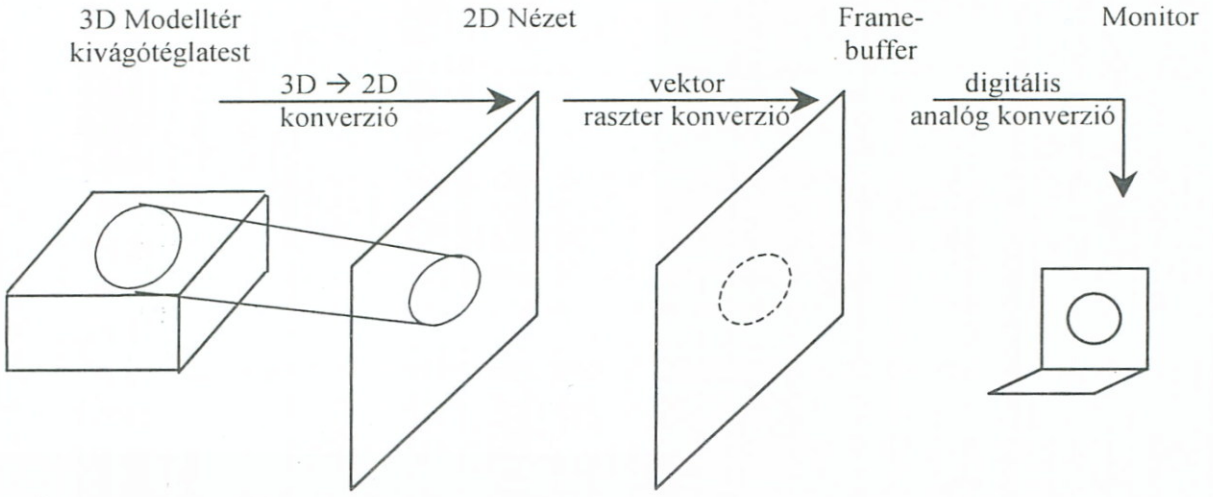
A modelltérben definiált grafikus objektumoknak tehát nincs geometriai megfelelőjük a számítógépen, ezek csak a matematikai összefüggéseknek megfelelő rendszerelemek számára, pl. mint paraméterállományok léteznek.

A vektorgrafikus objektumok képernyőn történő megjelenítéséhez először azt kell eldönteni, hogy a 3D-s végtelen modelltér melyik részét akarjuk a képen látni. A legtöbbször használt eljárás esetén egy kivágótéglatestet hozunk létre, mely a képernyőn szerepeltetni kívánt objektumokat tartalmazza.

Ezután a vektorosan kezelt grafikus objektumokat képi megjelenítésükhöz le kell képezni perspektíva vetítéssel egy kétdimenziós nézetre hasonló módon, mint ahogy a fényképezőgép objektíve előállítja a képet a filmkockán.

A nézetet a képernyőn történő megjelenítéshez ezután vektoros modellből raszteres képpé kell konvertálni és a frame buffer-be be kell tölteni.

A frame buffer adatait olvasva állítja elő a monitorvezérlőkártya digitális-analóg konvertere azt az analóg jelet, mely vezérli a monitor elektronágyúit a kép kirajzolása során.



27. sz. ábra
Vektorgrafikus objektumok képi megjelenítésének lépései

A vektorgrafikus modelltér és a vektor–raszter konverzió szétválasztása többek között azzal az előnnyel is jár, hogy a modelltérben definiált 3D-s objektumokkal a megjelenítéstől függetlenül is végezhetünk műveleteket (törlés, másolás, forgatás stb.). Ezeket letárolhatjuk és különböző képek generálásakor ismételten felhasználhatjuk.

2.1.2. A 2D-s és 3D-s modellterek jellemzői

A 2D-s vektorgrafikus modellterekben a grafikus objektumokat csak két dimenzióban írják le. Ezekben a rendszerekben a vektorgrafikus objektumok tulajdonképpen 3D-s objektumok különböző nézetei. Ezeket lehet nagyítani, törölni stb. A lényeges különbség a valódi 3D-s objektumkezeléssel szemben abban van, hogy ha egy 3D-s objektumot különböző nézetekből kell megjeleníteni, akkor ezek mindegyike a 2D-s rendszerben önálló objektum. (A 3D-s objektumból a 3D-s rendszerben természetesen csak egy van.)

A 2D-s modellezés eljárásait egyes rendszerekben bővítik az ún. 2 ½ D-s modellezés lehetőségeivel, mellyel meghatározott korlátok között a 3D-s testeket is lehet kezelni. Mindenképpen hangsúlyozni kell, hogy *a modelltér e rendszereknél is csak 2D-s, és a 3D-s testek jellemzését az objektumokhoz rendelt, a harmadik koordinátára vonatkozó tulajdonságokkal oldják meg.*

Így például, ha az xy síkban lévő háromszöghöz hozzárendeljük a z=5 magasságértéket, egy háromszögalapú hasábot kapunk. Ekkor azonban a tényleges vektorgrafikus objektum a háromszög, a z=5 magasságérték csak hozzárendelt tulajdonság. Emiatt például a 2 ½ D rendszerekben az előbb említett „hasábot" nem lehet a térben elforgatni.

A valódi 3D-s modellterek objektumait a számítógépen belül teljes értékű 3D-s alakzatokként kezeljük. Ezeknek az objektumoknak előállíthatjuk különböző irányú nézeteit, ezek képernyőparancsokkal interaktív módon feldolgozhatók.

Így az olyan tipikus 3D-s műveletek, mint az eltolás, forgatás, tükrözés, másolás a képernyőn való megjelenítéstől teljesen függetlenül végrehajthatók a 3D-s modelltér objektumain.

2.2. VILÁGKOORDINÁTA-RENDSZEREK ÉS TRANSZFORMÁCIÓK

2.2.1. Koordináta-rendszerek

A koordináta-rendszereket általában a következő feladatokra alkalmazzuk a számítógépes grafikában:

- a modelltérben koordináta-rendszerekben írjuk le a testeket, íveket, felületeket,
- a koordináta-rendszerek közötti (például akkor, ha a nézőpontot változtatunk az ábrázolandó tárgyhoz képest) transzformációkra,
- a képernyőn történő megjelenítéshez a háromdimenziós tér vetítésére a kétdimenziós nézetre.

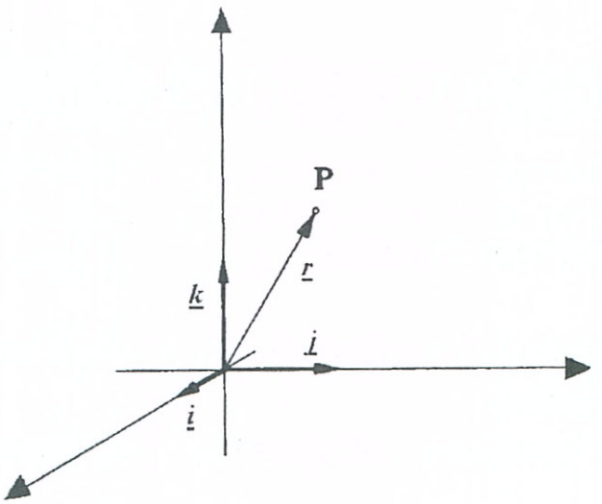
2.2.1.1. A koordináta-rendszer fogalma

Ha matematikai formában akarjuk megadni egy tárgy egyes pontjainak térbeli elhelyezkedését, akkor ehhez valamiféle megállapodásra van szükségünk: vonatkoztatási pontot kell kijelölnünk, léptéket kell megadnunk stb. E megállapodás szabályainak összessége az, amit *koordináta-rendszernek* nevezünk.

Ha rögzítettük koordináta-rendszerünket, a tárgy bármely pontjának térbeli helyzetét néhány (két dimenzióban kettő, három dimenzióban három) számérték segítségével adhatjuk meg. E számok a pont *koordinátái*. A koordinátákat három dimenzió esetén x, y és z-vel fogjuk jelölni.

2.2.1.2. A Descartes-koordináta-rendszer

Ez a leggyakrabban alkalmazott koordináta-rendszer. Szokásos ábrázolását a 28. sz. ábra szemlélteti. A koordináta-rendszer *tengelyeit* három egymásra merőleges egyenes alkotja. Ezek közös pontja az *origó*. Az origóból kiinduló, s a tengelyek irányába mutató, egymásra kölcsönösen merőleges három egységvektort nevezzük a koordináta-rendszer *alapvektorainak*. Jelölésük rendre $\underline{i}, \underline{j}, \underline{k}$.



28. sz. ábra
Descartes koordináta-rendszer

Az origótól a vizsgált P pontig húzott r vektort a ponthoz tartozó *helyvektor*nak nevezzük. Ez a vektor mindig előállítható a három alapvektor lineáris kombinációjaként:

$$\underline{r} = x\underline{i} + y\underline{j} + z\underline{k}$$

Ha a fenti összefüggés teljesül, akkor a P pont és az $\underline{r}$ vektor koordinátái (x, y, z).

2.2.2. Térelemek egyenletei és metszésük

Egy háromdimenziós koordináta-rendszerben minden koordinátahármas a tér egy pontját határozza meg. Ha a koordinátahármasok kiválasztásában megkötéseket kell betartanunk, a tér összes pontja helyett a pontoknak csak egy részét vehetjük figyelembe. Egyetlen

$$f(x, y, z) = 0$$

alakú egyenlet eggyel általában kisebb dimenziószámú térre korlátozza lehetőségeinket: a háromdimenziós tér helyett egy felületre. Két megkötésnek

$$g_1(x, y, z) = 0$$
$$g_2(x, y, z) = 0$$

megfelelő pontok kettővel kisebb dimenziószámú alakzaton helyezkednek el, vagyis legtöbbször egy térgörbén.

2.2.2.1. Egyenes paraméteres egyenlete

Az egyes térelemek egyenletét paraméteresen is megadhatjuk. Például, ha $\underline{v}$ egy egyenes irányú vektor és $\underline{p}$ az egyenes egy pontjához vezető vektor, akkor az egyenes paraméteres egyenlete

$$\underline{r} = \underline{p} + t\underline{v}$$

Koordinátákban:

$$x = p_x + t \cdot v_x \qquad \underline{v} = (v_x, v_y, v_z)$$
$$y = p_y + t \cdot v_y \qquad \underline{r} = (x, y, z)$$
$$z = p_z + t \cdot v_z \qquad \underline{p} = (p_x, p_y, p_z)$$

2.2.2.2. Sík egyenlete

Ha $\underline{r}$ a sík egy tetszőleges pontjához mutató helyvektor, $\underline{p}$ a sík egy adott pontjához vezető vektor és $\underline{n}$ a sík normálvektora (azaz egy tetszőleges, a síkra merőleges vektor), akkor

$$\underline{n} \cdot (\underline{r} - \underline{p}) = 0$$

a sík vektoros egyenlete. (A vektorok szorzása skaláris szorzatként értelmezendő.)

Koordinátákban:

$$n_x (x - p_x) + n_y (y - p_y) + n_z (z - p_z) = 0$$

Ha $\underline{a}$, $\underline{b}$, $\underline{c}$ a sík három adott pontjához vezető helyvektor, akkor a sík paraméteres egyenlete:

$$\underline{r} = \underline{a} + s (\underline{b} - \underline{a}) + t (\underline{c} - \underline{a})$$

Koordinátákban:

$$x = a_x + s (b_x - a_x) + t (c_x - a_x)$$
$$y = a_y + s (b_y - a_y) + t (c_y - a_y)$$
$$z = a_z + s (b_z - a_z) + t (c_z - a_z)$$

2.2.2.3. Gömb egyenlete

Az r sugarú origó középpontú gömb egyenlete:

$$x^2 + y^2 + z^2 = r^2$$

A különböző térelemek metszésének meghatározása (például egyenes döféspontja síkkal, sík metszete gömbbel stb.) egyenleteik alapján egyenletrendszerek megoldására vezethető vissza. Ezt egy példán mutatjuk be:

2.2.2.4. Egyenes döféspontja síkkal

Legyen adva egy egyenes a két pontjához vezető $\underline{a}$ és $\underline{b}$ vektorral, egy sík pedig a három pontjához vezető $\underline{c}$, $\underline{d}$ és $\underline{e}$ vektorral.

Ekkor

$\underline{r} = \underline{a} + t (\underline{b} - \underline{a})$ (egyenes paraméteres egyenlete)
$\underline{r} = \underline{c} + s (\underline{d} - \underline{c}) + v (\underline{e} - \underline{c})$ (sík paraméteres egyenlete)

A döféspontra

$$\underline{a} + t (\underline{b} - \underline{a}) = \underline{c} + s (\underline{d} - \underline{c}) + v (\underline{e} - \underline{c})$$

Koordinátákban

$$a_x + t\,(b_x - a_x) = c_x + s\,(d_x - c_x) + v\,(e_x - c_x)$$
$$a_y + t\,(b_y - a_y) = c_y + s\,(d_y - c_y) + v\,(e_y - c_y)$$
$$a_z + t\,(b_z - a_z) = c_z + s\,(d_z - c_z) + v\,(e_z - c_z)$$

azaz a t, s, v ismeretlenekre három egyenletet kaptunk, melyből a döféspont (ha az egyenes nem párhuzamos a síkkal) meghatározható.

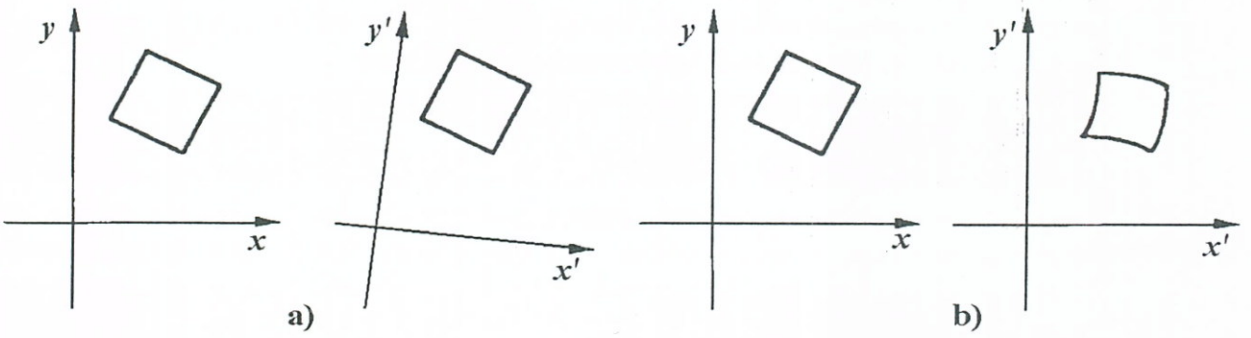
2.2.3. Transzformációk

Legáltalánosabb formában egy 3D-s transzformációról akkor beszélünk, ha tér $\underline{r} = (x, y, z)$ helyvektoraihoz az $\underline{r}' = (x', y', z')$ helyvektorokat az

$$x' = f_1(x, y, z),$$
$$y' = f_2(x, y, z),$$
$$z' = f_3(x, y, z)$$

összefüggések szerint hozzárendeljük.

Erre mutat egy példát a síkban a 29. sz. ábra.



29. sz. ábra
Koordináta- és ponttranszformáció

A $\underline{r}$ vektoroknak megfelelő pontokat tárgypontoknak, az $\underline{r}'$ vektoroknak megfelelő pontokat pedig képpontoknak nevezzük. A számítógépes grafikában két olyan transzformációval találkozunk, amelyek matematikailag teljesen azonos formában (az előbbi alakú transzformációs összefüggés segítségével) írhatók le, ugyanakkor lényegüket (értelmezésüket) tekintve eltérőek. Ezek: a koordinátatranszformáció és a ponttranszformáció.

! *Koordinátatranszformációról akkor beszélünk, ha a tárgypontok egy új koordináta-rendszerre vonatkozó koordinátáit határozzuk meg, a régiek ismeretében. Ilyenkor tehát a vizsgált tárgy változatlan, csupán nézőpontunkat változtatjuk meg.* Ezt szemlélteti a 29. sz. ábra a) része.

Koordinátatranszformációnál a grafikus objektum változatlan (nem torzul, nem változtatja meg az alakját) marad. Ilyen transzformációra például akkor van szükség, ha a nézőpontunkat változtatjuk a 3D-s térben.

! *Ponttranszformációról akkor beszélünk, ha a grafikus objektumhoz annak valamilyen értelemben vett hasonmását rendeljük. Tipikus példa erre például a fényképezés, ahol a 3D-s tárgyak egyes pontjaihoz egy 2D-s kép pontjait rendeljük hozzá. Ide értjük továbbá a testek elforgatását, elmozgatását stb. is.*

Ponttranszformáció esetén a transzformációs összefüggés a tárgy és a kép megfelelő pontjainak egymáshoz rendelését adja. A 29. sz. ábra b) részén egy négyzet leképezését látjuk görbeoldalú „négyzetté".

A térbeli tárgyaknak a számítógépes grafikában való feldolgozása során koordináta- és ponttranszformációt is alkalmazunk. Előbbi jellegzetesen a tárgyhoz képest elfoglalt nézőpontunk változtatása esetén szükséges, utóbbi pedig a testek különböző mozgatásainak, nagyításának, vetítésének matematikai leírására szolgál.

Ha az f_i függvények egyértékűek és értelmezési tartományuk a teljes tér, a transzformációs összefüggések bármely $\underline{r}$ vektorhoz egy $\underline{r}'$ vektort rendelnek. Kérdés, hogy a tárgypontokból egyértelműen visszakövetkeztethetünk-e a képpontokra.

Ha létezik egy olyan inverz g transzformáció, amelyre mindig igaz, hogy:

$$\underline{r} = g(\underline{r}') = g(f(\underline{r}))$$

akkor kölcsönösen egyértelmű transzformációról beszélünk. Ekkor $\underline{r}'$-ből $\underline{r}$ visszaállítható. Ha nem létezik megfelelő g inverz függvény, akkor $\underline{r}'$-ből $\underline{r}$ nem állítható vissza. Néha ez csak annyit jelent, hogy az $\underline{r}'$ pontoknak nem egy, hanem két (vagy néhány) $\underline{r}$ megfelelőjük van.

! *Ha végtelenül sok $\underline{r}$ tárgypont közös képe egyetlen $\underline{r}'$, és az új $\underline{r}'$ vektorok kevesebb dimenziója térben helyezkednek el* (háromdimenziós tér helyett például egy kétdimenziós felületen), *akkor dimenziócsökkenéssel járó, elfajuló leképezésről beszélünk.* Ilyen például a következő transzformáció:

$$x' = x$$
$$y' = y$$
$$z' = 0$$

mely a 3D-s teret levetíti az x, y síkra.

Koordinátatranszformációt – nézőpontváltoztatást – végezve a transzformáció kölcsönösen egyértelmű, hiszen az új koordináták is maradéktalanul őrzik az eredeti információtartalmat. A ponttranszformációknál viszont a tárgy és a kép pontjainak kapcsolata nem mindig kölcsönösen egyértelmű. Gondoljunk például a 3D-s modelltér leképezésére a 2D-s nézetre. Ez tipikus esete az elfajuló transzformációnak.

2.2.3.1. Koordinátatranszformációk

Ha a 3D-s euklideszi térben egy K koordináta-rendszerről egy K' koordináta-rendszerre térünk át és feltételezzük, hogy a koordinátatranszformáció az alábbi esetek valamilyen kombinációjából áll

- a koordináta-rendszer eltolása,
- a koordináta-rendszer elforgatása,
- a koordinátatengelyek felcserélése,
- a koordinátatengelyek tükrözése,
- a koordinátatengelyeken léptékváltás (ez felöleli a nagyítást, kicsinyítést, összenyomást, széthúzást)

akkor ezt matematikai formában a következő módon fejezhetjük ki

$$\underline{r} = \overline{\overline{T}} \cdot \underline{r}' + \underline{b} \qquad (2.1)$$

vagy komponensekben

$$\begin{aligned} x &= x' \cdot T_{11} + y' \cdot T_{12} + z' \cdot T_{13} + b_x \\ y &= x' \cdot T_{21} + y' \cdot T_{22} + z' \cdot T_{23} + b_y \\ z &= x' \cdot T_{31} + y' \cdot T_{32} + z' \cdot T_{33} + b_z \end{aligned}$$

Itt $\underline{r}$, $\underline{r}'$ egy P pont helyvektorát jelöli a K illetve K' koordinátarendszerben, $\underline{b}$ a K' rendszer origójához vezető helyvektor a K koordinátarendszerben. A

$$\overline{\overline{T}} = \begin{pmatrix} T_{11}, T_{12}, T_{13} \\ T_{21}, T_{22}, T_{23} \\ T_{31}, T_{32}, T_{33} \end{pmatrix}$$

mátrix pedig a transzformáció mátrixa.

Megjegyezzük, hogy a $\overline{\overline{T}}$ mátrix oszlopaiban az $\underline{i}'$, $\underline{j}'$, $\underline{k}'$ új alapvektorok K-beli koordinátáit találhatjuk meg.

A számítógépes grafikához szükséges koordinátatranszformációk jellegüknek megfelelően nem elfajulók (mert a testek megőrzik alakjukat), így a $\overline{\overline{T}}$ mátrixnak létezik $\overline{\overline{T}}^{-1}$ inverze (azaz a koordinátatranszformáció minden további nélkül megfordítható), melynek segítségével a K' koordináta-rendszerben is kifejezhetjük a régi és új koordináták közötti összefüggést.

Mivel a (2.1) összefüggés alapján

$$\overline{\overline{T}}^{-1}\ \underline{r} = \underline{r}' + \overline{\overline{T}}^{-1}\ b$$

$$\underline{r}' = \overline{\overline{T}}^{-1}\ \underline{r} + \underline{b}'$$

adódik, ahol $\underline{b}' = \overline{\overline{T}}^{-1} \cdot \underline{b}$ a régi origónak a helyvektorát jelenti a K' rendszerben. Komponensekben:

$$x' = x \cdot T_{11}^{-1} + y \cdot T_{12}^{-1} + z \cdot T_{13}^{-1} + b_x'$$

$$y' = x \cdot T_{21}^{-1} + y \cdot T_{22}^{-1} + z \cdot T_{23}^{-1} + b_y'$$

$$z' = x \cdot T_{31}^{-1} + y \cdot T_{32}^{-1} + z \cdot T_{33}^{-1} + b_z'$$

Fontos szerephez jut a számítógépes grafikában a következő két speciális eset:

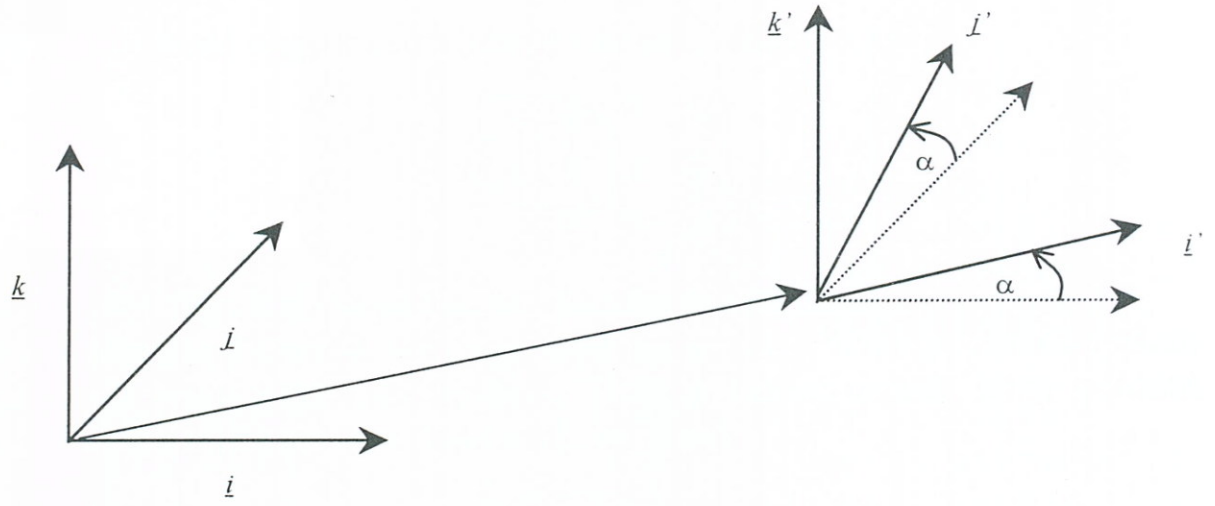
Az origó eltolása

Ekkor $\overline{\overline{T}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ egységmátrix és $\underline{b} \neq \underline{o}$

$$\underline{r}' = \underline{r} + \underline{b}' = \underline{r} - \underline{b}$$

A koordináta-rendszer mozgatása

A koordináta-rendszer mozgatása esetén a koordinátatengelyek által bezárt szög és a koordinátatengelyek beosztása (skálája) változatlan. Minden mozgatás visszavezethető egy eltolás és egy forgatás együttes alkalmazására (lásd 30. sz. ábra).



30. sz. ábra
A koordináta-rendszer mozgatása

Matematikailag bebizonyítható, hogy ebben az esetben a $\overline{\overline{T}}$ transzformációs mátrix azzal a speciális tulajdonsággal bír, hogy transzponáltja (azaz a mátrix főátlójára való tükrözéssel létrejött új mátrix) megegyezik $\overline{\overline{T}}$ inverzével.

Azaz

$$\overline{\overline{T}}^{*} = \overline{\overline{T}}^{-1}$$ (izometrikus mátrix)

A koordináta-rendszer mozgatása azért kiemelt jelentőségű a számítógépes grafikában, mert ezzel tudjuk modellezni azt az igényt, hogy egy 3D-s tárgyat kamerával „körbejárunk" és több oldalról, esetleg különböző távolságokból megvizsgáljuk a képét.

2.2.3.2. Ponttranszformációk

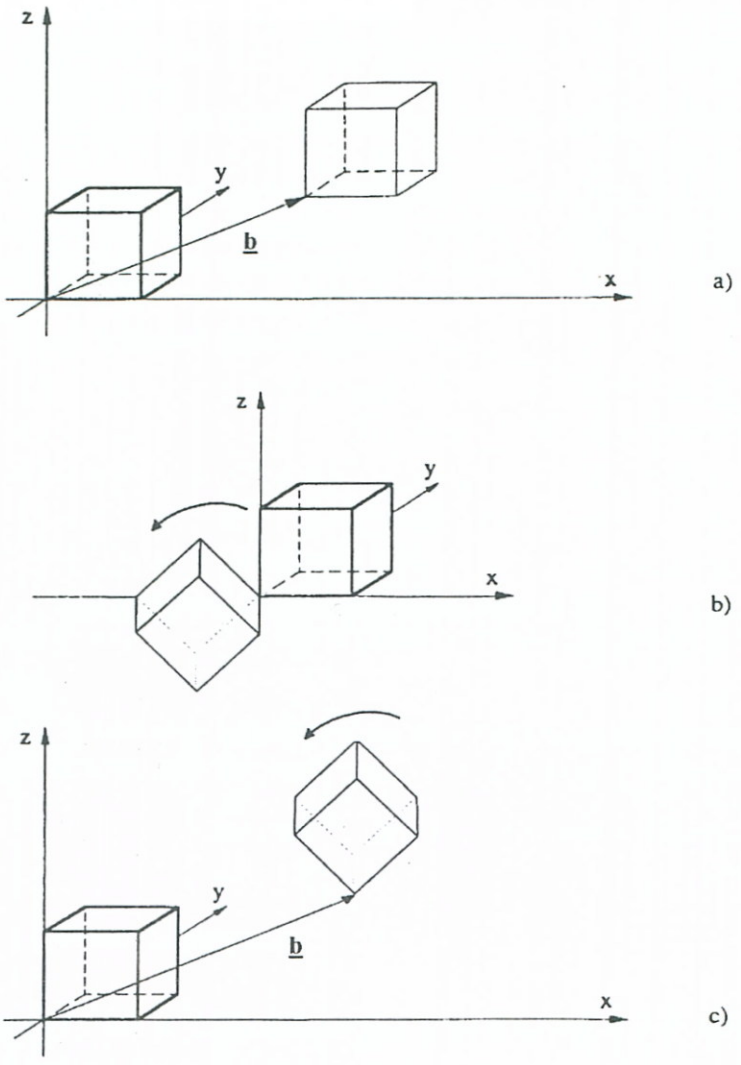
Ebben az esetben a koordináta-rendszerben ábrázolt test minden egyes tárgypontjához, hozzárendeljük egy másik test pontjait.

Ezt a leggyakrabban a test eltolásával, forgatásával és mozgatásával hajtjuk végre a számítógépes grafikában. Ezeket mutatja a 31. sz. ábra.

Matematikailag a ponttranszformációkat szintén az

$$\underline{r} = \overline{\overline{M}} \cdot \underline{r}' + \underline{b}$$

transzformációs egyenlettel tudjuk kifejezni.



31. sz. ábra
Ponttranszformációk: a) eltolás; b) forgatás; c) mozgatás

A ponttranszformációk közül leggyakrabban a számítógépes grafikában a következőket használjuk:

! *Eltolás:* $\overline{\overline{M}}$ egységmátrix, $\underline{b} \neq \underline{0}$

Forgatás: $\overline{\overline{M}}$ izometrikus mátrix, $\underline{b} = \underline{0}$

Mozgás: $\overline{\overline{M}}$ izometrikus mátrix, $\underline{b} \neq \underline{0}$

Nagyítás, kicsinyítés, összenyomás, széthúzás (léptékváltás vagy skálázás): ekkor $\overline{\overline{M}}$ diagonális mátrix (azaz a főátlón kívüli összes elem 0) és $\underline{b} = \underline{0}$

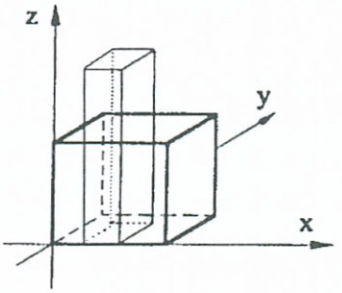
Utóbbi esetben a képpontok halmaza már nem egybevágó a tárgypontok halmazával, a test eltorzul.

Ha nagyításról vagy kicsinyítésről van szó, akkor

$$\overline{\overline{M}} = \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & A \end{pmatrix}$$

alakú. Ha $A>1$ nagyítás, ha pedig $A<1$ kicsinyítés történik.

Ha a főátlóbeli elemek nem egyeznek meg, akkor a tengelyek irányában összenyomás vagy széthúzás történik. Tipikus hatása ennek, hogy például egy kocka képe téglatest lesz (lásd a 32. sz. ábra).



32. sz. ábra
Kocka széthúzása, összenyomása téglatestté

Az előbbiekben megismert transzformációkat a következőképpen is csoportosíthatjuk:

- *Egybevágósági transzformációk:*
  - *eltolás,*
  - *forgatás,*
  - *tükrözés.*

  Ekkor a test képével egybevágó (méretazonos).
- *Hasonlósági transzformációk:*
  - *kicsinyítés,*
  - *nagyítás.*

  Ekkor a test képe az eredetihez képest méretarányosan változik meg. Nem változik meg viszont a test alakja és szögei.
- *Általános léptékváltás:*
  - *összenyomás,*
  - *széthúzás.*

  Ekkor a test képe különböző irányokban eltérő módon torzul.

Ezeket a transzformációkat összefoglaló néven affin transzformációknak nevezzük. Jellemzőjük többek között például a következő:

- pont képe pont, egyenes képe egyenes, sík képe sík,

- metsző térelemek képeinek metszéspontjai megegyeznek az eredeti metszésvonal képével.

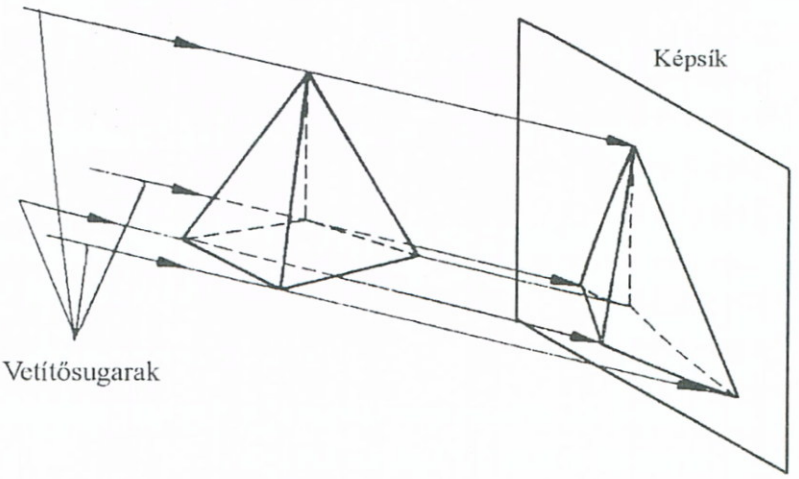
(Az affin transzformációk közé tartoznak az ún. „nyírások" is, erre azonban a számítógépes grafikában általában nincs szükség.)

2.2.3.3. Vetítések

Ha raszteres képpé konvertálva a 3D-s tárgyakat meg akarjuk jeleníteni a monitor képernyőjén, akkor ezeket a 3D-s modelltérből egy 2D-s nézetre kell leképezni. Ezért a számítógépes grafikában kiemelt jelentőségű transzformációk a vetítések.

! *Vetítésnek nevezzük azokat a dimenzióveszteséggel járó ponttranszformációkat, melyeknél a képpont és a neki megfelelő tárgypont egy egyenesen helyezkedik el.* (A fény egyenesvonalú terjedése folytán az optikai leképezések általában ilyen transzformációval egyenértékűek, innen származik a vetítés elnevezés is.)

! *A tárgy- és képpontokon áthaladó egyenest vetítősugárnak nevezzük. A vetítés eredménye a vetület, ami egy térbeli síkon – a képsíkon – képződik. Az egyes tárgypontok képe a vetítősugarak döféspontja a képsíkkal* (lásd a 33. sz. ábrát).



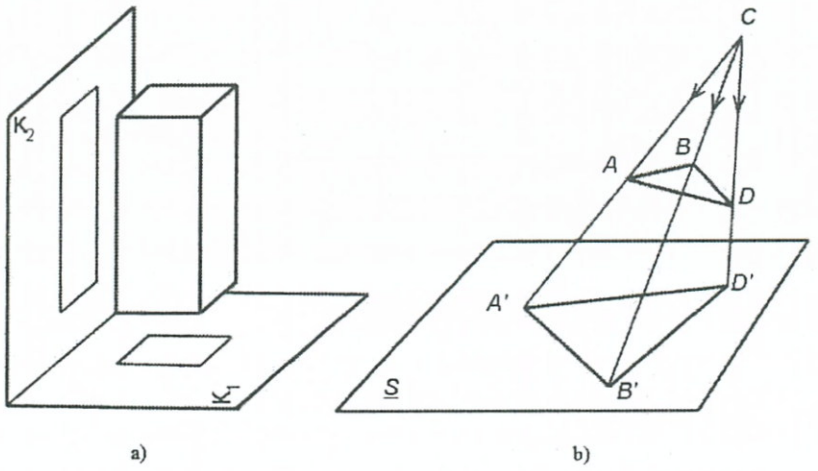
33. sz. ábra
Tárgy képének előállítása a vetítősugarakkal a képsíkon

A vetítés két alaptípusa a párhuzamos és a középpontos vetítés.

! *Párhuzamos vetítésről beszélünk, ha a vetítősugarak egymással párhuzamosak.* Ha ezen kívül a vetítősugarak még merőlegesek is a képsíkra, akkor merőleges a vetítés, egyébként pedig a ferde vetítés elnevezést használjuk (lásd a 34. sz. ábra a) részét).

(A műszaki rajznál a merőleges vetítés módszerét alkalmazzuk; ezért a műszaki rajz nézetei a tárgy ilyen vetületei.)

A középpontos vagy perspektivikus vetítés esetén a vetítősugarak mindegyike áthalad egy vetítési középponton, a centrumponton (lásd a 34. sz. ábra b) részét). A létrejövő kép igen közel áll az emberi szem, illetve a fényképezőgép által alkototthoz. A perspektivikus hatás elsősorban a tárgy és a centrumpont távolságától függ. Ha ez a távolság minden határon túl nő, a középpontos vetítés párhuzamos vetítésbe megy át.



34. sz. ábra
a) párhuzamos vetítés (merőleges); b) középpontos vetítés

A középpontos vetítés általunk értelmezett fogalma, tulajdonképpen két transzformáció együttese. Az első egy centrális projekció, mellyel a teljes teret a teljes térre képezzük le. A második lépésként a projektív transzformáció képpontjait merőlegesen vetítjük a képsíkra.

2.2.4. A térbeli transzformációk számítógépes algoritmusainak egyszerűsítése homogén koordináták bevezetésével

E fejezetben a célunk az lesz, hogy a különböző térbeli transzformációkhoz olyan matematikai formalizmust találjunk, melyet egyszerűen és hatékonyan lehet algoritmizálni a számítógépes rendszerekben.

Az affin transzformációk és a vetületképzés műveleteinek egységes, közös leírásához a homogén koordináták bevezetésével jutunk el.

Egészítsük ki háromdimenziós helyvektorainkat egy formális lépéssel négydimenzióssá. A tér egy P pontjához vezető

$$\underline{r} = (x, y, z)$$

vektort ezentúl írjuk így:

$$(w \cdot x,\ w \cdot y,\ w \cdot z,\ w)$$

ahol w egy tetszőleges, nemzérus skalár. Ebből az írásmódból bármikor visszatérhetünk a valóságos háromdimenziós koordinátákra úgy, hogy a vektor első három koordinátáját elosztjuk a negyedikkel.

A P $(x,\ y,\ z)$ pont homogén koordinátáinak nevezzük a $(w \cdot x,\ w \cdot y,\ w \cdot z,\ w)$ koordináta négyest.

Világos, hogy a P pontnak e definíció szerint végtelen sok homogén koordinátája létezik, aszerint, hogy w-nek milyen konkrét értéket választunk.

Ezt a többértelműséget elkerülhetjük, ha w értékét 1-nek választjuk. *Ekkor az*

$$(x, y, z, 1)$$

koordinátanégyest a P pont normalizált homogén koordinátáinak nevezzük.

A normalizált homogén és a szokásos 3D-s térkoordináták között a fenti megfeleltetés kölcsönösen egyértelmű:

- *ha P pont koordinátája 3D-ben (x, y, z), akkor P homogén koordinátája (x, y, z, 1),*
- *ha P homogén koordinátája (x, y, z, 1), akkor P 3D-s koordinátája (x, y, z).*

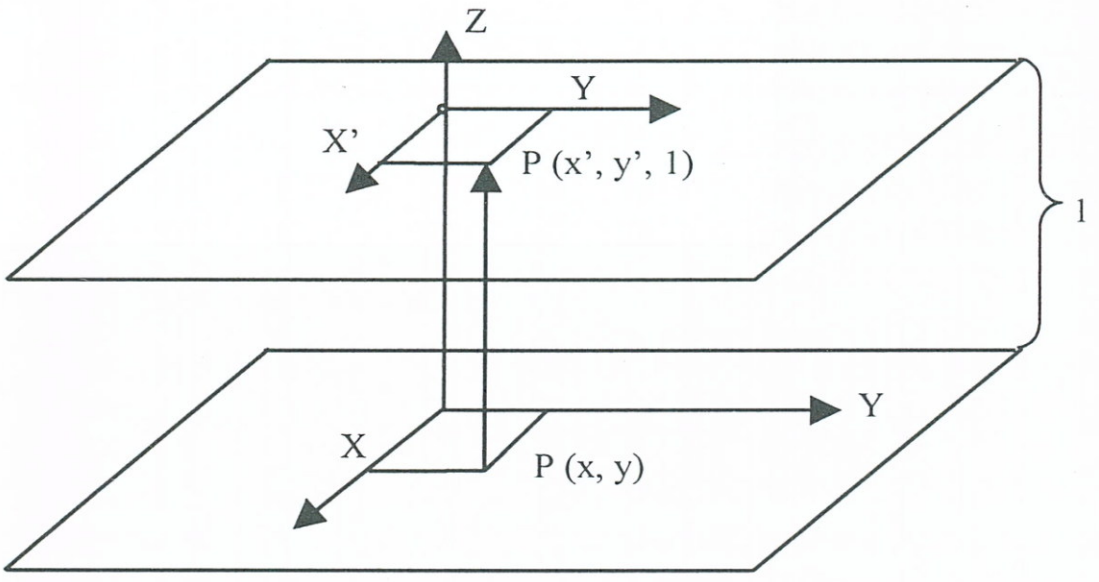
A 3D-s tér homogén koordinátáit szemléletesen nem lehet bemutatni. Ez viszont könnyen elvégezhető a 2D-s homogén koordinátákkal, csak ki kell lépnünk a 3D-s térbe. Az (x, y) alapsíkkal vegyünk fel egy párhuzamos síkot a térbeli koordináta-rendszerben, mely a z tengelyt az 1 pontban metszi.

Ekkor egy P (x, y) síkbeli pont normalizált homogén koordinátáit úgy kapjuk meg, hogy a P pontot merőlegesen „felvetítjük" a $z = 1$ síkra (lásd 35. sz. ábra).

A homogén koordináták bevezetése nem csak formális konstrukció. Ezt matematikailag az euklidészi tér úgynevezett projektív lezárása teszi szükségessé, ami alatt azt értjük, hogy a teret kiegészítjük „ideális" térelemekkel:

- A tér minden egyes egyenesét a közönséges pontokon túlmenően egy ideális ponttal egészítjük ki. Definíció szerint párhuzamos egyenesekhez egy ideális pont tartozik, melyben ezek „metszik" egymást.
- Egy sík ideális pontjai egy ideális egyenesen vannak. Párhuzamos síkok ideális egyenesei megegyeznek.
- A tér ideális egyenesei egy síkban vannak, a tér ideális síkjában.

(Az ideális térelemeket „végtelen" távoli térelemeknek is szokták nevezni.)



35. sz. ábra
Kétdimenziós normalizált, homogén koordináták

Az ideális térelemekhez homogén koordináta-rendszerben konkrét koordinátákat rendelhetünk hozzá. Így például (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0) az *x*, *y*, *z* tengely irányának megfelelő ideális pontok homogén koordinátái.

A számítógépes grafikában az ideális térelemek bevezetése annyiban érdekes, hogy a párhuzamos és nem párhuzamos térelemek kezelése ezzel a formalizmussal egységes módon megoldható, ami lényegesen leegyszerűsíti a transzformációk programalgoritmusait.

Emlékeztetünk arra, hogy a *számítógépes grafikában szükséges affin transzformációk legáltalánosabb formája*:

!

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}$$

Ez teljesen egyenértékű a normalizált homogén koordinátákkal felírt következő összefüggéssel:

!

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} T_{11} & T_{12} & T_{13} & b_x \\ T_{21} & T_{22} & T_{23} & b_y \\ T_{31} & T_{32} & T_{33} & b_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

Felhívjuk a figyelmet arra, hogy az affin transzformációk homogén koordinátákban adott mátrixának utolsó sora mindig (0 0 0 1).

Tehát az eltolást, forgatást, léptékváltást kifejező transzformációs egyenletet homogén koordinátákban egyetlen mátrixszorzással le tudjuk írni. Ez alapján több egymás utáni transzformáció – a transzformációk mátrixainak összeszorzásával – szintén egyetlen mátrixszorzással jellemezhető.

Néhány fontosabb transzformáció mátrixa

Eltolás a $\begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}$ vektorral $\begin{pmatrix} 1 & 0 & 0 & b_x \\ 0 & 1 & 0 & b_y \\ 0 & 0 & 1 & b_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Forgatás α szöggel az x tengely körül $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Léptékváltás (skálázás)
x tengely egysége e_x lesz
y tengely egysége e_y lesz
z tengely egysége e_z lesz
$\begin{pmatrix} e_x & 0 & 0 & 0 \\ 0 & e_y & 0 & 0 \\ 0 & 0 & e_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

A tükrözés a léptékváltás speciális esete. Így például a következő mátrix a teret

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

tükrözi az yz síkra, azaz e transzformáció az x tengely irányát „megfordítja”.

A koordinátatranszformációkat is minden nehézség nélkül jellemezhetjük 4x4-es homogén koordinátás mátrixokkal. Így például, ha az új koordináta-rendszer origója egybeesik a régi koordináta-rendszer origójával, és az új koordináta-rendszer alapvektorait $\underline{u}$, $\underline{v}$, $\underline{w}$ jelöli, akkor a transzformációs mátrix

$$\begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

lesz.

A homogén koordináták bevezetésének további előnye, hogy a perspektív transzformációk, azaz a középpontos vetítések is kifejezhetők mátrixszorzással.

Például egy olyan középpontos vetítés mátrixa, melynek a képsíkja az (x, y) koordinátasík a következő:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/N & 0 \end{pmatrix}$$

Itt N a vetítési középpont távolsága az origótól a z tengelyen.

A 4x4-es homogén koordinátás mátrixokkal megvalósított transzformációkat projektív transzformációknak nevezzük. Ezek egy részcsoportját képezik az affin transzformációk, amikor a mátrix 4-ik sora (0, 0, 0, 1). A „tiszta" projektív transzformációkat az jellemzi, hogy utolsó soruk (p, q, r, 1) formátumú, ahol p, q, r közül legalább egyik $\neq 0$. Ennek egyik speciális esete az

$$\overline{\overline{P}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/N & 1 \end{pmatrix}$$

mátrix, mely a 3D-s teret „z irányú" projekcióval képezi le a 3D-s térre. A centrális vetítés előzőekben megmutatott mátrixát ebből úgy kapjuk meg, hogy $\overline{\overline{P}}$ képpontjaira az (x, y) síkra vonatkozó merőleges vetítést alkalmazzuk, azaz a $z' = 0$ transzformációt.

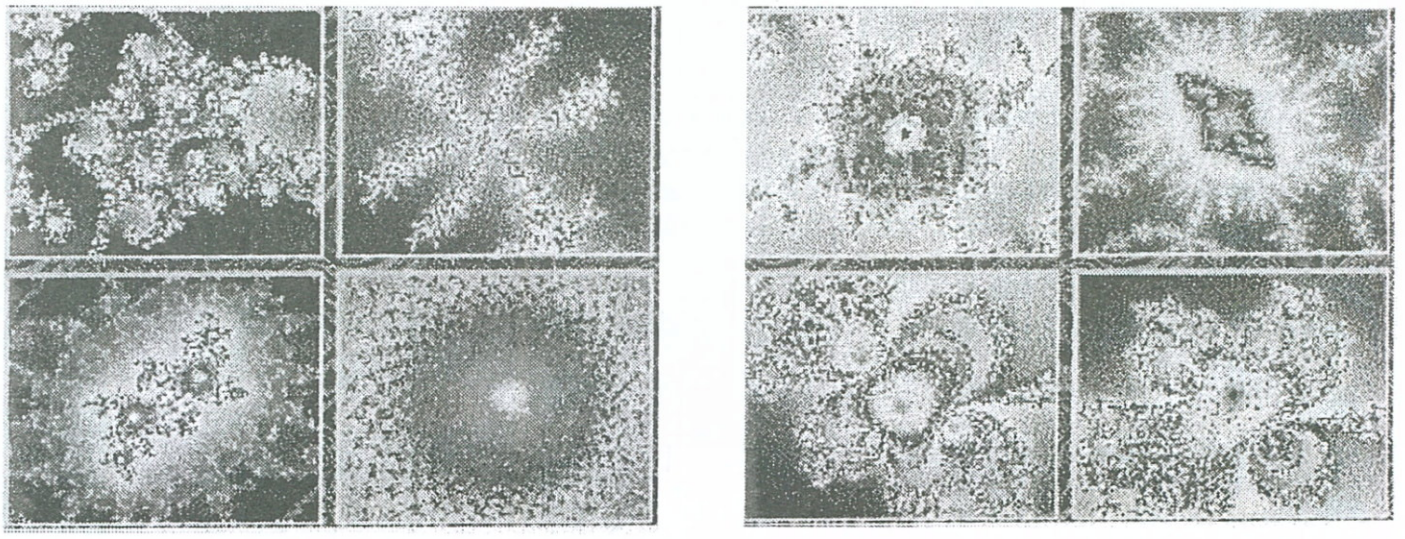
Ennek a mátrixformalizmusnak az is az előnye, hogy a korábbi „normális" transzformációkat és a középpontos vetítést egy mátrixszorzással összevonhatjuk. Így például egy 3D-s objektum elforgatására vonatkozó felhasználói utasítás végrehajtása során az elforgatást, majd az elforgatott test megjelenítéséhez szükséges perspektív transzformációt egy mátrixszorzással tudjuk modellezni.

Most már nyilvánvalóvá vált a homogén koordináták használatának előnye. Azáltal, hogy minden, a számítógépes grafikában lényeges geometriai transzformáció mátrixszorzással modellezhető, egyszerűbbé válik a transzformációkat megvalósító szoftver és a transzformációkra célhardver kifejlesztése is könnyebb lesz.

2.3. FRAKTÁLGEOMETRIA ALKALMAZÁSA A SZÁMÍTÓGÉPES GRAFIKÁBAN

2.3.1. A fraktál és fraktálgeometria fogalma

A számítógépes grafikában több területen is fontos szerephez jutnak a fraktálok. Ezekre mutat példát a 36. sz. ábra.



36. sz. ábra
Példák fraktálokra

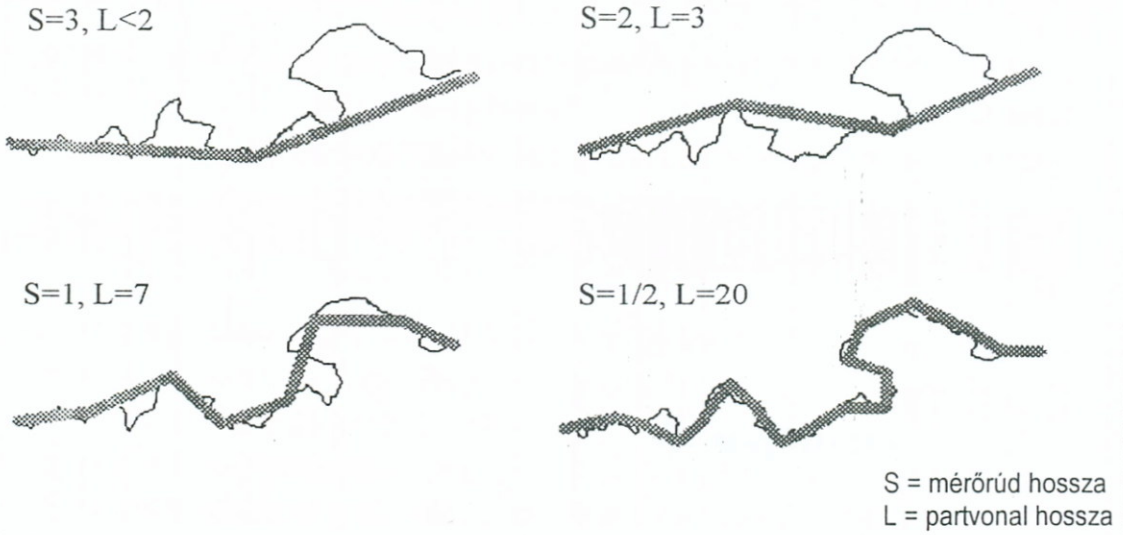
A fraktál fogalmát Benoit B. Mandelbrot vezette be a matematikába 1975-ben, a szó a latin „fractus" (törött) szóból származik.

A definíció előtt kövessük egyszerűsítve Mandelbrot gondolatmenetét.

Milyen hosszú Nagy-Britannia partvonala? Első pillantásra ennek a kérdésnek a megválaszolása egyszerűnek tűnik, egy térkép és egy vonalzó segítségével bárki rövid időn belül meg tudja mondani az eredményt. A gond csak az, hogy egy nagyobb léptékű térképpel megismételve a mérést, az előzőnél nagyobb eredményt kapnánk. Ha lemennénk a partra és ott végeznénk el a mérést, akkor egy még nagyobb érték adódna. Kiderül, hogy minél finomabb skálán végezzük a mérést, annál nagyobb eredményt kapunk, és ennek a növekedésnek nem lesz határa. Így, ha az (elméleti) mérésnél a felbontás végtelen kicsi lenne, akkor a becsült hossz végtelen naggyá válna.

Ha a mérésekről beszélünk, akkor a felbontás jellemezhető meghatározott hosszúságú mérőrudakkal: minél finomabb a felbontás, annál rövidebb a rúd. Így a görbét úgy is tekinthetjük, mintha egymáshoz kapcsolódó, azonos hosszúságú rudakból állna (37. sz. ábra). Világos, hogy az így kapott képen nem látszanak azok a részletek, amelyek kisebbek, mint a rudak.

A különböző felbontások által okozott problémák vizsgálata vezette el Mandelbrotot a fraktálok fogalmához. Észrevette, hogy a partvonal különböző felbontásokban való képeiben van egy közös lényeg: ezek többé-kevésbé „hasonlítanak" egymásra (érdesek, töröttek).



37. sz. ábra
Partvonal mérése különböző hosszúságú mérőrudakkal

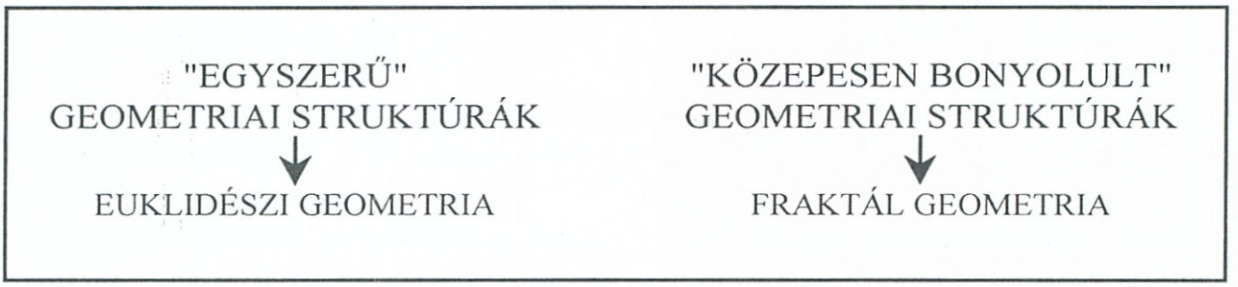
A fraktál egy olyan geometriai alakzat, melynek részei hasonlítanak az egész alakzathoz és a részek a még kisebb részekhez. Egy geometriai görbe vagy felszín tehát akkor fraktál, ha bármely részét felnagyítva az eredetivel azonos alapmotívumú, azaz önhasonló görbét vagy felszínt kapunk.

Ez azt is jelenti, hogy a különböző felbontásokhoz tartozó önhasonló részek közötti átmenetet egy meghatározott transzformációval képezhetjük. A legegyszerűbb fraktáloknál ezek a transzformációk az eltolás, forgatás és skálázás.

Talán meglepő, hogy a természetben számtalan olyan dolgot találhatunk, melyek fraktáltulajdonságokat mutatnak. Ilyenek például:

- a fa és ágai,
- a hegycsúcs, a hegy, a hegység,
- egy folyam vízgyűjtő területe (ér, patak, folyó, folyam),
- a turbulens áramlások,
- az érrendszer az élő szervezetekben.

A fraktálok elméletével a fraktálgeometria foglalkozik, mely a geometriai objektumok felbontás-invariáns, azaz önhasonló tulajdonságait tárgyalja. Átmenetet képez a klasszikus geometria egyszerű, ideális alakzatainak (például kiterjedés nélküli kör) szabályossága, és a szabályosságot nem mutató kaotikus jelenségek között (lásd a 38. sz. ábrát).



38. sz. ábra
Az euklidészi és a fraktálgeometria összehasonlítása

2.3.2. Fraktáldimenzió

A fraktáldimenzió segítségével meghatározható, mennyire szabálytalan egy fraktálgörbe. Általában a vonalakat egydimenziósnak, a felületeket kétdimenziósnak, a testeket pedig háromdimenziósnak tartjuk. Azonban egy nagyon szabálytalan görbe ide-oda vándorolhat a felületen, olyannyira, hogy szinte teljesen ki is töltheti azt. Így például egy Brown-mozgást végző részecske pályagörbéje, ha az idő minden határon túl növelhető, átmegy a sík minden egyes pontján. A nagyon tekervényes felület, mint pl. egy fa lombozata, vagy a tüdő belső felülete majdhogynem háromdimenziós lehet. Így a szabálytalanságra úgy tekinthetünk, mint olyan tényezőre, mely a dimenziót növeli: egy szabálytalan görbe dimenziója 1 és 2 között lesz, míg egy szabálytalan felületé 2 és 3 közé esik.

Egy fraktálgörbe dimenziója olyan szám, amely azt jellemzi, hogy a görbe két kiválasztott pontja között hogyan nő a távolság, midőn növeljük a felbontást. Tehát amíg a vonal és a felület dimenziója mindig 1, illetve 2, addig a fraktáldimenzió lehet egy ezek közti érték is.

A D fraktáldimenzió definíciója

$$D = \frac{\log(L2/L1)}{\log(S1/S2)}$$

ahol $L1$ és $L2$ a görbén mért hosszúságok, $S1$ és $S2$ pedig a használt mérték nagysága (azaz a felbontás).

Példa

A 33. sz. ábrán látható partvonalnál S=1, ill. S=1/2 esetben L=7, ill. L=20. Így

$$D = \log(20/7) / \log(1/(1/2)) = 1.51$$

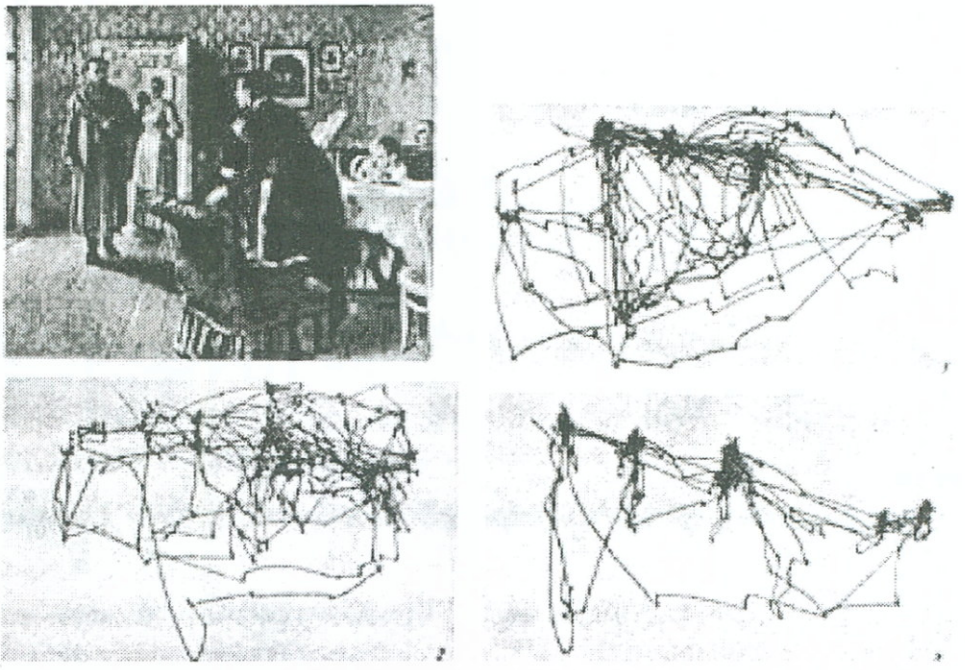
A fraktáldimenziót egy négyzetleszámolási eljárással is meghatározhatjuk, melynek az az előnye, hogy egyszerűen algoritmizálható. Ezért ezzel a módszerrel könnyen készíthetünk olyan programokat, melyek képesek egy raszteres képen ábrázolt fraktál dimenziójának kiszámítására. Ennek lényege a következő:

a) A vizsgált képet négy részre osztjuk.
b) Megvizsgáljuk, hogy a képrészletekben van-e olyan képpont, mely a fraktálhoz tartozik. Azt a számot, hogy hány képpontot tartalmazó képrészletet találtunk, letároljuk.
c) A következő lépésben függőlegesen és vízszintesen megfelezzük a négy képrészletet (összesen így 4x4=16 képrészletet kapunk) és a b) lépéssel folytatjuk.
d) Az előbbi felezési eljárást addig folytatjuk, míg a képrészletek mérete meg nem egyezik egy képpontéval. Ekkor a letárolt számokat átlagoljuk és ez lesz a fraktáldimenzió.

Az algoritmus egy Mandelbrot által javasolt eljáráson alapszik, mely szerint, ha egy fraktált tartalmazó képre egy E oldalhosszú négyzetekből álló hálót borítunk és N darab olyan négyzetet találunk, melyet a fraktál metsz, akkor N $1/E^D$ értékével arányos, ahol D a fraktáldimenzió.

A fraktáldimenzióra a gyakorlatban talán meglepő példákat is találhatunk. Egy pszichológiai kísérlet során a vizsgált személyek *Rjepin „Váratlan látogató"* című képét nézte, és eközben szemmozgásukat regisztrálták. Különböző kérdésekre kellett válaszolnia, ez alatt a szeme az ábrán látható utakat járta be (lásd a 39. sz. ábrát).

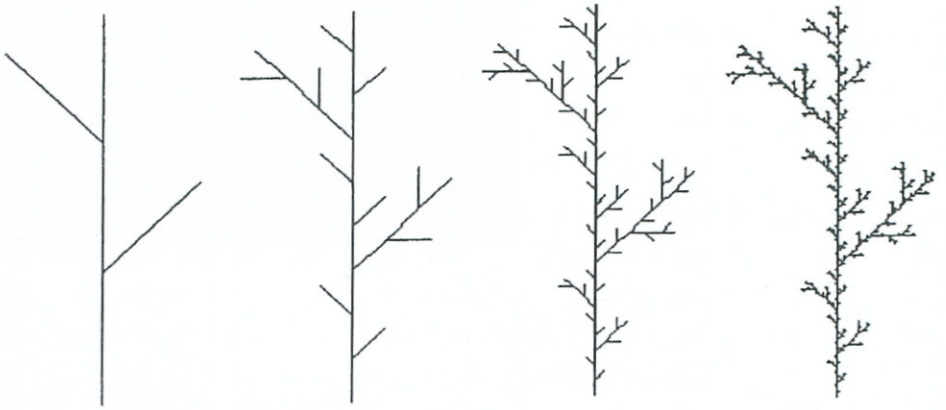
A görbék elemzése megmutatta, hogy valamennyi útvonal fraktáldimenziója azonos = 1,5.



39. sz. ábra
A szemmozgás fraktál tulajdonsága

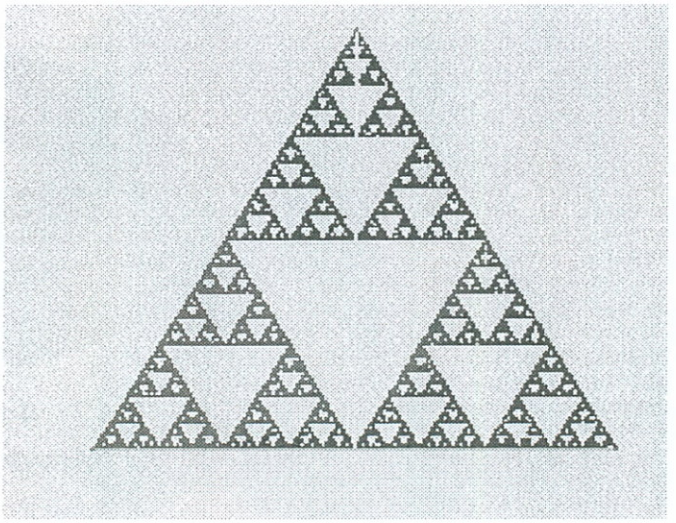
2.3.3. Lineáris, nem lineáris és sztohasztikus fraktálok

A fraktálok úgy generálhatók, hogy az önhasonlóságot jellemző mintázatot ismételjük egyre kisebb és kisebb mérettartományokban (azaz egyre nagyobb és nagyobb felbontásnál). Ezek az úgynevezett klasszikus fraktálok, melyekre egy példát (vonalelemekből felépített fraktál) a 40. sz. ábra mutat.



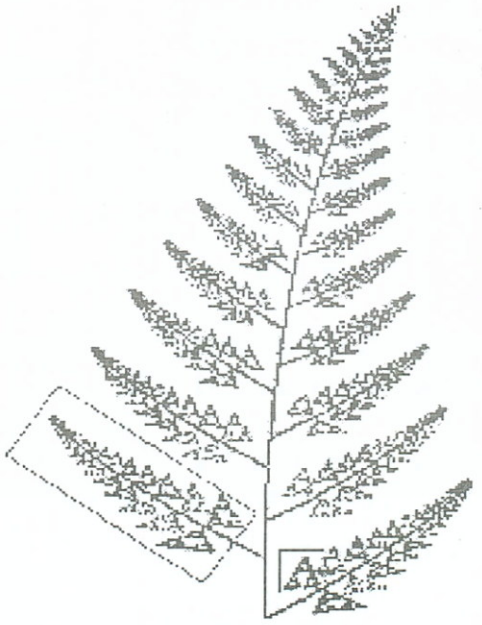
40. sz. ábra
Vonalas fraktál

Ha az egyes felbontások között az átmenetet affin transzformációkkal képezzük, akkor lineáris fraktálokat kapunk. (A nevüket onnan kapták, hogy az iterációnál lineáris transzformációt alkalmazunk.) Erre jó példa a Sierpinski-háromszög (41. sz. ábra).



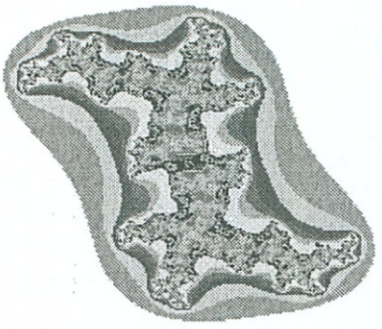
41. sz. ábra
Sierpinski-háromszög

Az előző ábrán látható háromszögekből olyan komplexebb formákat is előállíthatunk, mint a 42. sz. ábrán található Bransley „páfrány".

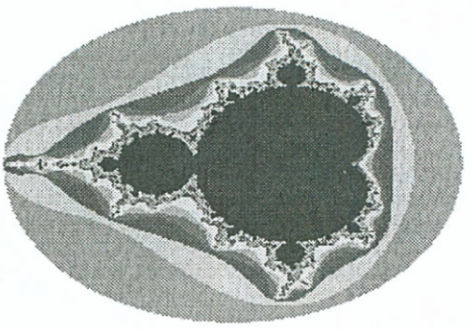


42. sz. ábra
Bransley páfrány

Ha az egyes iterációs lépésekben nem lineáris leképezést alkalmazunk, akkor ennek eredményeként nem lineáris fraktálokat kapunk. Ezek közül talán a leghíresebb a Julia-Fatoue és a Mandelbrot-halmaz (lásd a 43. sz. és 44. sz. ábrákat).



43. sz. ábra
Julia-Fatoue-halmaz



44. sz. ábra
Mandelbrot-halmaz

Legyen $g(z) = z^n + c$ alakú függvény a komplex számsíkon értelmezve. A Julia-Fatoue-halmaz és a Mandelbrot-halmaz a $g(z)$ leképezés ismételt alkalmazásával származtatható. Az ábrák az $n = 3$ esetre és $c = 0.91$ esetre mutatnak példát.

A Mandelbrot-halmaz tökéletesen nem önhasonló, de egyes részei mégis ilyen tulajdonságokat mutatnak. (Érdemes például a halmaz szélén lévő részeket kinagyítva tanulmányozni.)

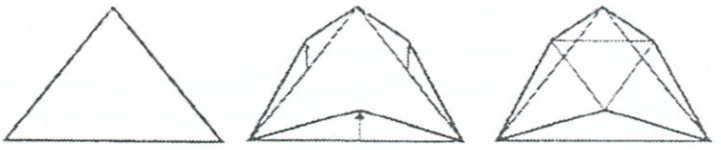
! *Véletlen iterációs algoritmusokkal is generálhatunk fraktálokat. Ezt például megtehetjük úgy, hogy megadunk* $\overline{\overline{A_1}}, \overline{\overline{A_2}}, \ldots \overline{\overline{A_n}}$ *véges számú konkrét affin transzformációt, és ezekhez tartozó* $p_1, p_2 \ldots p_n$ *valószínűségeket, ahol*

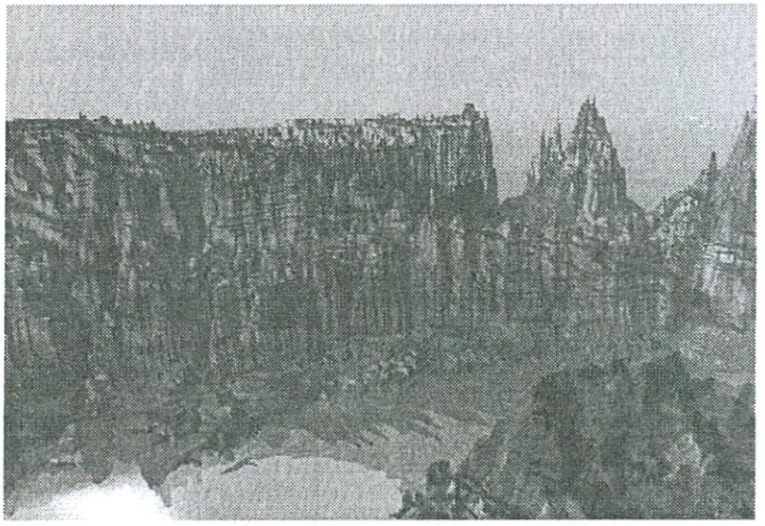
$$p_1 + p_2 + \ldots + p_n = 1$$

Az egyes iterációs lépésekhez $\overline{\overline{A_1}}, \overline{\overline{A_2}}, \ldots \overline{\overline{A_n}}$ *közül választunk ki egy konkrét* $\overline{\overline{A_i}}$ *transzformációt, mégpedig olyan gyakorisággal, hogy elég sok ismétlés esetén ez megfeleljen a* p_i *valószínűség értékének.*

A véletlen fraktáloknak kiemelt jelentősége van a számítógépes grafikában, mert segítségükkel teljesen természetes, valószerű sokaságokat tartalmazó képeket tudunk generálni. Például egy fűszálból felépíthetjük a rét képét, vagy egy fából erdőt generálhatunk.

Egy kopár hegység generálására mutat példát a 45. sz. ábra, melyet síkbeli háromszögek síkjukból való véletlenszerű kimozdításával generáltak.





45. sz. ábra
Tájgenerálás véletlen fraktállal

2.3.4. Fraktálok alkalmazása a számítógépes grafikában

Először a fraktálok néhány általános felhasználási területét mutatjuk be példaszerűen, a teljesség igénye nélkül.

Tudomány

- Kémia
  Nem egyensúlyi rendszerek például elektrokémiai fémleválás
- Fizika
  Nemlineáris dinamika
- Biológia
  Önszervező rendszerek, genetikus kód
- Informatika
  Nyelvtani modellek (Lindenmayer-fraktálok)
- Légkörfizika, időjárás
  Felhők rendszerezése fraktáldimenziójuk szerint
- Orvostudomány
  Hörgők fraktálstruktúrája

Technika

- Nanotechnológia

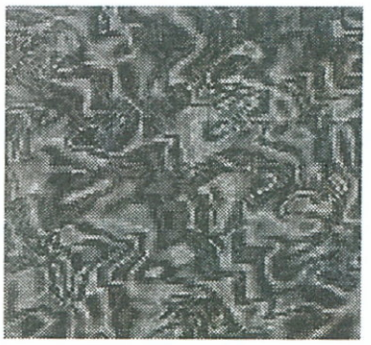
Címezhető tulajdonság felismerés, spektrális jelkészletek összehasonlító elemzése

Művészet

- Fraktálokkal generált művészi(?) képek
- Grafikák fraktáldimenziója
- Fraktálékszerek készítése

A számítógépes grafikában a fraktálokat a következő területeken használjuk:

- Természethű képek generálása véletlen fraktálokkal (rét, erdő, hegységek, felhők stb.).
- Fraktálok felhasználása díszítőelemként. Különösen a WEB lapok hátterének fraktálokkal való felépítése terjed rohamosan (lásd a 46. sz. ábrát).
- A fraktálok igen hatékonyan alkalmazhatók képtömörítésre (például egy véletlen fraktállal generált páfránylevél néhány száz adattal leírható, míg képpontonkénti tárolása több nagyságrenddel nagyobb memóriát igényel).



46. sz. ábra
Fraktál díszítőelemek WEB-laphoz

ELLENŐRZŐ KÉRDÉSEK ?

45. *Mit jelent a grafikus objektumok modellezése?*
46. *Hogyan írjuk a grafikus objektumokat a modellezés során és ez hogyan realizálható a számítógépen?*
47. *Jellemezze a vektorgrafika modellterét!*
48. *Jellemezze a rasztergrafika modellterét!*
49. Miért használunk kivágótéglatestet?
50. *Milyen lépések szükségesek a vektorgrafikus objektumok képének a monitor képernyőjén való megjelenítéséhez?*
51. Milyen előnyök származnak a vektorgrafikus modellezés és a raszteres megjelenítés szétválasztásából?
52. *Mi jellemzi a 2D-s vektorgrafikus modelltér objektumait?*
53. *Mit értünk 2 ½ D modellezés alatt?*
54. Milyen feladatokra alkalmazunk koordináta-rendszereket a számítógépes grafikában?
55. Jellemezze a Descartes koordináta-rendszert!
56. Mi az egyenes és a sík paraméteres egyenlete?
57. Hogyan határozható meg egy egyenes és egy sík döféspontja?
58. *Mi a koordinátatranszformáció és mire használható a számítógépes grafikában?*
59. *Mi a ponttraszformáció és mire használható a számítógépes grafikában?*
60. *Mit értünk elfajuló transzformáció alatt?*
61. Írja fel az általános koordinátatranszformáció vektoregyenletét!
62. Milyen transzformációkra vezethető vissza a koordináta-rendszer mozgatása, és ez mire használható a számítógépes grafikában?
63. *Milyen ponttranszformáció-típusokat használunk a számítógépes grafikában?*
64. *Mit nevezünk affin transzformációnak?*
65. *Mit nevezünk vetítésnek?*
66. *Mit nevezünk párhuzamos és középpontos vetítésnek?*
67. *Mi a P pont normalizált homogén koordinátája, ha 3D-s koordinátái x, y, z?*
68. *Írja fel az affin transzformációk egyenletét homogén koordinátákban!*
69. Adja meg az eltolás mátrixát homogén koordinátákban!
70. Adja meg az x tengely körüli forgatás mátrixát homogén koordinátákban!
71. Adja meg az yz síkra való tükrözés mátrixát homogén koordinátákban!
72. Adja meg a léptékváltás mátrixát homogén koordinátákban!
73. Adja meg a középpontos vetítés mátrixát homogén koordinátákban!

74. *Mit nevezünk fraktálnak?*
75. Mondjon példát olyan természeti jelenségekre, melyek fraktáltulajdonságokat mutatnak!
76. Jellemezze a fraktálgeometria és az euklidészi geometria viszonyát!
77. Mi a fraktáldimenzió?
78. Hogyan generáljuk a lineáris fraktálokat?
79. Mit nevezünk nem lineáris fraktálnak? Mondjon ezekre példákat!
80. *Mit nevezünk véletlen fraktálnak?*
81. Mire használhatók a véletlen fraktálok a számítógépes grafikában?
82. *Jellemezze a fraktálok felhasználási területeit a számítógépes grafikában!*

III.

TÉRBELI GÖRBÉK ÉS FELÜLETEK VEKTOR-GRAFIKUS MODELLEZÉSE

E fejezetben a térbeli görbék és felületek modellezésével kapcsolatos legfontosabb ismereteket foglaljuk össze. Ennek matematikailag precíz és teljes körű kifejtése jelentősen meghaladja a könyv kereteit, ezért csak a legfontosabb fogalmak és eljárások bemutatására törekszünk. (A matematikailag pontos kifejtés a modern matematika olyan szakterületeinek ismeretét is feltételezi, mint a funkcionálanalízis, a projektív és differenciálgeometria, a kombinatorikus topológia. Ezek a diszciplinák jelentősen meghaladják a műszaki egyetemek és főiskolák matematika tananyagát.) Megadjuk az alapvető matematikai összefüggéseket és algoritmusokat, de ezeket – részletezésük helyett – inkább szemléletesen, ábrákkal fogjuk bemutatni. A témakör tárgyalása során a 3D-s térre szorítkozunk, de a megadott algoritmusok, képletek stb. minden nehézség nélkül átvihetők 2D-s vektorgrafikus világkoordináta-rendszerekre is. A könnyebb megértés érdekében a fejezet jelentős mennyiségű ábrát és képet tartalmaz. Ezek közül a matematikai összefüggéseket bemutatókat a szemléltetés érdekében általában 2D-s koordinátarendszerben fogjuk ábrázolni.

3.1. GÖRBÉK ÉS FELÜLETEK MODELLEZÉSÉNEK MATEMATIKAI ÉS GEOMETRIAI ALAPJAI

3.1.1. Görbék leírása

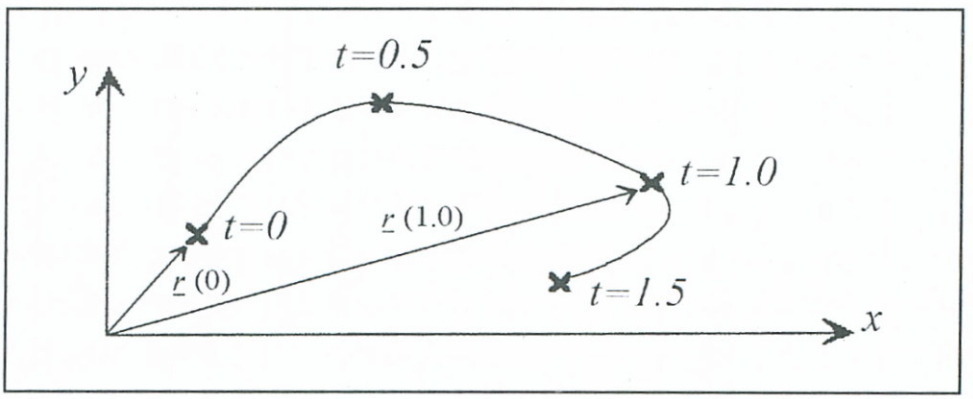
A térbeli görbék modellezésének legfontosabb eszköze a paraméteres vektor egyenlet, mely alkalmazásával a síkbeli és térbeli görbéket azonos módon írhatjuk le.

!

Ez egy $\underline{r} = \underline{r}(t)$, $t \in [t_1, t_2]$ *paraméteres vektor-skalár függvény megadását jelenti, ahol* $[t_1, t_2]$ *a paramétertartomány.* Az egyenlet komponensekben:

$$x = x(t)$$
$$y = y(t) \qquad t_1 \le t \le t_2$$
$$z = z(t)$$

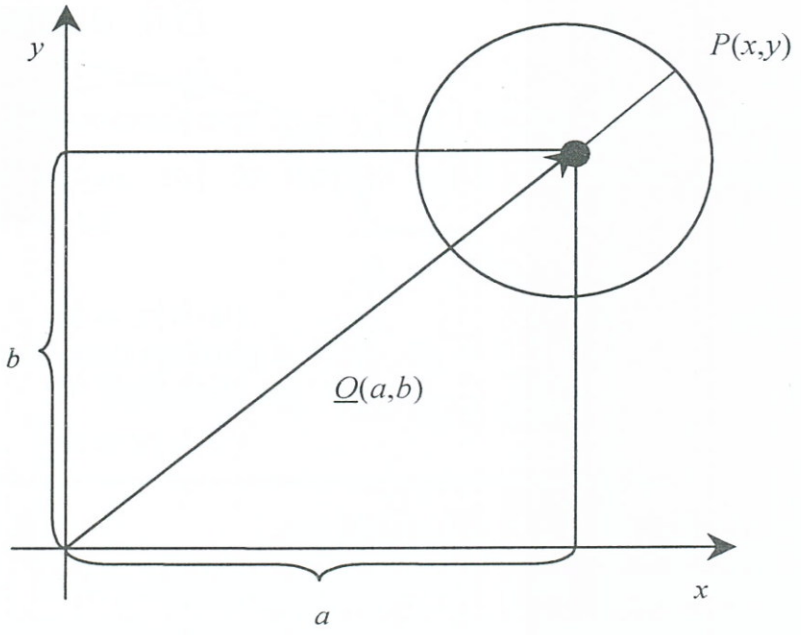
Ezek szerint minden egyes t konkrét számértéket behelyettesítve, az egyenlet egy olyan konkrét $\underline{r}$ vektort állít elő, mely a térgörbe egy pontjára mutat. Ezt szemlélteti 2D-s koordináta-rendszerben a 47. sz. ábra.



47. sz. ábra
Paraméteres vektoregyenlettel adott görbe pontjai

Példa: Egy r sugarú $\underline{Q}$ (a,b) középpontú kör paraméteres egyenlete a síkban

$$x = a + r \cdot \cos t$$
$$y = b + r \cdot \sin t$$



48. sz. ábra
Kör paraméteres egyenlete

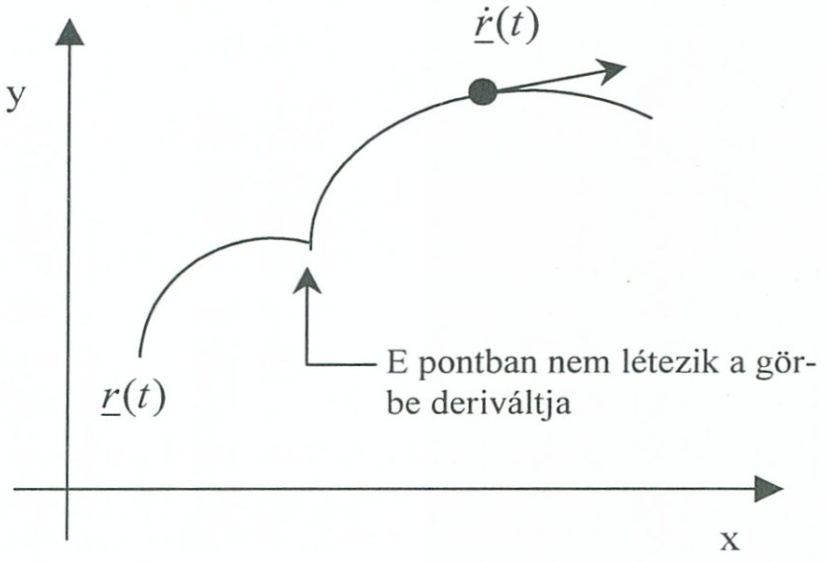
Az $\underline{r}(t)$ függvényről általában feltételezzük, hogy

- kölcsönösen egyértelmű és folytonos leképezés (azaz egy konkrét t_0 értékhez egy és csak egy $\underline{r}_0$ vektort rendelünk hozzá, továbbá a görbe nem szakad)
- t-szerint folytonosan differenciálható és deriváltja nem 0.

Az utóbbi kikötés szemléletesen azt jelenti, hogy a görbén nincsenek „csúcsok", „hegyes" részek és az érintővektort a görbe bármely pontjában meg lehet húzni (lásd 49. sz. ábra). Az érintővektort az

$$\dot{\underline{r}}(t) = \begin{pmatrix} x'(t) \\ y'(t) \\ z'(t) \end{pmatrix}$$

derivált függvényből kaphatjuk meg, ahol $x'(t)$, $y'(t)$, $z'(t)$ a komponensenként képzett skalárfüggvények differenciálhányadosát jelenti.



49. sz. ábra
Egy pontban nem differenciálható görbe
(a többi pontban meghúzható az érintővektor)

Egy görbének nem csak egyfajta paraméteres vektor-skalár egyenlete lehet. Válasszunk például az $\underline{r}(t)$ $t \in (t_1, t_2)$ görbéhez egy olyan $t=t(s)$ $s \in [s_1, s_2]$ függvényt, mely az $[s_1, s_2]$ intervallumot kölcsönösen egyértelmű módon képezi le a $[t_1, t_2]$ intervallumra, akkor $t(s)$-t t helyébe helyettesítve az $\underline{r}(t(s)) \cdot s \in [s_1, s_2]$ egyenlet ugyanazt a görbét definiálja, mint az $\underline{r}(t)$.

A számítógépes grafikában sokszor szükségünk van a lineáris vagy más néven affin paramétertranszformációra. Ennek alkalmazásakor

!
$$t = \frac{t_2 - t_1}{s_2 - s_1}(s - s_1) + t_1 \qquad s \in [s_1, s_2]$$

Ez szemléletesen a $[t_1, t_2]$ paramétertartomány eltolását, nagyítását illetve kicsinyítését jelenti.

A görbéket jellemző geometriai tulajdonságokat (például a görbe érin-
! *tőjének iránya), olyan mennyiségek fejezik ki, melyek a paramétertartomány transzformációja esetén változatlanok maradnak. Ezt úgy is kifejezzük, hogy ezek a tulajdonságok a transzformációval szemben invariánsak.*

3.1.2. Felületek leírása

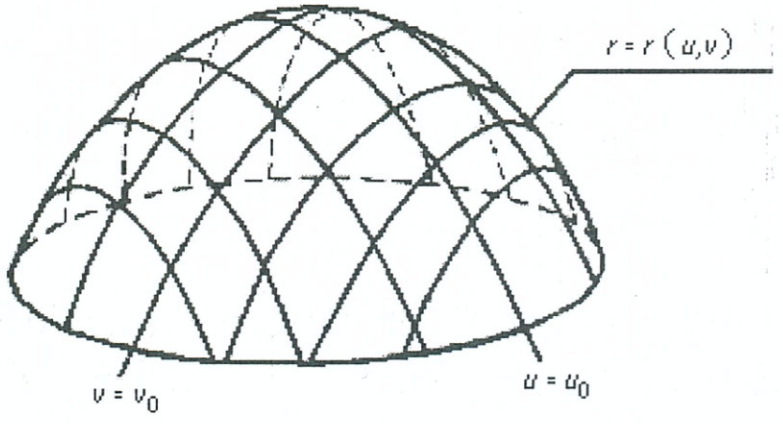
A térbeli felületeket kétparaméteres $\underline{r} = \underline{r}(u,v)$ *vektor-skalár egyenletükkel modellezhetjük, ahol* $u \in [u_1, u_2]$ *és* $v \in [v_1, v_2]$ *a paramétertartomány.* Ez komponensekben felírva az

$$\begin{aligned} x &= x(u,v) \\ y &= y(u,v) \\ z &= z(u,v) \end{aligned} \qquad \begin{aligned} u_1 \le u \le u_2 \\ v_1 \le v \le v_2 \end{aligned}$$

egyenleteket jelenti.

Ezek jelentése – a térgörbékhez hasonlóan – az, hogy minden egyes konkrét u és v paraméterérték (melyet az egyenletbe helyettesítünk) a felület egy pontjához vezető helyvektort határoz meg.

A térbeli felületeket gyakran szintvonalaikkal ábrázoljuk. Ez azt jelenti, hogy megadjuk a felületen azoknak a pontoknak a halmazát, melyek egy konkrét u_0 vagy v_0 értékhez tartoznak. Ez jellemzően egy térgörbe lesz (lásd 50. sz. ábra), amit szintvonalnak nevezünk.



50. sz. ábra
Felület szintvonalakkal ábrázolása

Mint az ábrából is kitűnik, az u_0 és v_0 szerint képzett szintvonalak (meghatározott matematikai feltételek teljesülése esetén) egy és csak egy pontban metszik egymást a felületen. Ha az (u_0, v_0) pontban metsző szintvonalak u, illetve v szerinti parciális differenciálhányadosait képezzük az u_0 és v_0 paramétereknek megfelelő pontban, akkor a szintvonalaknak megfelelő térgörbék érintő vektorait kapjuk.

$$\underline{r}_u = \frac{\partial}{\partial u}\underline{r}(u, v_0) = \begin{pmatrix} \frac{\partial}{\partial u}x(u, v_0) \\ \frac{\partial}{\partial u}y(u, v_0) \\ \frac{\partial}{\partial u}z(u, v_0) \end{pmatrix}$$

!

$$\underline{r}_v = \frac{\partial}{\partial v}\underline{r}(u, v) = \begin{matrix} \frac{\partial}{\partial v}x(u_0, v) \\ \frac{\partial}{\partial v}y(u_0, v) \\ \frac{\partial}{\partial v}z(u_0, v) \end{matrix}$$

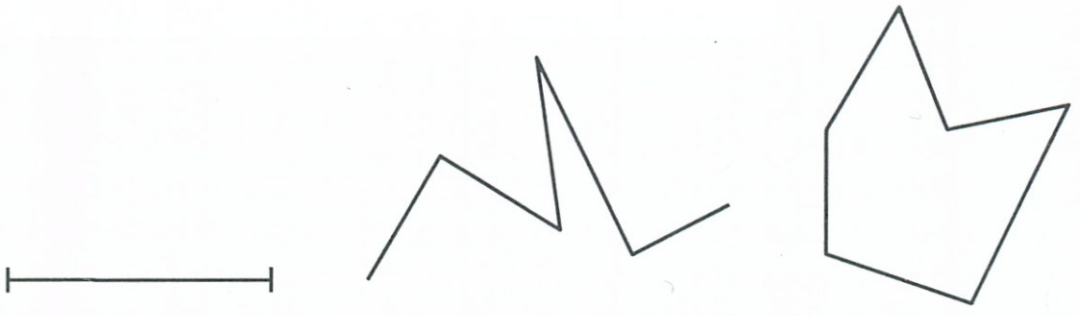
Az $\underline{r}_u$ és $\underline{r}_v$ vektorok együttesen egy síkot határoznak meg, mely a felü-
! letet az (u_0, v_0) pontban érinti. *Egy, az érintő síkra az (u_0, v_0) pontban állított merőleges vektort az $\underline{r}_u$ és $\underline{r}_v$ vektoriális szorzatával állíthatunk elő.*

$$\underline{n} = \underline{r}_u \times \underline{r}_v$$

Ezt az $\underline{n}$ vektort a felület normálvektorának nevezzük. Ez a felület pontjai megvilágításerősségének kiszámításában játszik fontos szerepet.

3.2. GÖRBÉK ÉS FELÜLETEK MEGJELENÍTÉSÉHEZ SZÜKSÉGES POLIGON ÉS POLIÉDER MODELLEK

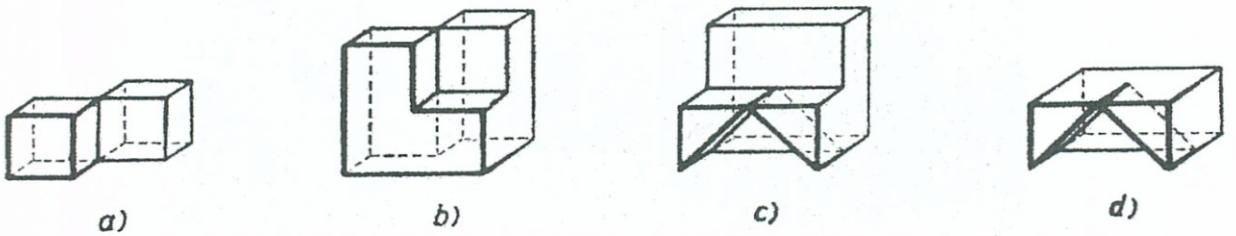
A vektorgrafikus objektumok raszteres képét pl. a képernyőn a vektor-
raszter konverziót végző eszközök (korábban a szoftver, napjainkban a monitorvezérlőkártya) csak egyenes szakaszokkal „összerakott" alakzatokból tudják előállítani. Ezért a görbék és felületek modellezésében fontos szerepet töltenek be az egymáshoz kapcsolódó egyenes szakaszokból felépített poligonok és a sokszöglapokból álló térbeli alakzatok, a poliéderek (lásd 51. és 52. sz. ábra).



szakasz nyílt poligon zárt poligon, vagy sokszög

51. sz. ábra
Poligonok

Sokszöglapokból állónak nevezünk egy térbeli alakzatot, ha olyan sokszöglapokból épül fel, melyeknek a szomszédos lapokkal legalább egy közös élük van. Ha a sokszöglapok egy zárt térrészt határolnak el, akkor ezt poliédernek hívjuk.



52. sz. ábra
Példák poliéderekre

A raszteres kép előállítása során tehát a térbeli görbéket mindig poligonokkal, a felületeket pedig sokszöglapokból – legtöbbször háromszögekből – felépített alakzatokkal közelítjük. Erre a legegyszerűbb eljárás a következő:

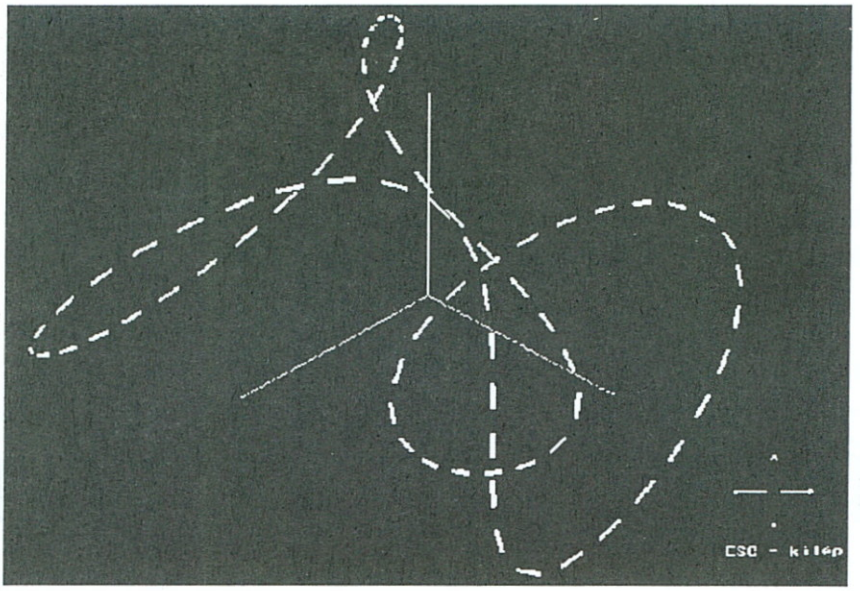
Ha a modellezni kívánt térgörbe vagy felület $\underline{r} = \underline{r}(t)$, illetve $\underline{r} = \underline{r}(u,v)$ vektor-skalár egyenlete adott, akkor a képernyőn való kirajzolását előkészíthetjük a következő eljárással. Osszuk fel a *paramétertartományt* Δt, illetve felületnél Δu, Δv részekre, az osztópontok legyenek

$$t_1, t_2 \ldots t_{i-1}, t_i \ldots t_n \qquad \text{illetve}$$
$$u_1, u_2 \ldots u_{i-1}, u_i \ldots u_n$$
$$v_1, v_2 \ldots v_{i-1}, v_i \ldots v_n$$

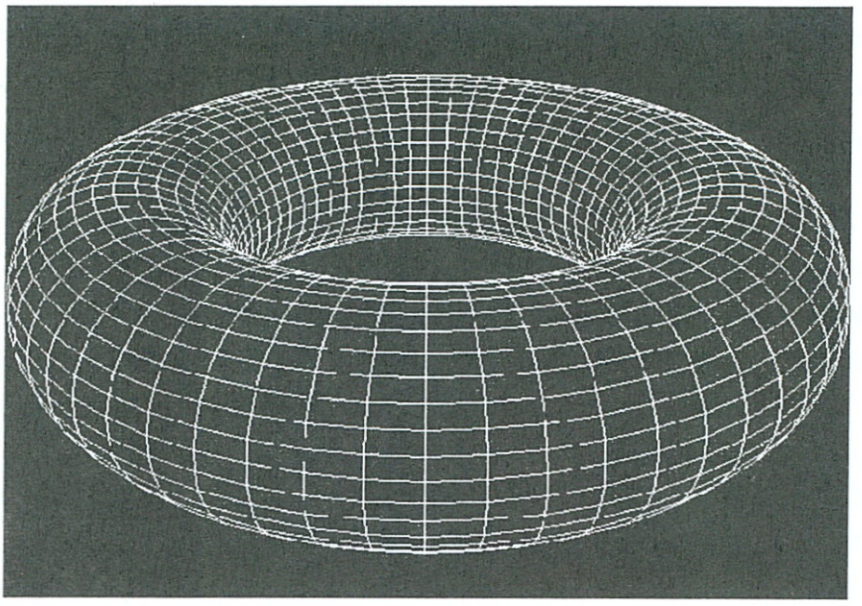
ahol $t_i - t_{i-1} = \Delta t$, $u_i - u_{i-1} = \Delta u$ és $v_i - v_{i-1} = \Delta v$.

Helyettesítsük be a paraméteres egyenletbe az összes t_i, u_i és v_i értéket és határozzuk meg a görbe, illetve a felület pontjaihoz vezető $\underline{r}(t_i)$ és $\underline{r}(u_i,v_i)$ helyvektorokat.

Az egymást követő $\underline{r}(t_i)$ és $\underline{r}(t_{i+1})$ pontok között a térgörbét közelítsük egyenes-szakasszal, a felületet pedig az $\underline{r}(u_i,v_i)$, $\underline{r}(u_i,v_{i+1})$, $\underline{r}(u_{i+1},v_i)$ és $\underline{r}(u_{i+1},v_{i+1})$ csúcspontú négyszöggel. Az így kialakult alakzatot vetítsük a 2D-s ábrázolási térbe, majd raszteresen jelenítsük a képernyőn. Ennek az eljárásnak az eredményét mutatja a 53. sz. és az 54. sz. ábra.



53. sz. ábra
Térgörbe közelítése egyenesszakaszokkal a paramétertartomány felosztásával



54. sz. ábra
Torusz közelítése téglalapokkal a paramétertartomány felosztásával

Nyilvánvaló, a kép minősége attól függ, hogy a paramétertartományban az osztópontok milyen sűrűn helyezkednek el. Így például, ha Δt értéke elég kicsi, a képernyőn kirajzolt görbét folytonos vonalnak fogjuk látni. Felvetődik a kérdés, hogy az előző eljárás mennyiben alkalmazható a vektorgrafikus rendszerekben. Először a programozói ráfordítást vizsgáljuk. A 53. sz. ábrán látható görbét előállító program egy részlete a következő:

```
{$N+}
uses graph, crt, grindhi;
const
      t0=0; t1=2*pi;
      a=70; ds0=20; eps=1;
      kx0=230; ky0=240; tgh=200;
      fel = #72; le = #80; bal = #75; jobb = #77;
      alf=pi/120; bet=pi/60;
var
      ap, bp, cp: real;
      kr: char;
function fx(t: real): real;
begin fx:=3*a*sin(6*t)*cos(3*t) ;
end;
function fy(t: real): real;
begin y:=a*(cos(2*t)-3*sin(t);
end;
function fz(t: real): real;
begin fz:=a*(2*sin(3*t)+cos(t));
end;
procedure nyíl;
const
      xk=580; yk=400;
begin
      line (xk-5,yk,xk-30,yk);
          line (xk-30,yk,xk-26,yk+2);
stb.
```

A görbét generáló teljes program mérete az előzőeknek körülbelül tízszerese.

Ebből is látható, hogy a görbék és felületek paraméteres egyenlettel és a paramétertartomány felosztásával történő modellezése komoly programozói ráfordítást igényel.

Az is kézenfekvő, hogy az osztópontok számának sűrítésével – amelynek a hatására a raszteresen kirajzolt alakzat folytonosabbnak tűnik – jelentősen növekszik az algoritmus műveletigénye.

Az előbbieknél azonban sokkal súlyosabb problémát jelent az, hogy ezzel a módszerrel nem tudjuk kielégíteni azt a vektorgrafikus rendszerekben természetes felhasználói igényt, hogy szükség van:

- adott pontokon keresztülhaladó görbék és felületek modellezésére,
- az alakzatok interaktív parancsokkal történő transzformálására (forgatás, elmozgatás stb.), és
- az alakzatok formájának a felhasználók által történő megváltoztatására (például a görbe egy pontjának megfogása egérrel és „elhúzása").

Mindezeket a görbék és felületek vektor-skalár egyenleteiből kiinduló modellezéssel gyakorlatilag nem tudjuk megoldani. Az előzőekben bemutatott modellezési módszer a vektorgrafikában azért sem adhat professzionális megoldást, mert legtöbbször a görbe és a felület pontos matematikai egyenletét sem ismerjük.

3.3. GÖRBÉK ÉS FELÜLETEK VEKTORGRAFIKUS MODELLEZÉSÉVEL SZEMBEN TÁMASZTOTT KÖVETELMÉNYEK

A számítógépes grafikát igénylő gyakorlati feladatok esetében – mint erre az előzőekben már utaltunk – legtöbbször nem ismerjük a görbék és felületek egyenleteit. Ehelyett például a feladatból fakadóan a következő típusú feltételeket kell kielégítenünk a modellezés során:

- a görbének át kell haladnia előre adott pontokon,
- a felületnek egy adott egyenesszakaszhoz kell csatlakoznia,
- a felületnek egy hagyma alakú kupolára kell hasonlítania stb.

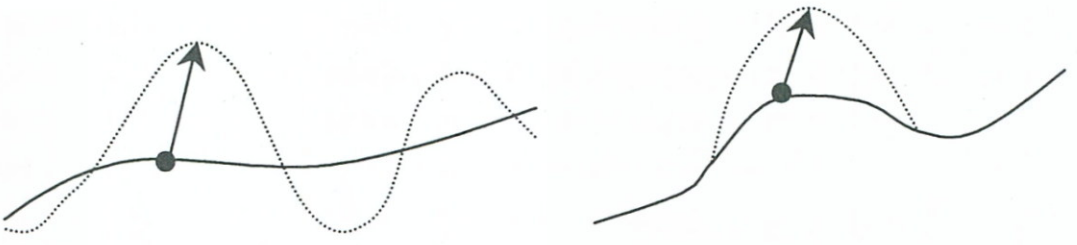
Emellett a generált 3D-s grafikus objektumnak a fenti funkcionális követelmények kielégítése mellett matematikailag egzakt módon nehezen megfogalmazható esztétikai szempontoknak is meg kell felelnie. Ilyenek például:

- a felülettel generált test „arányos" legyen, ne legyen túl magas vagy túl széles,
- a felület szép sima legyen, a nézőnek „áramvonalasnak" tűnjön.

Mindezeket a felhasználó a grafikus rendszerben interaktív módon szeretné érvényesíteni, azaz szüksége van a 3D-s grafikus objektum lépésenkénti, képernyőn történő megjelenítésére és esetleges módosítására.

A számítógépes grafikában ezekre figyelemmel dolgozták ki *a különböző görbe és felületmodellezési algoritmusokat.* Ezeknek tehát *meg kell felelniük a következő gyakorlati követelményeknek:*

- *az algoritmusokat programtechnikailag egységesen, relatíve nem nagy ráfordítással kell megvalósítani. Ezért olyan modellezési eljárások kellenek, melyek kizárják az egyedi, egy-egy speciális függvényhez kötődő megoldásokat. Ugyanakkor a modellezési eljárásnak a legkülönbözőbb formájú, alakú görbéket, felületeket is elő kell tudnia állítani,*
- *a modellezési eljárás biztosítsa az előre megadott pontokon áthaladó térgörbék és felületek hatékony generálását, támogassa a különböző térelemek illeszkedésének (metszéspontok, érintés) kezelését,*
- *a modellezési eljárásban felhasznált függvényeknek viszonylag egyszerűen kiszámíthatóaknak kell lenniük. Az egységes programtechnikai kezelés miatt ezek azonos tulajdonságú függvénycsaládokból kerüljenek ki,*
- *a felhasználóval való hatékony, interaktív kapcsolattartás érdekében*
  - *a modellezési eljárás biztosítsa, hogy a felületek és görbék transzformálása* (például térbeli mozgatás, vagy a paramétertartomány lineáris transzformációi) *relatíve egyszerű algoritmussal, nem túl nagy számításigénnyel megvalósítható legyen.* (Például ne legyen szükség nagy számú görbepont transzformált vektorainak egyenkénti kiszámítására.) *Másként fogalmazva a modellezési eljárásnak invariánsnak kell lennie az affin transzformációkra és a vetítésekre,*
  - *a modell olyan legyen, hogy a képernyőn való megjelenítéshez szükséges poligon és sokszöglapos közelítésre gyorsan konvergáló algoritmusokat lehessen kidolgozni,*
  - *a modell tegye lehetővé a görbék és felületek lokális változtathatóságát, azaz ha például egy pontban kissé megváltoztatjuk a görbét, ez csak a pont közvetlen környezetében okozzon változást* (lásd 55. sz. ábra).



a) b)

55. sz. ábra
Görbék a) globális b) lokális változtathatósága

3.4. GÖRBE ÉS FELÜLET MODELLEZÉSI MÓDSZEREK

A görbék és felületek modellezésénél a felhasználó szemléletes, geometriai adatokat közöl a vektorgrafikus rendszerrel, mely ez alapján generálja a megfelelő görbét vagy felületet. A leggyakrabban olyan térbeli pontokat adunk meg, melyeken az előállítani kívánt görbének vagy felületnek át kell haladnia.

3.4.1. Interpoláció és approximáció

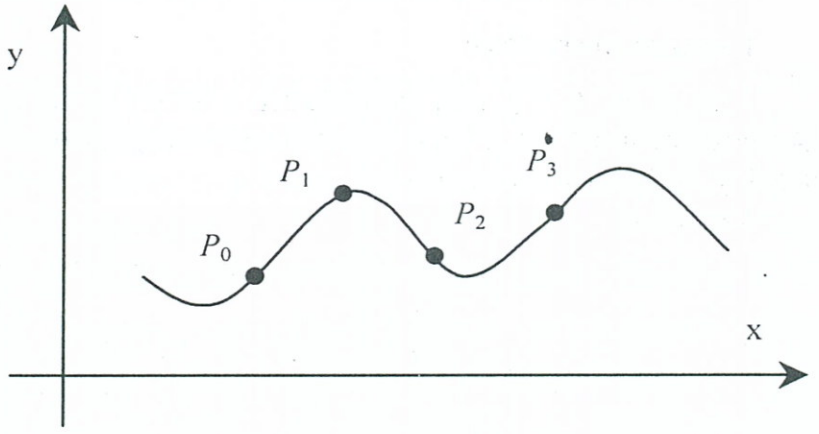
A térbeli görbék és felületek közül azok kiválasztását, melyek a tér előre adott P_1 ... P_n pontjain áthaladnak, egy interpolációs feladat megoldásának nevezzük.

Például síkbeli görbék esetében legyen adott a P_0, P_1, P_2, P_3 pont az $\underline{r}_0$, $\underline{r}_1$, $\underline{r}_2$ és $\underline{r}_3$ vektorokkal. Keressük azt az $\underline{r} = \underline{r}(t)$ vektor-skalár függvényt, mely kielégíti a következő feltételt: található olyan t_0, t_1, t_2 és t_3 paraméter-érték, hogy

$$\underline{r}_0 = \underline{r}(t_0), \quad \underline{r}_1 = \underline{r}(t_1), \quad \underline{r}_2 = \underline{r}(t_2), \quad \underline{r}_3 = \underline{r}(t_3)$$

teljesül.

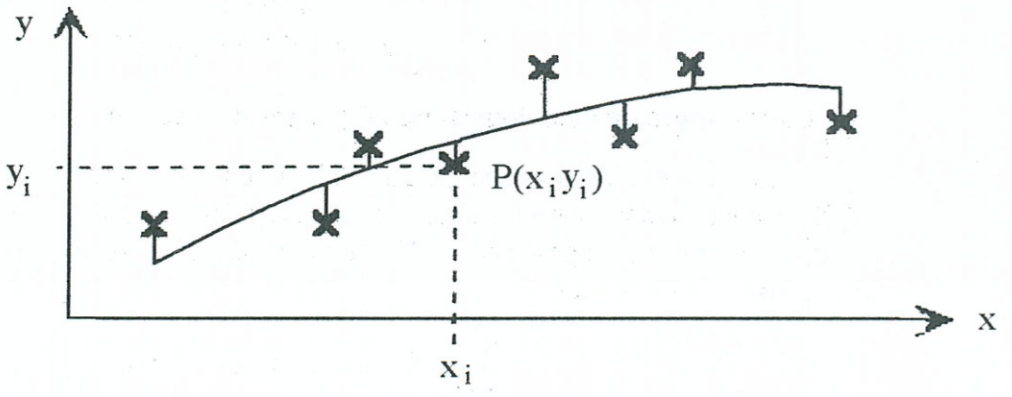
Ebben az esetben az $\underline{r}(t)$ vektor-skalár függvényt interpolációs görbének, a P_0, P_1, P_2, P_3 pontokat pedig az interpolációs görbe kontrollpontjainak nevezzük (lásd 56. sz. ábra).



56. sz. ábra
Interpolációs görbe és tartópontjai

A térbeli görbék és felületek modellezésének másik elterjedt módszere az approximáció. Ebben az esetben egy görbecsaládból (például *n*-edfokú polinomok) azt a görbét választjuk ki, mely az előre megadott P_1, P_2 ... P_n térbeli pontokat a lehető legjobban megközelíti.

Az approximációs görbe általában nem halad át a megadott kontroll pontokon (legalábbis nem mindegyiken) csak azoktól mért „távolsága" a lehető legkisebb. Másképp fogalmazva approximáció esetén a közelítő görbe kontrollpontjainak csak egy része helyezkedik el a görbén (lásd 57. sz. ábra).



57. sz. ábra
Approximációs görbe és kontrollpontjai

Példák az interpolációs és approximációs feladat megoldására.

Lagrange-interpoláció

Adott $\underline{P}_0$, $\underline{P}_1$, $\underline{P}_2$... $\underline{P}_n$ pont helyvektorával a térben, és különböző t_0, t_1, t_2 ... t_n paraméterérték. Keressük azt az $\underline{r}(t)$ – komponensként t legfeljebb n-edfokú polinomjaiból álló függvényt, melyre $\underline{r}(t_0) = \underline{p}_0$ ... $r(t_n)=p_n$

Megoldás:

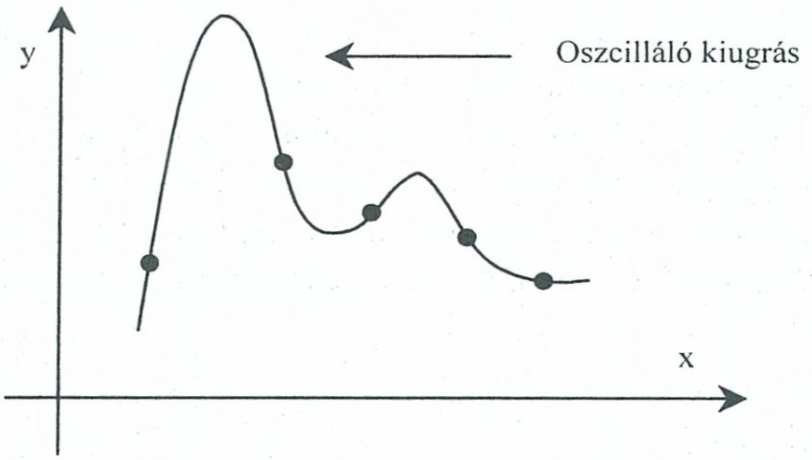
$$\underline{r}(t) = \underline{p}_0 \cdot L_0(t) + \underline{p}_1 \cdot L_1(t) \ldots + \underline{p}_n L_n(t), \quad \text{ahol}$$

$$L_i(t) = \frac{(t-t_0)\ldots(t-t_{i-1})(t-t_{i+1})\ldots(t-t_n)}{(t_i-t_0)\ldots(t_i-t_{i-1})(t_i-(t_{i+1})\ldots(t_i-t_n)}$$

az úgynevezett Lagrange-interpolációs polinomok. Könnyen látható, hogy

$$L_i(t_i) = 1 \quad \text{és} \quad L_i(t_j) = 0 \qquad i \neq j \quad \text{esetén}$$

A Lagrange-interpoláció sajnos nem igazán alkalmas görbemodellezésre a számítógépes grafikában, mivel esetenként két pont között erős kiugrások vannak a görbén (lásd 58. sz. ábra).



58. sz. ábra
Lagrange-interpolációs polinom

Gauss-approximáció (legkisebb négyzetek módszere)

Adott n darab pont a térben, például valamilyen mérés eredményei. Választunk egy függvénytípust (például polinom, exponenciális függvény), jelölje ezt $\underline{r}(t)$. Az $\underline{r}(t)$ paramétereit úgy válasszuk meg, hogy a pontoktól mért távolságainak négyzetösszege minimális legyen. Síkbeli $y = f(x)$ esetében:

$$\sum_{i=1}^{n} (f(x_i) - y_i)^2 = \min$$

ahol (x_i, y_i) $i = 1,2, \ldots n$ az adott pontok koordinátái.

Például ezt az eljárást alkalmazzák lineáris trendek számításánál, amikor $f(x)=ax+b$ alakú.

3.4.2. Interpoláció és approximáció polinomokkal

A térbeli görbék és felületek interpolációval vagy approximációval történő modellezése a számítógépes grafikában többnyire elegendő a feladatok megoldásához, mivel nem a precíz függvényábrázolás a célunk, hanem a szép, „sima" görbék rajzolása.

Ahhoz, hogy az interpolációs vagy approximációs eljárás a gyakorlatban is használható legyen, valamilyen módon szűkítenünk kell a szóba jöhető végtelen sok $\underline{r} = \underline{r}(t)$ vektor-skalár függvény halmazát.

Az egyszerű számíthatóság szempontjából az tűnik ésszerű választásnak, ha az összes lehetséges

$$\underline{r}(t) = \begin{cases} x(t) \\ y(t) \\ z(t) \end{cases}$$

függvények közül csak a polinomokat vesszük figyelembe, azaz az

$$x(t) = a_{x0} + a_{x1} \cdot t + a_{x2} \cdot t^2 + \ldots + a_{xn} \cdot t^n$$
$$y(t) = a_{y0} + a_{y1} \cdot t + a_{y2} \cdot t^2 + \ldots + a_{yn} \cdot t^n$$
$$z(t) = a_{z0} + a_{z1} \cdot t + a_{z2} \cdot t^2 + \ldots + a_{zn} \cdot t^n$$

alakú függvények között keressük a feladatnak megfelelőket, ahol a_{ij}=konstans.

Az ehhez szükséges vizsgálatokhoz a Taylor-sorfejtés matematikáját használjuk fel. Ezek szerint:

Tegyük fel, hogy az *f* függvény (*n*+1)-szer differenciálható valamely (*a*-t tartalmazó) *I* intervallumon. Ekkor

$$t_n(x) = f(a) + \frac{f(a)}{1!}(x-a) + \frac{f'(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n$$

az *f* függvénynek az „*a*" helyhez tartozó Taylor-polinomja. Ekkor bármely $x \in I$ esetén van legalább egy olyan ξ pont az *a* és *x* között, amelyre

$$R_m(x) = f(x) - t_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}$$

Ebből a matematikai tételből következik, hogy az

$$f(x)=\sum_{k=0}^{n}(x-a)^k+\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}$$

Taylor-polinommal (az $R_n(x)$-nek megfelelő hibával) bármely végtelen sokszor differenciálható $f(x)$ függvény tetszőleges pontossággal megközelíthető.

Emiatt polinomokkal rendkívül változatos alakú függvényeket is modellezhetünk, ugyanis például az összes ún. elemei függvény (sin$x$, cos$x$, ln$x$, e^x stb.) tetszőlegesen sokszor differenciálható.

Most már csak az a kérdés, hogy hányadfokú polinomokat használjunk a számítógépes grafikában az interpolációs és approximációs feladatok megoldásához. *Nyilvánvaló, minél kisebb a polinom fokszáma, annál kevesebb művelettel számíthatjuk ki a függvényértékeket. Viszont túl alacsony fokszámú polinomot sem választhatunk, mivel akkor a „bonyolultabb" görbéket és felületeket nem tudnánk jól közelíteni.* Például *másodfokú polinom esetén*

!

$$\underline{r}=\underline{a}_0+\underline{a}_1\cdot t+\underline{a}_2\cdot t^2$$

a három adott $\underline{r}_0$, $\underline{r}_1$, $\underline{r}_2$ pont már egyértelműen meghatározná a polinom $\underline{a}_0$, $\underline{a}_1$, $\underline{a}_2$ együtthatóit. Ez a három vektor viszont egy síkot határoz meg, melyben mindig benne lesz a polinommal generált ívdarab. Így másodfokú polinommal csak *síkbeli görbeszakaszokat tudnánk modellezni.*

!

Ezért a számítógépes grafikában a térgörbék és felületek modellezésére a harmadfokú polinomokat választották. Ezt a döntést az is indokolja, hogy harmadfokú polinomokkal modellezhetők olyan geometriai tulajdonságok, mint az önmetszés, csúcspont, vagy az inflexiós pont (ahol az érintő átmetszi a görbét). *Tehát a térgörbéket az*

$$\underline{r}=\underline{a}_0+\underline{a}_1\cdot t+\underline{a}_2\cdot t^2+\underline{a}_3\cdot t^3$$

!

alakú köbös paraméteres ívekkel, a felületeket pedig kétparaméteres köbös felületekkel közelítjük (a felületeknél tehát $x(u,v)$, $y(u,v)$, $z(u,v)$ u és v harmadfokú polinomja).

3.4.3. A spline és a köbös kétparaméteres felületfolt (bicubic patch) fogalma

Az interpolálással, illetve approximálással képzett görbék és felületek előállítására két, lényegében eltérő módszer létezik:

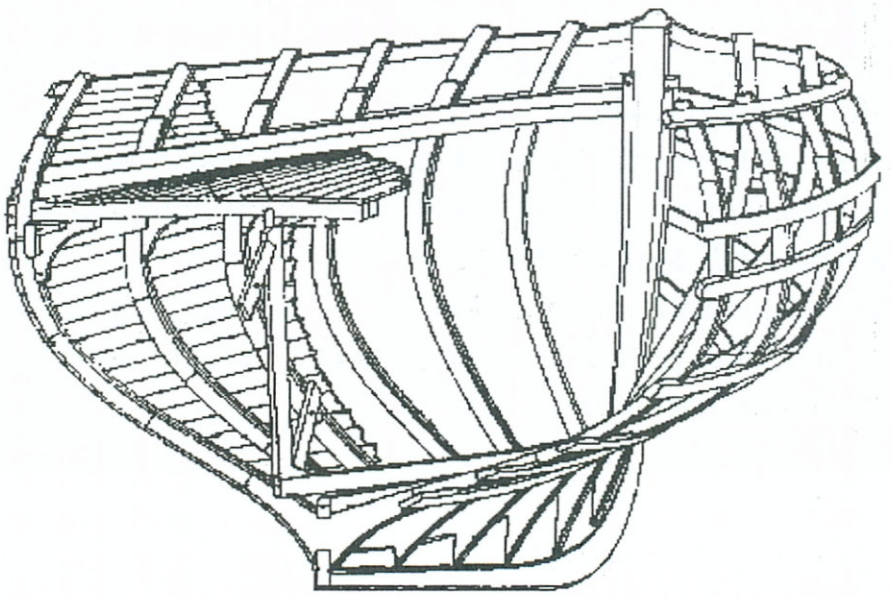
- *az összes interpolálandó vagy approximálandó pontot figyelembe véve egyetlen görbét vagy felületet határozunk meg.*
- *az interpoláló vagy approximáló görbét, illetve felületet egymáshoz folytonosan kapcsolódó részekből állítjuk össze.*

Az elmúlt években a második módszer alkalmazása vált majdnem egyeduralkodóvá a számítógépes grafikában.

Az egyetlen approximáló vagy interpoláló görbével vagy felülettel való modellezésnek a grafikus algoritmusok szempontjából az a hátránya, hogy a görbe alakja közvetlenül függ a kontrollpontok számától és a polinom fokszámától. Egy kontrollpont megváltoztatása kihat a teljes görbére. Erre a problémára kerestek megoldást a spline-ok alkalmazásával.

Ha a modellezendő térgörbét, több egymáshoz folytonosan kapcsolódó ívből állítjuk elő, akkor ezt a görbét spline-nak nevezzük.

Ez a fogalom a hajlékony (görbe) vonalzó angol nevéből származik, melyet a több egyforma hajót gyártó angol hajógyárakban alkalmaztak először (lásd az 59. sz. ábrát) a XIX. században.



59. sz. ábra
Hajóbordák gyártása spline-nal

A spline-ok tehát polinomívdarabokból álló görbék, mely ívdarabokat a paramétertartomány

$$t_0 \leq t_1 \leq \ldots \leq t_n$$

felosztásával képzünk minden $[t_i, t_{i+1}]$ *intervallumban külön-külön.*

Az ívdarabok összeillesztése a csatlakozópontokon folytonosan, differenciálisan történik. Ha a paramétertartományt azonos közökre osztjuk fel, azaz t_{i+1}-t_i, értéke minden „*i*"-re azonos, akkor a spline-t uniformnak nevezzük.

Spline felületek

Ha egy modellezendő felületet részekből állítunk össze, akkor ezt spline felületnek nevezzük. A spline felületek részeit leggyakrabban köbös polinomokkal generáljuk, ezeket két paraméteres köbös felületfoltoknak (bicubic patch) nevezzük.

3.4.4. Az ívdarabok és felületfoltok folytonos csatlakoztatása

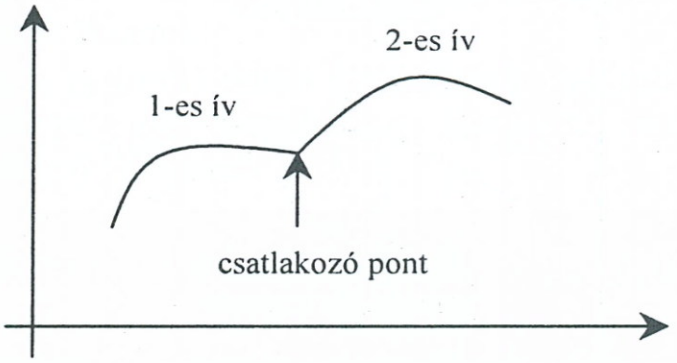
Ha a görbéket ívdarabokból, a felületeket pedig felületfoltokból „rakjuk össze", akkor fontos kérdés, hogy ezek milyen módon csatlakoztathatók folytonosan az illesztési pontokban.

Nyilván azt szeretnénk, ha az illeszkedés törésmentesen, „simán" történne, azaz a képernyőn való megjelenítéskor nem lehet észrevenni, hogy a görbe vagy a felület több részből áll.

Ezt a követelményt matematikailag az illesztési pontokban történő folytonos átmenetekkel tudjuk kezelni. Ezeket az egzakt definíció helyett ábrákon szemléltetve mutatjuk be.

C_0 matematikai folytonosság

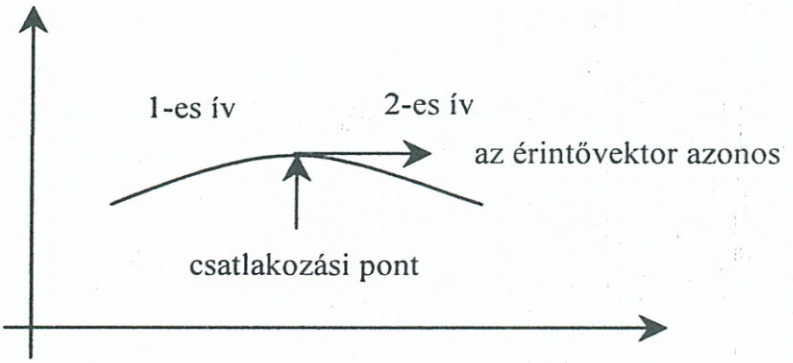
Ebben az esetben a két görbedarabnak az illesztési pontban lehet törése, azaz „kicsúcsosodhat" (lásd 60. sz. ábra), de hézagmentesen csatlakozik a két rész.



60. sz. ábra
Ívek C_0 folytonosságú illesztése

C_1 matematikai folytonosság

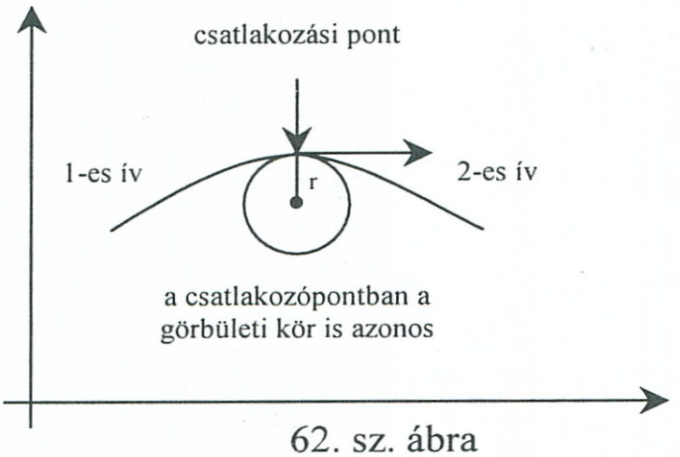
A két ívdarabnak az illesztési pontban az érintője is megegyezik (azaz első deriváltjuk azonos) (lásd 61. sz. ábra).



61. sz. ábra
Ívek C_1 folytonosságú illesztése

C_2 matematikai folytonosság

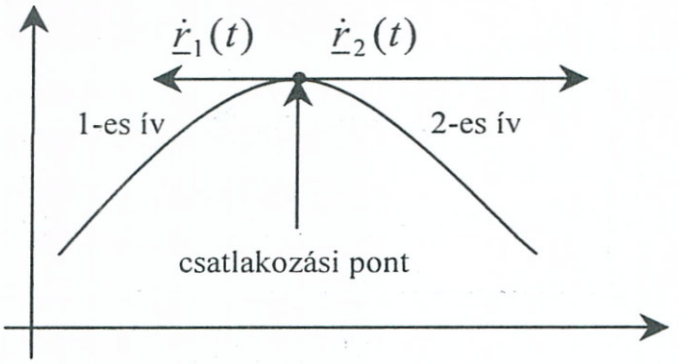
Ebben az esetben a két ívdarabnak a csatlakoztatási pontban nemcsak az érintője, hanem a görbülete is megegyezik (azaz az első és második derivált is azonos).



62. sz. ábra
Ívek C_2 folytonosságú illesztése

G_1 geometriai folytonosság

Ebben az esetben a két ívdarabnak a csatlakoztatási pontban nem kell azonos deriváltakkal rendelkeznie. Csak az érintőegyenes azonos, de a deriváltvektor nagysága és előjele eltérő lehet.



63. sz. ábra
Ívek G_1 folytonosságú illesztése

3.5. TÉRBELI GÖRBÉK MODELLEZÉSE

E fejezetben először a számítógépes grafikában használatos legfontosabb ívtípusokat (Hermite, Bézier, B-spline alapfüggvények) tekintjük át, majd az ezekből összerakható spline görbékkel és racionális változataikkal foglalkozunk.

A fejezetben szereplő interpolációs és extrapolációs feladatok mindig a következő formátumúak lesznek:

Legyen adva $\underline{r}_0 \ldots \underline{r}_n$ $(n+1)$ darab pont a térben. Keressük azt az $\underline{r}(t)$ vektor skalár-függvényt, mely ezeket a pontokat extrapolálja vagy approximálja a következő formában:

$$\underline{r}(t) = \underline{r}_0 \cdot f_0(t) + \underline{r}_1 \cdot f_1(t) + \ldots + \underline{r}_n \cdot f_n(t)$$

ahol $f_0, f_1 \ldots f_n$ a t paraméter egyváltozós függvényei polinomok.

A számítógépes grafika szempontjából meghatározó jelentőségű kérdés, hogy az $\underline{r}(t)$ függvény invariáns-e az affin transzformációkra. Az invariancia azért előnyös, mert ilyen tulajdonságú függvények összes pontja helyett elegendő az $\underline{r}_0$, $\underline{r}_1 \ldots \underline{r}_n$ kontrollpontokat transzformálni, ha például a görbét elmozgatjuk a térben. Ezzel jelentős mennyiségű számítást takaríthatunk meg. Mikor lesz egy $\underline{r}(t)$ függvény garantáltan invariáns az affin transzformációkra? Erre a kérdésre viszonylag egyszerűen megtalálhatjuk a választ:

Az $\underline{r}(t) = \sum_{i=0}^{n} \underline{r}_i f_i(t)$ alakú függvények, akkor invariánsak az affin transzformációra, ha

!

$$f_0(t) + f_1(t) + \ldots + f_n(t) = 1$$

minden t paraméterértékre.

Állításunk könnyen belátható. Legyen $\overline{\overline{A}}$ egy tetszőleges affin transzformáció mátrixa 4x4-es homogén koordinátákban megadva, és az $\underline{r}_0 \ldots \underline{r}_n$ vektorok normalizált homogén koordinátás megfelelői legyenek $\underline{r}_0^* \ldots \underline{r}_n^*$ Jelölje $\underline{r}^*(t)$ az $\underline{r}(t)$ vektor homogén koordinátás megfelelőjét. Ekkor

$$\overline{\overline{A}} \cdot \underline{r}^*(t) = \overline{\overline{A}} \left(\sum_{i=0}^{n} \underline{r}_i^* \cdot f_i(t) \right) = \sum_{i=0}^{n} \overline{\overline{A}} \cdot \underline{r}_i^* \cdot f_i(t)$$

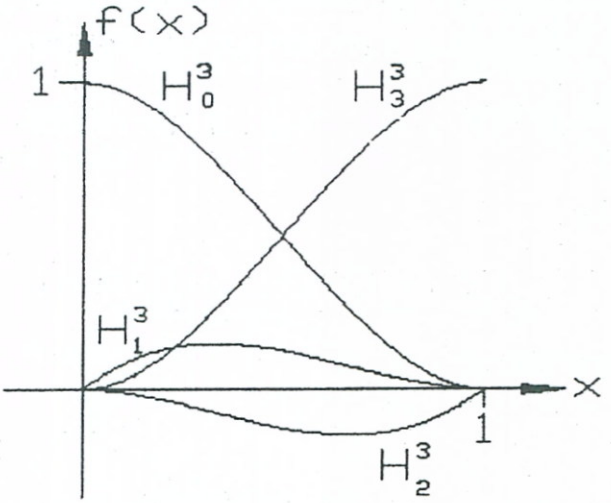
Mivel $\overline{\overline{A}}$ negyedik sora (0, 0, 0, 1) és az összes $\underline{r}^*$ 4-ik koordinátája 1-el egyenlő, ezért az előbbi összefüggést a 4-ik koordinátában vizsgálva adódik $\underline{r}_4^* \cdot (t) = \sum_{i=0}^{n} f_i(t) = 1$. Azaz $\underline{r}^*(t)$ transzformáltja a $\overline{\overline{A}} \cdot \underline{r}_i^*$ transzformáltjával előállítható.

3.5.1. Hermite interpolációs ívek

A Hermite-ívek képzése a Hermite 3-adfokú alappolinomokkal történik:

$$H_0(t) = (1-t)^2 \cdot (1+2t)$$
$$H_1(t) = t(1-t)^2$$
$$H_2(t) = -t^2(1-t)$$
$$H_3(t) = (3-2t)t^2$$

Ezeket a polinomokat a $t \in [0,1]$ intervallumon a 64. sz. ábra mutatja be.

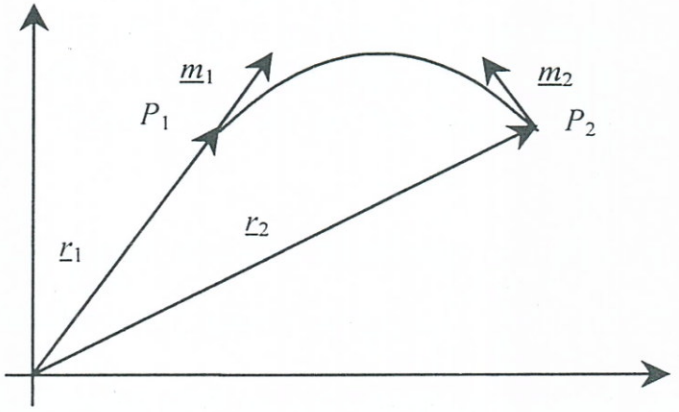


64. sz. ábra
Hermite-alappolinomok

Legyen most adva a térben a P_1 és P_2 pont az $\underline{r}_1$ és $\underline{r}_2$ vektorokkal, és legyen a generálandó ívdarab érintője P_1-ben az $\underline{m}_1$, P_2-ben pedig az $\underline{m}_2$ vektor. Ekkor a Hermite-ívet az

$$\underline{r}(t) = \underline{r}_1 H_0(t) + \underline{m}_1 H_1(t) + \underline{m}_2 H_2(t) + \underline{r}_2 H_3(t)$$

kifejezéssel számíthatjuk ki. Erre szemléletes példát 2D-ben a 65. sz. ábra mutat.



65. sz. ábra
Hermite-ív képzése

Szemléletesen látható, hogy a Hermite-ívekből törésmentesen (C_1 matematikai folytonossággal) tudunk spline térgörbéket összeállítani. Ezeket az jellemzi, hogy a görbe áthalad a tartópontokon. Ugyanakkor a Hermite-íveknek a számítógépes grafikában történő alkalmazásában gondot okoz, hogy ezek a paramétertartomány affin transzformációira nem invariánsak.

3.5.2. Bézier-approximációs ívek, de Casteljau-algoritmus

A Bézier-ívekkel történő modellezést különösen az autógyártás CAD rendszereiben alkalmazzák, ennek célja a megvilágított karosszéria törésmentes fényjátéka. Maga az eljárás is innen származik, ezt majdnem egyidejűleg dolgozták ki P. Bézier a Renault és P. de Casteljau a Citroën autógyárak tervezőmérnökei 1962-ben, illetve 1959-ben.

A harmadfokú Bézier-íveket [jelölése $\underline{B}_3(t)$] a következő képlettel definiálhatjuk, ha adott a görbe P_0, P_1, P_2, P_3 kontroll pontja az $\underline{r}_0$, $\underline{r}_1$, $\underline{r}_2$ és $\underline{r}_3$ vektorokkal:

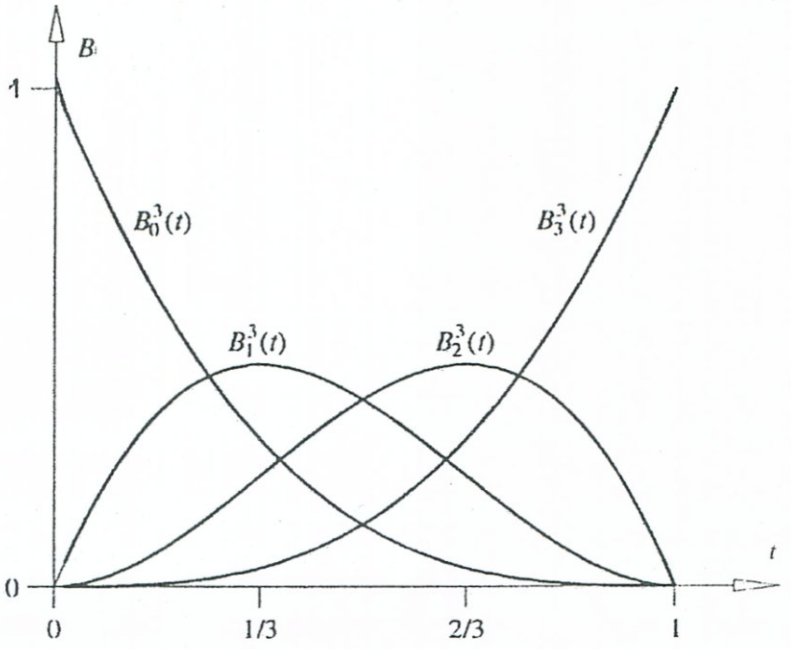
$$\underline{B}_3(t) = \sum_{j=0}^{3} \underline{r}_i \cdot B_i^3(t) \qquad t \in [0,1]$$

ahol $B_i^3(t) = \binom{3}{i} \cdot t^i (1-t)^{3-i}$ $i = 0, 1, 2, 3$ $0 \le t \le 1$

(A $B_i^3(t)$ kifejezésben a 3 nem hatványkitevő, hanem felső index, ami a polinom fokszámát jelöli.)

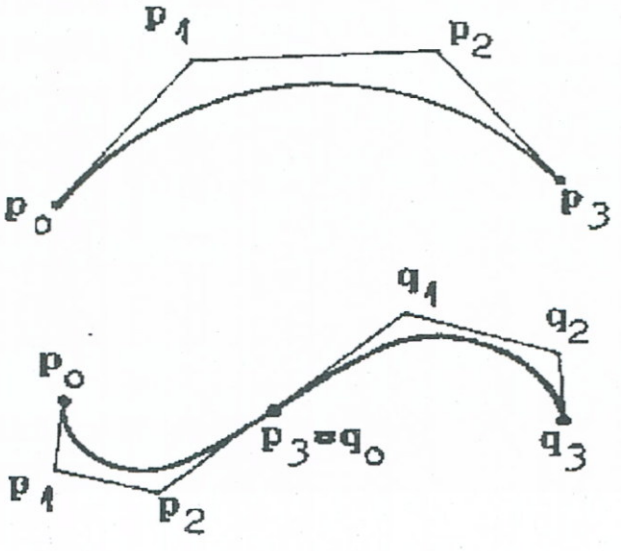
A $B_i^3(t)$ függvényeket skalár súlyfüggvényeknek (blending functions) nevezik, ezek az úgynevezett Bernstein-polinomok.

A harmadfokú Bernstein-polinomokat mutatja a 66. sz. ábra.



66. sz. ábra
Harmadfokú Bernstein-polinomok

Szemléletesen a harmadfokú Bézier-íveket a következőképpen ábrázolhatjuk:



67. sz. ábra
3-adfokú Bézier-ív és két Bézier-ív folytonos csatlakoztatása

Ezek szerint a Bézier-ívek tartópontjai közül kettő (P_0 és P_3) a görbén helyezkedik el, kettő pedig (P_1, P_2) a görbe két végpontjában húzott érintőkön található. A 63. sz. ábrán azt is láthatjuk, hogy a Bézier-íveket egymáshoz C_2 folytonossággal illeszthetik, és az így létrejövő összetett ív szintén Bézier-ív lesz.

A harmadfokú esethez hasonlóan definiálhatjuk az (n+1) db $\underline{r}_0,\underline{r}_1...\underline{r}_n$ kontrollponttal adott n-edfokú Bézier-íveket. Legyenek $B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}\ i = 0,1,...n$ az n-edfokú Bernstein-polinomok. Ekkor az (n+1) kontrollponttal definiált n-edfokú Bézier-görbét a következő összefüggés adja meg:

$$\underline{B}_n(t) = \sum_{j=0}^{n} \underline{r}_i B_j^n(t) \qquad t \in [0,1]$$

Az n-edfokú Bézier-ív az első és utolsó kontrollponton áthalad, azaz a végpontokban interpolál.

Ez a Bernstein-polinomoknak abból a tulajdonságából következik, hogy

$$\underline{B}_i^n(0) = \begin{cases} 1 & \text{ha } i = 0 \\ 0 & \text{egyébként} \end{cases} \quad \text{és} \quad \underline{B}_i^n(1) = \begin{cases} 1 & \text{ha } i = n \\ 0 & \text{egyébként} \end{cases}$$

A definíciónál az egyszerűség kedvéért a $t \in [0,1]$ paramétertartományt választottuk. Ezt azért tehetjük meg, mert a Bézier-ívek a paraméter tartomány affin transzformációira invariánsak, tehát a $t \in [0,1]$ esetből levont következtetések tetszőleges paramétertartomány esetén is érvényesek.

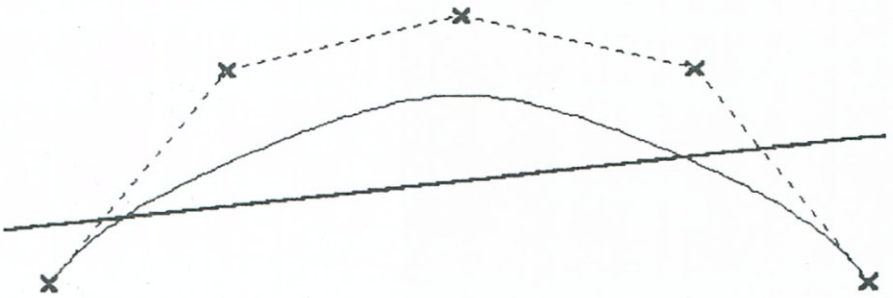
A Bézier-ívek néhány, a számítógépes grafikában fontos tulajdonsága:

- *a görbe íve mindig a P_0, P_1, P_2, P_3 kontrollpontok által meghatározott négyszög „belsejében" helyezkedik el.*

Precízebben fogalmazva az $\underline{r}_0$ ... $\underline{r}_n$ kontrollpontokkal rendelkező $\underline{B}_n(t)$ Bézier-ív a kontrollpontjainak konvex burkán belül halad.

A konvex burok fogalmához először a konvex ponthalmazt kell definiálnunk. Egy H térbeli ponthalmaz konvex, ha bármely két hozzátartozó pont esetén az ezeket összekötő szakaszt is tartalmazza. Egy G ponthalmaz konvex burkán az összes olyan konvex H halmaz metszetét értjük, melyek G-t tartalmazzák. Állításunk abból következik, hogy $\sum_{i=0}^{n} B_i^n(t) = \sum_{i=0}^{n} \binom{n}{i} \cdot (1-t)^{n-i} \cdot t^i = (1-t+t)^n = 1$. Ezért a $\underline{B}_n(t)$ Bézier-görbe az $\underline{r}_0...\underline{r}_n$ kontrollpontok konvex lineáris kombinációja.

- *A Bézier-ívek kontrollpontjaik affin transzformációjával szemben invariánsak. Ez azt jelenti, hogy például a görbe mozgatásakor elegendő a kontrollpontokat transzformálni, mellyel igen sok számítás megtakarítható.* (Ez azonban csak az affin transzformációkra igaz és nem minden esetben teljesül a projektív transzformációkra, például a centrális vetítésre.)
- *A Bézier-ívek a paramétertartomány affin transzformációira invariánsak.*
- *A Bézier-ívek globálisan változtathatók, azaz ha a kontrollpontok közül egyet elmozgatunk, az az egész görbére kihat.*
- *Egy egyenes pontosan annyi pontban metszi a Bézier-ívet, ahány metszéspontja van a tartónégyszögével* (lásd 68. sz. ábra) (Variation diminishing property).



68. sz. ábra
A Bézier-ívek variation diminishing tulajdonsága

A Bézier-ívek megjelenítése szempontjából nagyon fontos, hogy létezik-e olyan algoritmus, mely alapján az ív egyenesszakaszokkal közelíthető. Többek között erre a problémára ad választ a de Casteljau-algoritmus. Ez egy rekurzív eljárást ad a Bézier-ív egy konkrét „t" paraméterértékhez tartozó pontjának kiszámítására és az ív egyenesszakaszokkal való közelítésére.

A de Casteljau-algoritmus lényege a következő:
Legyen adva egy harmadfokú Bézier-ív P_0, P_1, P_2, P_3 kontrollpontjainak megfelelő $\underline{r}_0$, $\underline{r}_1$, $\underline{r}_2$ és $\underline{r}_3$ vektor. Legyen $\underline{r}_i^0(t) = \underline{r}_i \qquad i = 0,1,2,3$
Definiáljuk a következő rekurzív összefüggést:

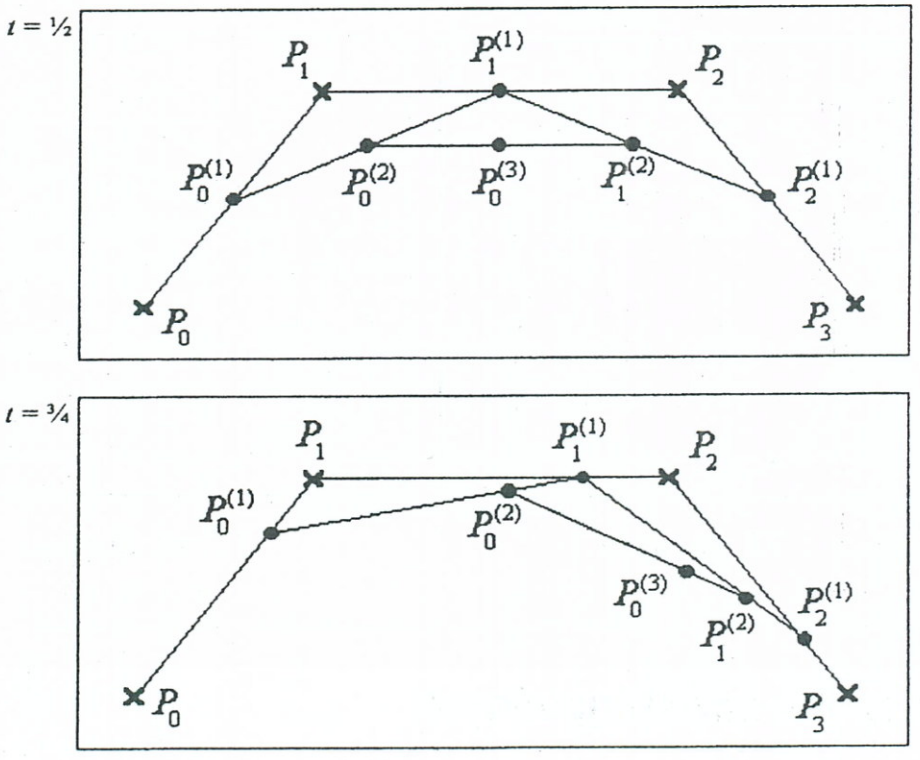
$$\underline{r}_i^s(t) = (1-t)\underline{r}_i^{s-1}(t) + t\underline{r}_{i+1}^{s-1}(t) \qquad \text{ahol} \quad \begin{matrix} s = 1,2,3 \\ i = 0,1,\ldots,(3-s) \end{matrix}$$

Ennek lényege: az $\underline{r}_i^s(t)$ -nek megfelelő P_i^s pont az $\underline{r}_i^{s-1}(t)$ és $\underline{r}_{i+1}^{s-1}(t)$ -nek megfelelő P_i^{s-1} és P_{i+1}^{s-1} pontokat összekötő szakasz egy belső pontja, melyet az (1-*t*) és a t súlyok jelölnek ki. Ez konvex lineáris kombináció, mivel (1-*t*)+*t* = 1.

A rekurzióval kapott vektorokat a következő sémában szokták elrendezni:

$$
\begin{array}{ccccc}
\underline{r}_0 & & & & P_0 \\
& \underline{r}_0^1 & & & P_0^1 \\
\underline{r}_1 & & \underline{r}_0^2 & & P_1 \quad P_0^2 \\
& \underline{r}_1^1 & & \underline{r}_0^3 & \text{vagy a pontokkal} \quad P_1^1 \quad P_0^3 \\
\underline{r}_2 & & \underline{r}_1^2 & & P_2 \quad P_1^2 \\
& \underline{r}_2^1 & & & P_2^1 \\
\underline{r}_3 & & & & P_3
\end{array}
$$

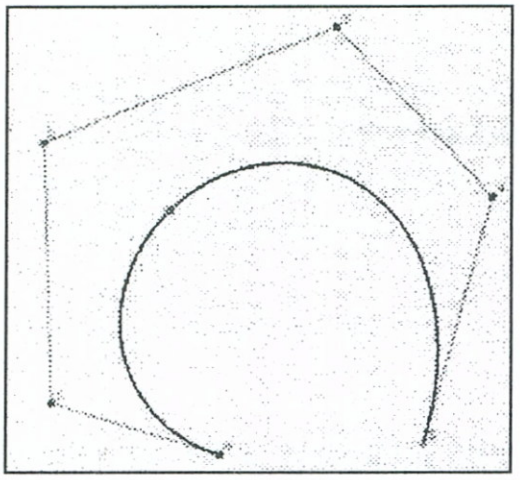
A de Casteljau-algoritmus geometriai jelentését a 69. sz. ábrán láthatjuk.



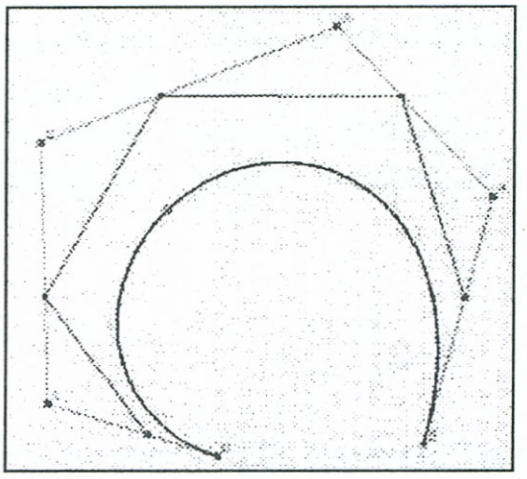
69. sz. ábra
Bézier-görbe $t = ½$ és $t = ¾$ pontjának szerkesztése a de Casteljau-algoritmussal

Az ábrán látható P_0^3 pontok a Bézier-ív egy-egy pontját állítják elő a t=½ és ¾ paraméterértékek esetén. Az utolsó P_0^3-at tartalmazó egyenes-szakasz egyúttal az ív érintőjét is előállítja. Az ábrából is látható, hogy az érintő szakaszok nagyon gyorsan közelítik (konvergálnak) az ívet.

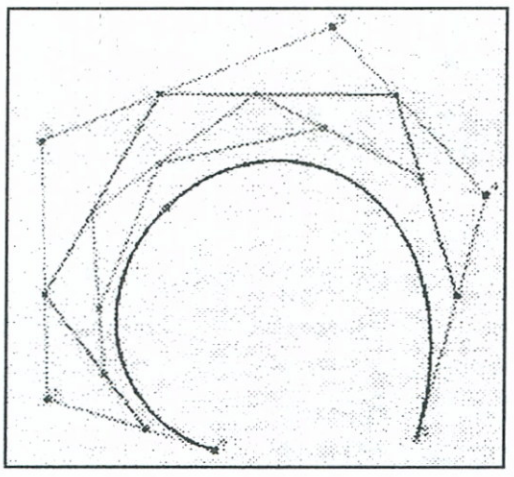
A de Casteljau-algoritmus programmal történő végrehajtására mutatnak példát a 70., 71., 72., 73. sz. ábrák.



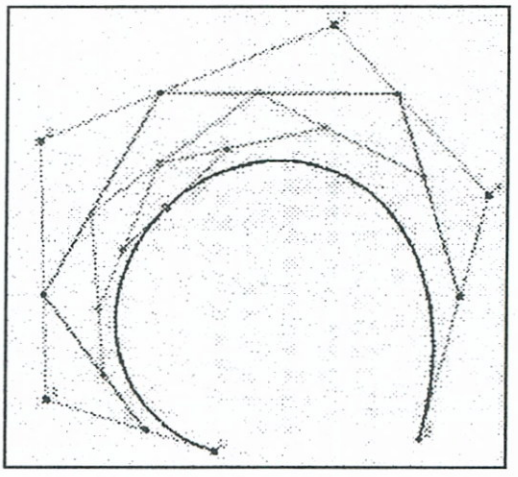
70. sz. ábra



71. sz. ábra



72. sz. ábra



73. sz. ábra

3.5.3. B-spline bázisfüggvények

Ha a térbeli görbéket spline-okkal modellezzük, akkor ezeket részenként ív-darabokból állítjuk elő (lásd 3.5.4. fejezet). Attól függően, hogy a részívdarabokhoz milyen ívtípust választunk, különböző fajta spline-okat

kapunk. Ismeretesek például a Bézier-spline-ok, amikor a spline ívdarabjai Bézier-ívek. C_1 folytonosságú interpolációs spline-okat kapunk, ha például az ívdarabok Hermite-ívekből állnak.

Legelterjedtebben a számítógépes grafikában a *spline-ok generálásához részívnek úgynevezett B-spline bázisfüggvényeket választanak. Ekkor az így létrejövő íveket B-spline görbéknek nevezzük.* Ezek a matematikában már több mint száz éve ismertek voltak, de (elsőként CAD feladatokra) a számítógépes grafikában csak a hetvenes években kezdtek a B-spline-okat alkalmazni. A B-spline-ok főként C. de Boor, M. Cox és L. Mansfield munkássága nyomán kezdtek elterjedni, akik felfedezték a bázisfüggvények rekurzív tulajdonságait és megalkották a számítógépes kezelésükhöz szükséges algoritmusokat.

!

A normalizált B-spline bázisfüggvényeket a következőképpen definiálhatjuk:

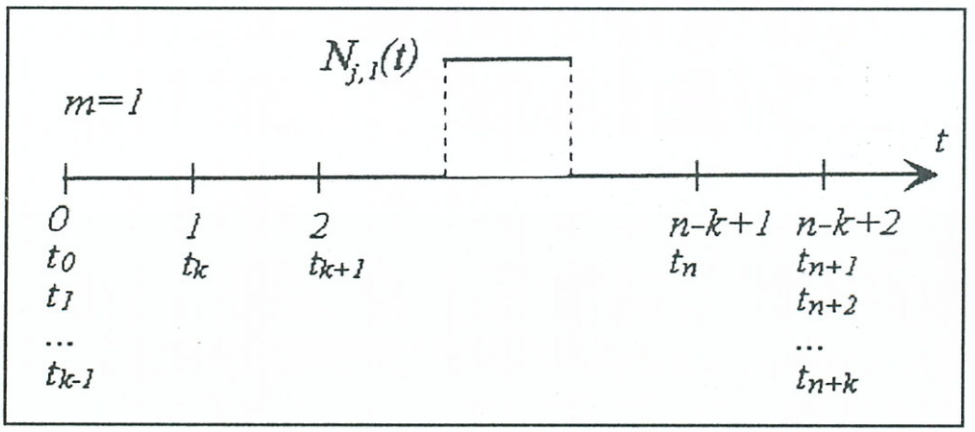
Legyen T egy véges vagy végtelen intervallum, és osszuk fel az intervallumot $t_i \le t_{i+1}$ osztópontokkal. Az $N_{j,m}(t)$ B-spline bázisfüggvényeket ezt követően az alábbi rekurziós összefüggésekkel határozhatjuk meg:

$$N_{j,1}(t) = \begin{cases} 1 & \text{ha} \quad t_j \le t < t_{j+1} \\ 0 & \text{különben} \end{cases}$$

$$N_{j,m}(t) = \frac{(t - t_j) \cdot N_{j,m-1}(t)}{t_{j+m-1} - t_j} + \frac{(t_{j+m} - t) \cdot N_{j+1,m-1}(t)}{t_{j+m} - t_{j+1}}$$

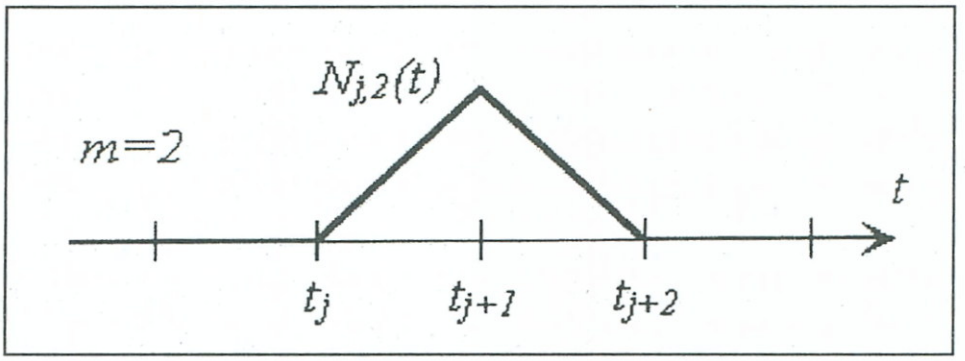
A t_j skalárokat csomóértékeknek nevezzük (knot values). (A rekurziós képletben előforduló 0/0 értéket definíció szerint tekintsük 0-nak.)

A normalizált B-spline $N_{j,m}(t)$ bázisfüggvények legfeljebb m-1 fokú polinomok. Ezekre mutatnak néhány példát a következő ábrák.



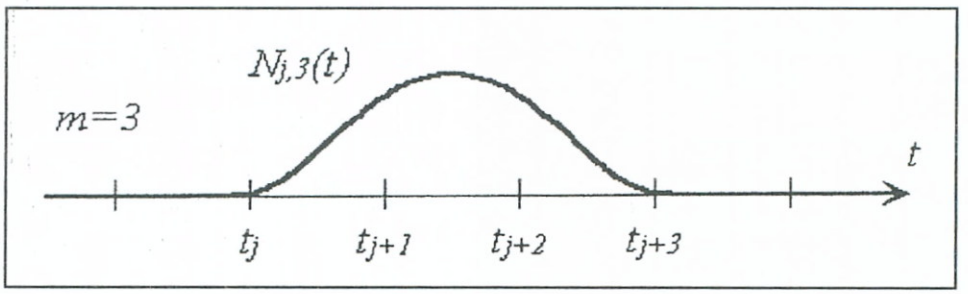
74. sz. ábra
Az $N_{j,1}(t)$ normalizált B-spline bázisfüggvény

Látható, hogy $N_{j,1}(t)$ szakaszonként 0, vagy 1 értéket felvevő úgynevezett „lépcsős" függvény.



75. sz. ábra
Az $N_{j,2}(t)$ normalizált B-spline bázisfüggvény

Az $N_{j,2}$ elsőfokú polinomoknak megfelelő szakaszokból van összerakva, mely két paraméter intervallumra terjed ki.



76. sz. ábra
Az $N_{j,3}(t)$ normalizált B-spline bázisfüggvény

Az $N_{j,3}(t)$ egy másodfokú polinom, mely három intervallumban különbözik a 0-tól.

Mindegyik normalizált B-spline bázisfüggvény rendelkezik a következő tulajdonságokkal:

- $N_{j,m}(t)=0$, ha t nincs benne a $[t_j, t_{j+m}]$ intervallumban.
- $N_{j,m}(t)\geq 0$ bármilyen t esetén
- $\sum_j N_{j,m}(t) = 1$, azaz egy konkrét m értékhez tatozó B-spline bázisfüggvények összege 1.

A B-spline bázisfüggvények matematikailag a Bernstein-polinomok általánosításainak tekinthetők.

3.5.4. B-spline görbék, a Cox–de Boor-féle algoritmus

A Bézier-ívek – mint ezt a 3.5.2. fejezetben láttuk –, a vetítésekre vonatkozó invarianciától eltekintve a görbék modellezésével szemben támasztott összes követelménynek (lásd 3.3. fejezet) eleget tesznek, kivéve a következőket:

- ha a kontrollpontok száma nagy, az algoritmusok számításigénye rendkívül megnő,
- a kontrollpontok megváltoztatása az egész görbe alakját megváltoztatja.

Ezt a nehézséget küszöbölhetjük ki a B-spline görbék alkalmazásával.

Legyen adva P_0, P_1... P_n $(n+1)$ db kontrollpont az $\underline{r}_0$, $\underline{r}_1$... $\underline{r}_n$ vektorokkal (B-spline görbék esetén ezeket Boor-pontoknak nevezzük). Ekkor egy B-spline görbét a következő képlettel határozhatunk meg:

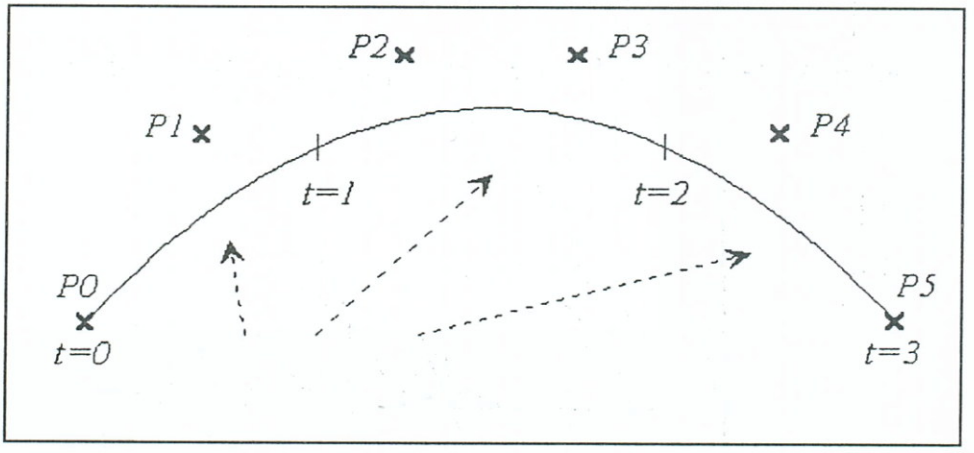
$$B_{n,k}(t) = \sum_{i=0}^{n} \underline{r}_i \cdot N_{i,k}(t)$$

Az összefüggésben szereplő $N_{i,k}(t)$ B-spline bázisfüggvényeket súlyfüggvényeknek is nevezik (blending functions).

A P_0 ... P_n kontrollpontok által meghatározott poligont Boor-poligonnak nevezzük.

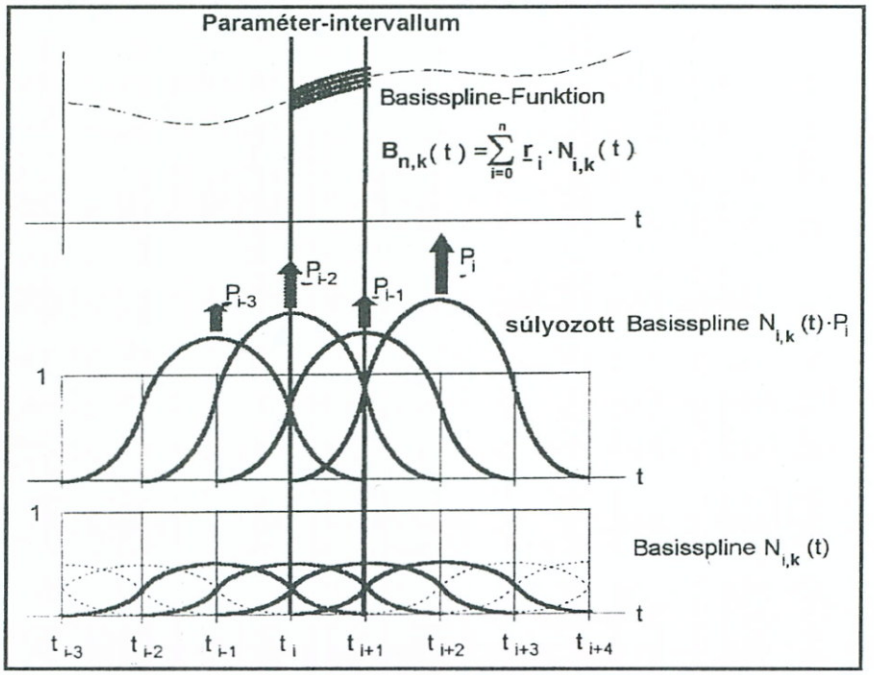
A k értékét a B-spline görbe rendjének nevezzük. (A gyakorlatban k értéke 4, ekkor köbös polinomokat használunk.)

A 77. sz. ábrán egy 3 részből álló B-spline-t mutatunk be, melynek 6 kontrollpontja van.



77. sz. ábra
Hat kontrollpontos B-spline

Tehát a súlyozott B-spline alappolinomok összegzésével jön létre a B-spline görbe, ezt mutatja be a 78. sz. ábra.

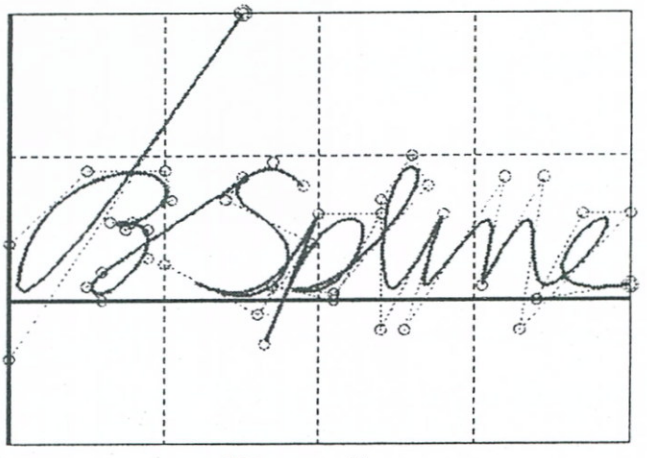


78. sz. ábra
B-spline görbe képzése bázis spline-okkal

! *Matematikailag bizonyítható, hogy a B-spline görbék rendelkeznek a Bézier-görbék összes előnyös tulajdonságával és ezen túlmenően lokálisan is változtathatók, azaz egy kontrollpont elmozgatása csak a neki megfelelő* $[t_i, t_{i+k}]$ *intervallumban módosítja a görbe alakját.*

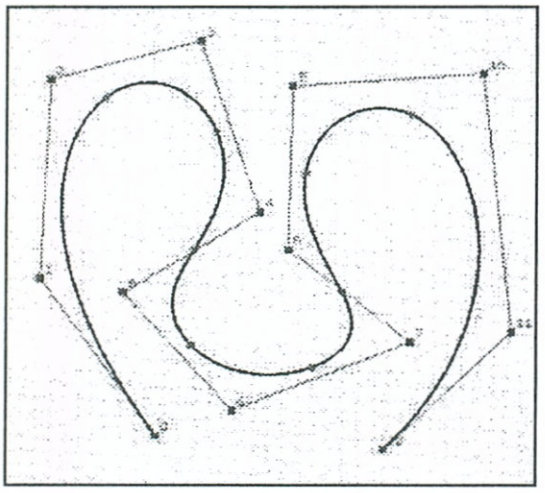
Ezt az állítást könnyen beláthatjuk, ha figyelembe vesszük, hogy minden egyes $N_{i,k}(t)=0$, ha t nincs benne a $[t_i, t_{i+k}]$ intervallumban.

Egy komplex B-spline görbére mutat példát a 79. sz. ábra.

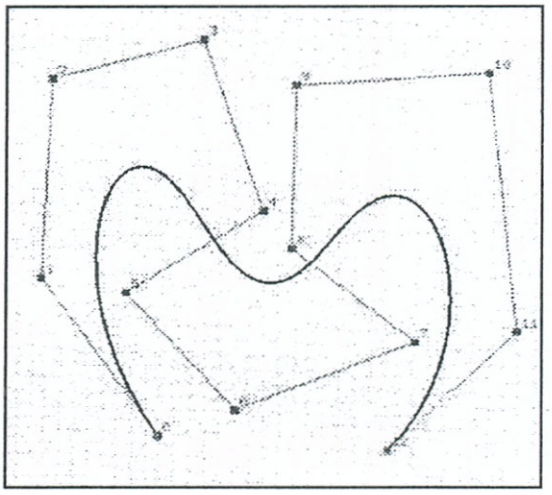


79. sz. ábra
Komplex B-spline görbe

A B-spline görbék jellemzője, hogy jobban „simulnak" a kontrollpoligonjukhoz, mint a Bézier-görbék. Erre mutatnak példát a következő ábrák, melyeken azonos kontrollpoligon esetén kirajzolt B-spline, illetve Bézier-görbe található.



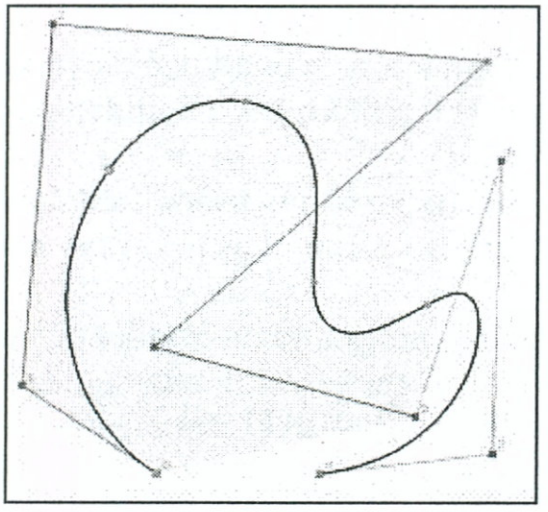
80. sz. ábra
B-spline görbe



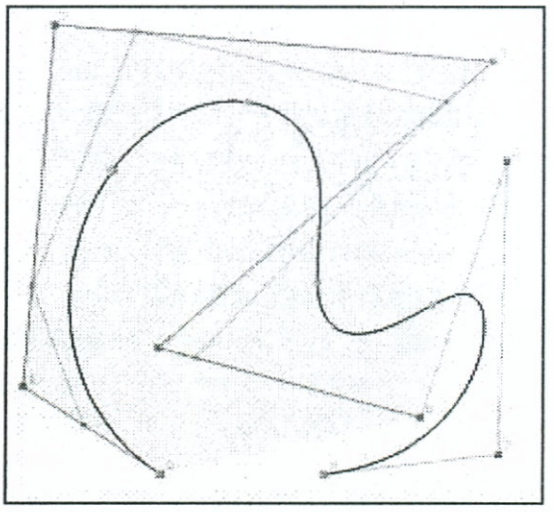
81. sz. ábra
Bézier-görbe

A B-spline görbék ugyanúgy, mint a Bézier-görbék, *közelíthetők poligonokkal a kontrollpoligon szakaszainak rekurzív felosztásával, és ez az algoritmus elő is állítja a B-spline görbe egy pontját egy konkrét t értékre. Ekkor a közelítő sokszögek a görbéhez tartanak határértékben. Ezzel egy hatékony eljárást kapunk a B-spline-ok raszteres képernyőn való megjelenítésére. Ezt az eljárást Cox-de Boor-algoritmusnak nevezzük, mely a de Casteljau-algoritmus általánosítása.*

Erre mutatnak példát a következő ábrák.

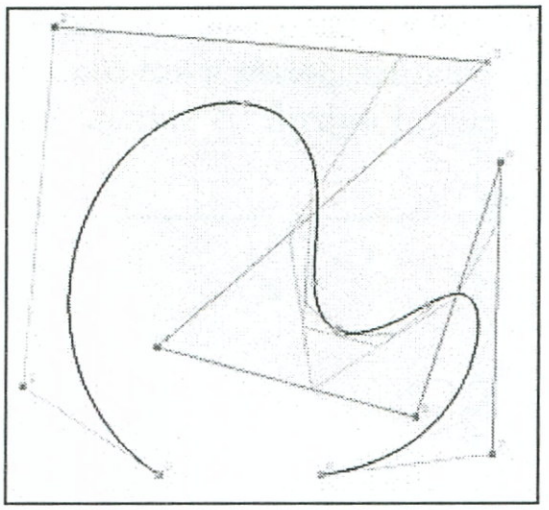


82. sz. ábra

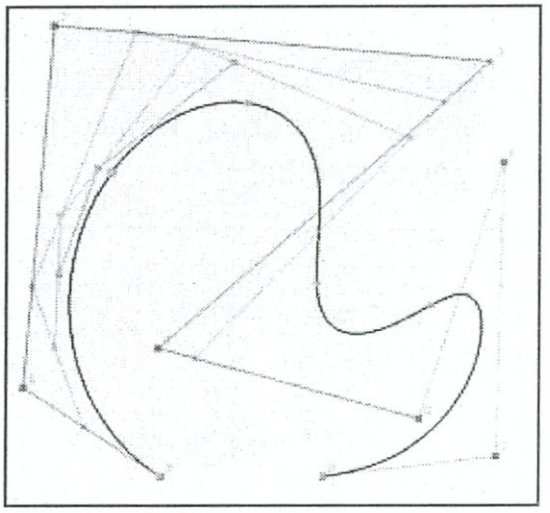


83. sz. ábra

A Cox-de Boor-algoritmus



84. sz. ábra



85. sz. ábra

A Cox-de Boor-algoritmus

3.5.5. Racionális B-spline görbék, NURBS

A B-spline görbék már majdnem kielégítik a modellezéssel kapcsolatos összes követelményt, de még mindig nem rendelkeznek a centrális vetítésekre vonatkozó invariancia tulajdonságával. Ezért vezették be a racionális görbéket a számítógépes grafikában, melyeket először S. A. Coons alkalmazott.

Napjainkban legelterjedtebben a racionális görbéket használják a grafikus szoftverekben a térgörbék modellezésére. Ennek oka, hogy sokfajta alakváltoztatási lehetőséget biztosítanak, segítségükkel az összes ismertebb görbe modellezhető és ami legfontosabb, invariánsak a vetítésekre (általában a projektív transzformációkra).

Különösen a perspektivikus transzformációra vonatkozó invariancia fontos a számítógépes grafikában, mivel a 3D-s alakzatok megjelenítéséhez erre nagyon gyakran szükség van.

A racionális görbéknek a vetítésekre vonatkozó invariáns tulajdonsága miatt a képernyőn történő ábrázolás előkészítése során elegendő csak a kontrollpontok képét kiszámítani.

Legyenek $\underline{r}_0,\underline{r}_1,...\underline{r}_n$ egy térgörbe kontrollpontjaihoz vezető helyvektorok, $S_{0,n}(t),S_{1,n}(t),...S_{n,n}$ súlyfüggvények, $\sum_{i=0}^{n} S_{i,n}(t) = 1$, $w_0,w_1,...w_n$ nemnegatív valós számok. Ekkor az

$$\underline{R}_n(t) = \frac{S_{0,n}(t) \cdot \underline{r}_0 \cdot w_0 + S_{1,n}(t) \cdot \underline{r}_1 \cdot w_1 + \ldots + S_{n,n}(t) \cdot \underline{r}_n \cdot w_n}{S_{0,n}(t) \cdot w_0 + S_{1,n}(t) \cdot w_1 + \ldots + S_{n,n} \cdot w_n}$$

egy n-edrendű racionális görbét határoz meg.

A definíció „szemléletes" jelentése a következő:

Az (x, y, z, w) 4 dimenziós homogén koordináta-rendszerben polinomokkal approximálunk egy 4D térgörbét, melynek kontrollpontjai 4D-ben

$$\begin{pmatrix} w_0 \cdot x_0 \\ w_0 \cdot y_0 \\ w_0 \cdot z_0 \\ w_0 \end{pmatrix} \cdots \begin{pmatrix} w_n \cdot x_n \\ w_n \cdot y_n \\ w_n \cdot z_n \\ w_n \end{pmatrix}$$

Ezt követően az approximáló görbét vetítsük centrálisan az origóból a w=1 háromdimenziós térbe (ez a 4 dimenziós tér úgynevezett 3D-s hipersíkja), így kapjuk az $\underline{R}_n(t)$ 3D-s térgörbét.

Ha az $S_{i,n}(t)$ súlyfüggvények megegyeznek a $B_i^n(t)$ Bézier-súlyfüggvényekkel, akkor $R_n(t)$-t racionális Bézier-görbének nevezzük, ha pedig $S_{i,n}(t)$ súlyfüggvények az $N_{i,n}(t)$ B-spline bázisfüggvényekkel azonosak, akkor $R_n(t)$-t racionális B-spline görbének nevezzük.

Látható, hogy a racionális Bézier és B-spline görbék a Bézier és B-spline görbék általánosításai, ugyanis ha w_i = konstans minden i-re, akkor a racionális görbék a Bézier és a B-spline görbékbe mennek át.

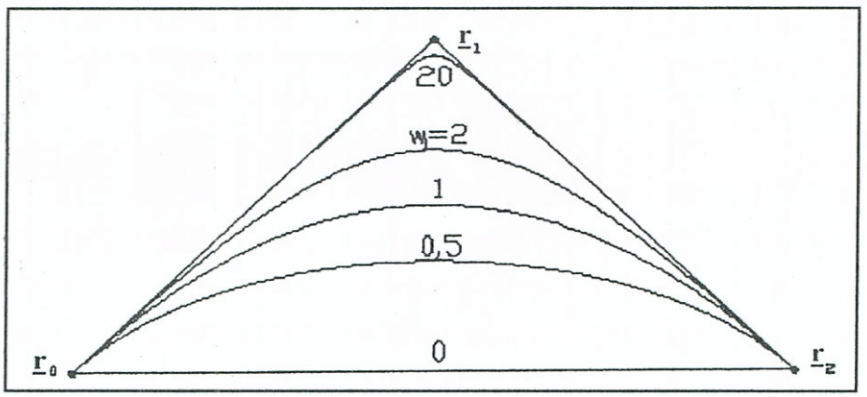
A racionális B-spline-ok közül a legfontosabbak a nem uniform (azaz nem egyenközű $[t_i, t_{i+1}]$ *felosztással kapott) racionális B-spline görbék, melyeket NURBS-nek szoktak rövidíteni. NURBS = Non Uniform Rational B-Spline.*

A NURBS görbék öröklik a B-spline görbék összes előnyös tulajdonságait. Tekintettel arra, hogy a NURBS alapú modellezés jelenleg a leghatékonyabb a számítógépes grafikában, ezeket – részben ismételve a korábbiakat – e fejezetben is összefoglaljuk:

- *A NURBS invariáns az affin transzformációkra és a vetítésekre* (projektív transzformációkra). Ez azt jelenti, hogy például a

NURBS görbe centrális vetítésekor elegendő a Boor-pontokat homogén koordinátákban transzformálni, és ezekből kiszámítani a vetületi görbét.

- Egy kontrollpont megváltoztatása csak a megfelelő paraméter tartományhoz tartozó pontokban módosítja a görbét, azaz a *NURBS lokálisan változtatható.*
- *A NURB-s görbe a kontrollpoligon „belsejében"* (konvex burkában) *helyezkedik el.*
- Ha a kontrollpontok egy egyenesen helyezkednek el, akkor a NURBS egyenesszakasz.
- *A NURBS Variation Diminishing tulajdonságú.*
- *Ha az egyik $\underline{r}_i$ kontrollpont w_i súlyát megnöveljük, a görbe határértékben az $\underline{r}_i$ ponthoz tart* (lásd 86. sz. ábra).



86. sz. ábra

Ha az $\underline{r}_1$ pont w_i súlyát növeljük, a görbe a $\overline{\underline{r}_0\underline{r}_1}$ és $\overline{\underline{r}_1\underline{r}_2}$ szakaszokhoz közeledik

Ez utóbbi tulajdonság arra is rávilágít, hogy a w_i súlyok alkalmazása újabb rugalmas alakmódosítási lehetőséget jelent a korábban tárgyalt nem racionális görbékhez képest. Így például egy w_i súly növelésével az $\underline{r}_i$ ponthoz tudjuk „húzni" a görbét.

A racionális Bézier-görbékre matematikailag könnyen tudjuk általánosítani a de Casteljau-algoritmust, a NURBS-okra pedig a Cox-de Boor-féle algoritmust is.

Az általánosítás lényege, hogy az algoritmust felírjuk 4D-ben homogén koordinátákban, majd centrálisan az origóból levetítjük a $w = 1$ hipersíkra.

3.6. TÉRBELI FELÜLETEK MODELLEZÉSE

A térbeli felületeket általában a felhasználóhoz közel álló geometriai jellemzőkkel határozzuk meg a számítógépes grafikában. Ilyenek lehetnek pontok, görbék, érintők, érintősíkok stb. A görbék modellezésénél megismertek többsége értelemszerűen alkalmazható felületek esetében is. Így például

- approximálhatunk vagy interpolálhatunk felületeket,
- összetett alakzatokat modellezhetünk egy felülettel, vagy felületfoltokkal,
- a felületmodell lehet globálisan vagy lokálisan változtatható.

3.6.1. Vonalfelületek, Coons-foltok, felületgenerálás görbe mozgatással

Ha egy felület bármely pontján át húzható olyan egyenes, melynek pontjai a felülethez tartoznak, akkor ezt a felületet vonalfelületnek nevezzük.

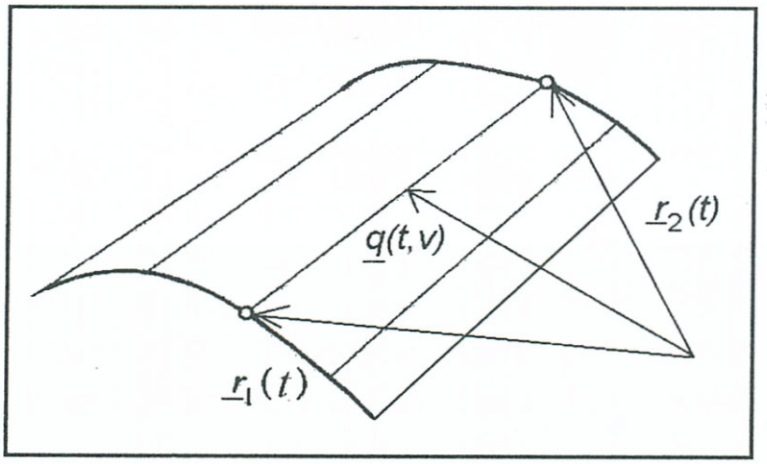
A vonalfelületek közül a számítógépes grafikában legtöbbször azokat alkalmazzuk, melyek két térgörbe pontjait kötik össze egyenes szakaszokkal. Ez matematikailag a következő formában fogalmazható meg.

Ha $\underline{r}_1(t)$ és $\underline{r}_2(t)$ két térgörbe paraméteres egyenlete, akkor a két görbét összekötő egyenesszakaszhoz vezető $\underline{q}(t, v)$ helyvektorokra:

$$\underline{q}(t,v) = (1-v)\underline{r}_1(t) + v \cdot \underline{r}_2(t)$$

ahol $0 \le v \le 1$.

Egy ilyen felületre mutat példát a 87. sz. ábra



87. sz. ábra
Két görbe által generált vonalfelület

A Coons-foltok (Coons patches) generálásakor két egymást metsző görbepár által meghatározott görbe oldalú térbeli „négyszögre" illesztünk egy interpoláló felületet.

Legyen tehát adva egy $q(u,v)$ felület térbeli négyszög részét meghatározó [0,1]x[0,1] paramétertartomány és a $\underline{q}(u,0)$, $\underline{q}(u,1)$ $u \in [0,1]$, és $\underline{q}(0,v)$ és $\underline{q}(1,v)$ $v \in [0,1]$ görbék mint a felület oldalgörbéi.

Ekkor képezünk két felületet:

$$\underline{Q}_1(u,v) = (1-v)\cdot \underline{q}(u,0) + v \cdot \underline{q}(u,1)$$

$$\underline{Q}_2(u,v) = (1-u)\cdot \underline{q}(0,v) + u \cdot \underline{q}(1,v)$$

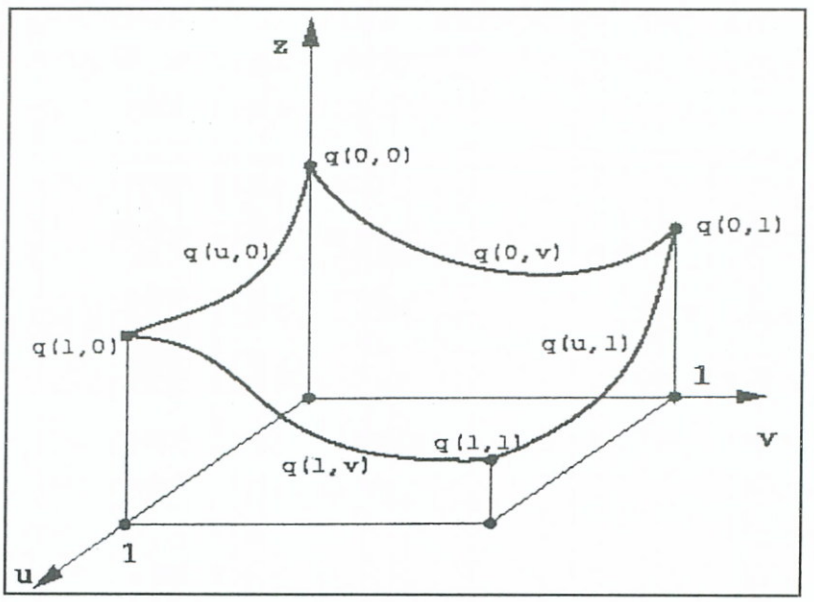
mely a szemközti oldalgörbéket interpolálja.

Az ebből a

$$\underline{Q}(u,v) = \underline{Q}_1(u,v) + \underline{Q}_2(u,v) - ((1-u),u)\begin{pmatrix} \underline{q}(0,0) & \underline{q}(0,1) \\ \underline{q}(1,0) & \underline{q}(1,1) \end{pmatrix}\begin{pmatrix} 1-v \\ v \end{pmatrix}$$

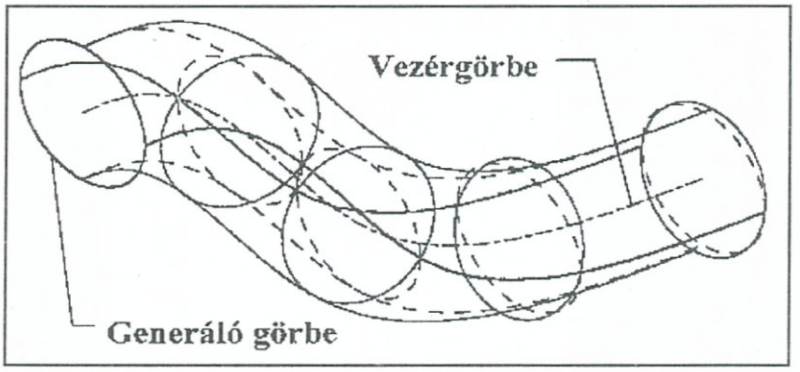
összefüggéssel generált felületet Coons-foltnak nevezzük. Matematikailag könnyen levezethető, hogy $Q(u,v)$ mindegyik oldalgörbét interpolálja.

A Coons-foltra mutat példát a 88. sz. ábra



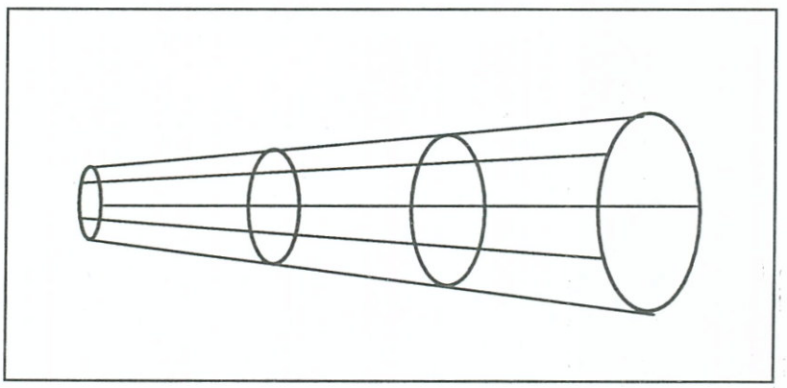
88. sz. ábra
Coons-felületfolt

Felületeket létrehozhatunk úgy is, hogy egy adott generáló térgörbét önmagával párhuzamosan mozgatunk egy vezérgörbe mentén (lásd 89. sz. ábra).



89. sz. ábra
Felület létrehozása görbe mozgatásával

Ez a módszer általánosítható oly módon, hogy a generáló görbe alakját mozgás közben változtatjuk. Egy változó sugarú körrel generált felületre mutat példát a 90. sz. ábra.



90. sz. ábra
Változó sugarú körrel generált felület

3.6.2. Bézier, B-spline és NURBS felületek

A Bézier és B-spline felületeket mozgó és alakjukat változtató Bézier és B-spline görbékből származtathatjuk.

Legyenek $\underline{a}_i$ (i=0,1,...n) egy Bézier vagy B-spline görbe kontrollpontjai, melyet mozgatni fogunk. A felületet úgy fogjuk származtatni, hogy az $\underline{a}_i$ kontrollpontokat szintén Bézier vagy B-spline görbén mozgatjuk el. Az $\underline{a}_i$ pályáját meghatározó görbe kontrollpontjai legyenek a $\underline{b}_{ij}$ (j= 0,1...m) pontok.

Ezzel teljes egészében meghatároztuk azt a felületet, melyet az $\underline{a}_i$ kontrollpontokkal rendelkező görbe a $\underline{b}_{ij}$ kontrollpontoknak megfelelő görbék mentén mozgatva a térben létrehoz.

! *Bézier-felületnek nevezzük azokat a felületeket, melyek Bézier-görbék mozgatásával úgy jönnek létre, hogy a mozgatott görbe kontrollpontjai szintén Bézier-görbéken mozognak.*

! *B-spline felületnek nevezzük azokat a felületeket, melyek B-spline görbék mozgatásával úgy jönnek létre, hogy a mozgatott görbe kontrollpontjai szintén B-spline görbéken mozognak.*

A származtatást matematikailag a Bézier-felületek példáján mutatjuk be. Legyen

$$\underline{a}(u) = \sum_{i=0}^{n} \underline{a}_i \cdot B_{in}(u)$$

a mozgatott Bézier-görbe. Feltételeink szerint az $\underline{a}_i$ kontrollpont az

$$\underline{a}_i(v) = \sum_{j=0}^{m} \underline{b}_{ij} \cdot B_{jm}(v)$$

görbén mozog. Ezt figyelembe véve a létrejött felület egy $\underline{q}(u,v)$ pontjára

$$\underline{q}(u,v) = \sum_{i=0}^{n} \left(\sum_{j=0}^{m} \underline{b}_{ij} B_{jm}(v) \right) B_{in}(u) = \sum_{i=0}^{n} \sum_{j=0}^{m} \underline{b}_{ij} \cdot B_{in}(u) \cdot B_{jm}(v)$$

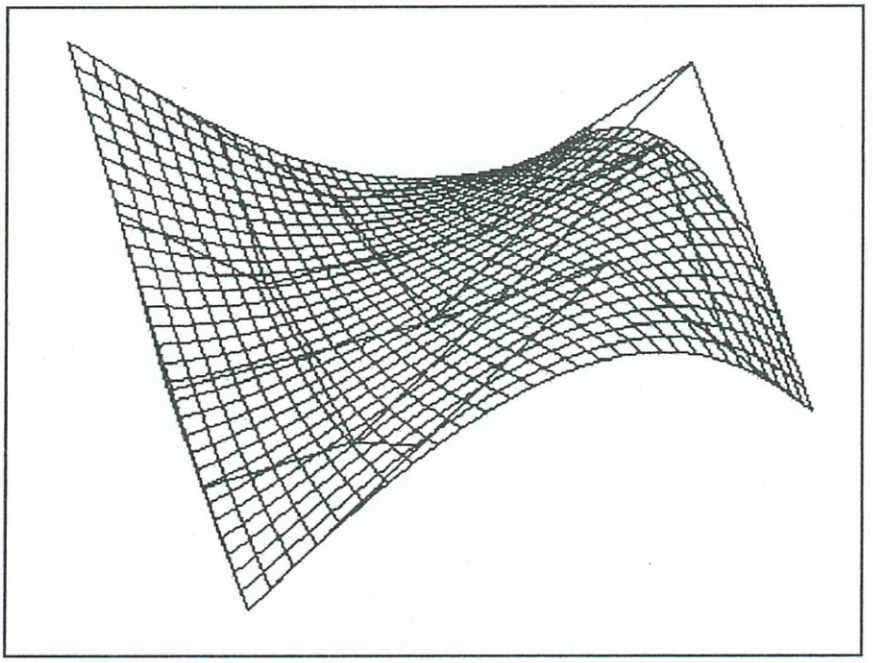
Nyilvánvaló, ha a B_{ij} függvények helyett a B-spline alapfüggvényeket alkalmazzuk a levezetésben, minden nehézség nélkül megkapjuk a B-spline felület

$$\underline{s}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \underline{b}_{ij} \cdot N_{in}(u) \cdot N_{jm}(v)$$

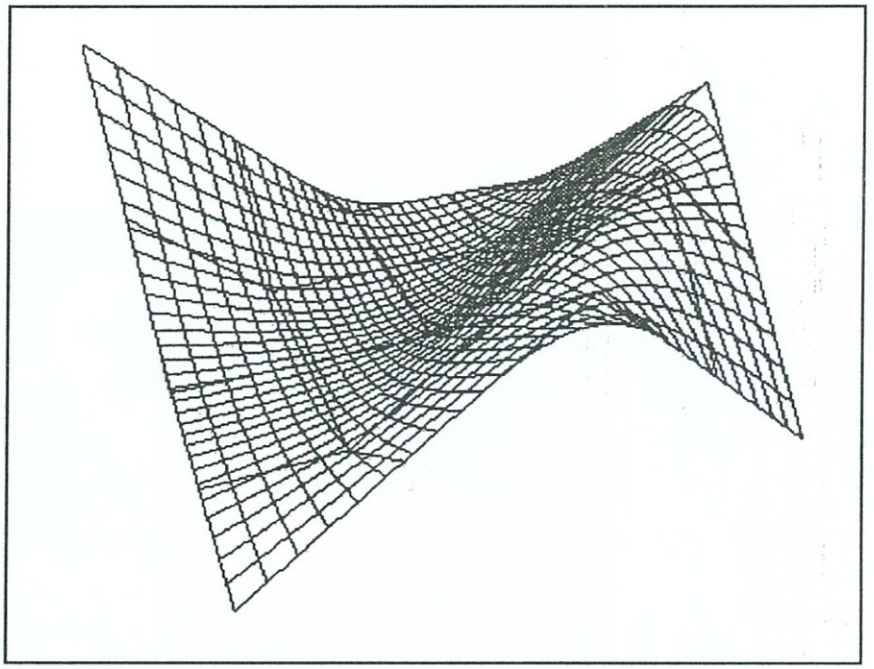
egyenletét.

Megjegyezzük, hogy az előzőekben tárgyalt felületeket tenzori szorzattal előállított felületeknek nevezik.

Ezzel a módszerrel generált felületekre mutatnak példát a 91. sz. és 92. sz. ábrák.

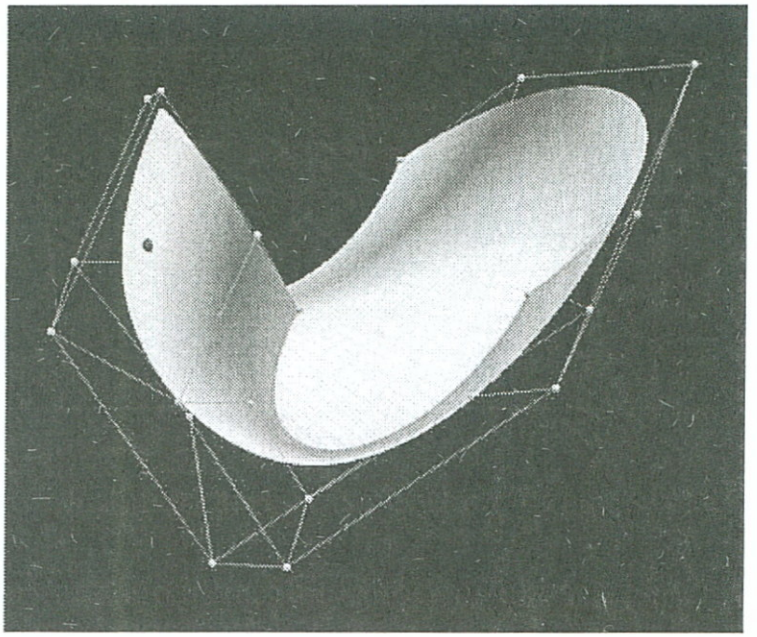


91. sz. ábra
Bézier-felület



92. sz. ábra
B-spline felület

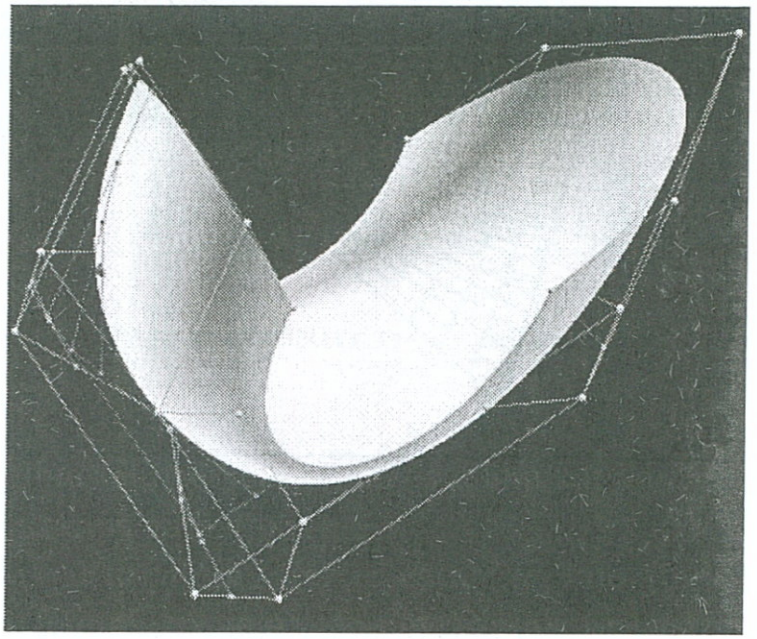
A $\underline{b}_{ij}$ pontokat a felület kontrollpontjainak nevezzük, az általuk meghatározott hálót pedig kontrollhálónak. Erre mutat példát a 93. sz. ábra.



93. sz. ábra
Felület kontrollhálója

Az előbbiekben definiált felületek (felületfoltok) a generáló görbék tulajdonságait (például a transzformációkra való invariancia) átöröklik.

Viszonylag egyszerű módon lehetőség van a de Casteljau és a Cox-de Boor iterációs algoritmusoknak a felületekre vonatkozó általánosítására is (lásd 94. sz. ábra).



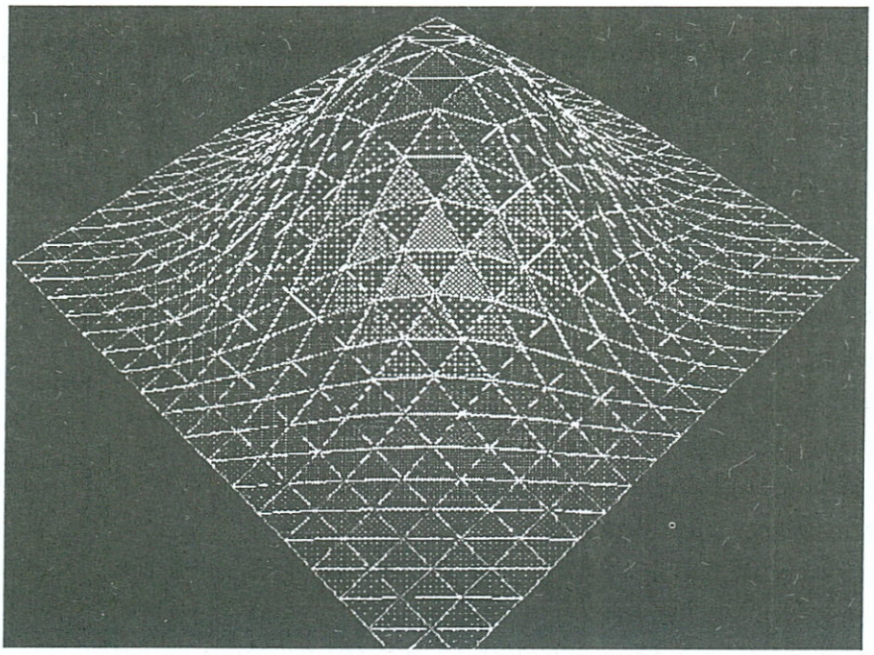
94. sz. ábra
Cox-de Boor-algoritmussal generált felület

A NURBS felületeket teljesen analóg módon származtatjuk a Bézier és B-spline felületekkel. Ekkor egy racionális, nem uniform B-spline görbét mozgatunk úgy, hogy kontrollpontjai szintén NURBS görbéken mozogjanak.

!

A NURBS felületek szintén átöröklik a NURBS görbék jellemzőit. Előnyös tulajdonságuk, hogy a w_{ij} súlyok megválasztásával viszonylag nagy rugalmassággal módosíthatjuk a NURBS felületek alakját.

Megjegyezzük, hogy mindegyik felülettípus közelítő algoritmusainak van olyan változata is, mely lehetővé teszi a felület háromszöglapokkal való közelítését. Ez napjainkban egyre nagyobb szerephez jut, mivel a megjelenítést végző kártyák chipkészlete alapvetően a háromszögekkel modellezett felületek megjelenítésére van felkészítve. Egy háromszögekkel közelített felületre mutat példát a 95. sz. ábra.



95. sz. ábra
Felület közelítése háromszögekkel

? ELLENŐRZŐ KÉRDÉSEK

83. *Hogyan írjuk le matematikailag a térbeli görbéket?*
84. Hogyan kapjuk meg a térbeli görbék érintővektorát?
85. *Mit értünk affin paramétertranszformáció alatt?*
86. *Milyen egyenlettel modellezhetők a térbeli felületek?*
87. Mit jelent a felületek szintvonalas ábrázolása?
88. *Hogyan képezhetjük a felület normálvektorát a felület egy pontjában?*
89. Mi a szerepük a poligonoknak és poliédereknek a görbék és felületek modellezésében?
90. *Mivel közelítjük a görbéket és felületeket a megjelenítéshez?*
91. Hogyan közelíthetjük a térgörbéket egyenesszakaszokkal?
92. Milyen problémákhoz vezet a térgörbéknek és a felületeknek a paramétertartomány felosztásával való modellezése?
93. Milyen felhasználói igényeket kell megoldani a görbék és felületek modellezésével?
94. *Hogyan lehet megoldani a modellezési algoritmusok egységes kezelését?*
95. *Milyen illeszkedési feladatot kell megoldani a görbék és felületek modellezésével?*
96. *Hogyan biztosítható a modellezés során felhasznált függvények egyszerű kiszámíthatósága?*
97. *Sorolja fel, milyen követelményeket kell kielégíteni a vektorgrafikus modellezésnek a felhasználóval való interaktív kapcsolattartás miatt!*
98. *Mit jelent a lokális változtathatóság?*
99. Mit jelent az interpoláció?
100. Mit jelent az approximáció?
101. Milyen függvénycsaládot célszerű választani az interpolációhoz és az approximációhoz és miért?
102. *Milyen fokszámú polinomokat célszerű választani a modellezéshez és miért?*
103. *Mivel közelítjük a térgörbéket és a felületeket a számítógépes grafikában?*
104. *Milyen két eltérő módszer létezik az interpolációs és approximációs feladat megoldására?*
105. Milyen hátránya van az egyetlen görbével vagy felülettel történő modellezésnek?
106. Mit jelent a spline fogalma és honnan származik?
107. *Mit értünk spline görbék alatt?*
108. *Mit értünk spline felületek alatt?*
109. Mit jelent a C_0 matematikai folytonosság?
110. Mit jelent a C_1 matematikai folytonosság?
111. Mit jelent a C_2 matematikai folytonosság?
112. Mit jelent a geometriai folytonosság?

113. Mikor mondhatjuk egy kontrollpontokkal generált görbére, hogy invariáns az affin transzformációkra?
114. Hogyan képezzük a Hermite-íveket?
115. Mi okoz gondot a Hermite-ívekkel való modellezéskor?
116. *Definiálja a Bézier-íveket!*
117. Ábrázolja a harmadfokú Bernstein-polinomokat!
118. *Milyen tulajdonságai vannak a Bézier-íveknek?*
119. *Mit lehet megoldani a de Casteljau-algoritmussal?*
120. Mit jelent szemléletesen a de Casteljau-algoritmus?
121. Milyen típusú spline görbék vannak?
122. *Milyen részekből épülnek fel a B-spline görbék?*
123. Mi jellemzi a $N_{j,1}(t)$ bázis B-spline függvényt?
124. Mi jellemzi a $N_{j,2}(t)$ bázis B-spline függvényt?
125. Mi jellemzi a $N_{j,3}(t)$ bázis B-spline függvényt?
126. Milyen jellemző tulajdonságai vannak a B-spline bázisfüggvényeknek?
127. *Definiálja a B-spline görbéket!*
128. *Mi a B-spline görbe súlyfüggvénye és Boor-poligonja?*
129. Mit nevezünk a B-spline görbe rendjének?
130. *Miben különböznek a B-spline és a Bézier-görbék?*
131. Miben különbözik a B-spline és Bézier-görbék alakja, ha kontrollpontjaik azonosak?
132. *Mi a Cox-de Boor-algoritmus lényege?*
133. *Miért vezették be a racionális görbéket a számítógépes grafikában?*
134. Mi az előnye a racionális görbék alkalmazásának?
135. Miért előnyösek a racionális görbék a képi megjelenítéskor?
136. Mit nevezünk racionális Bézier görbének?
137. Mit nevezünk racionális B-spline-nak?
138. *Minek a rövidítése a NURBS és mit jelent?*
139. *Melyek a NURBS legfontosabb tulajdonságai?*
140. Milyen rekurzív algoritmust használunk NURBS-ok kezelésére?
141. Hogyan definiálja a felhasználó a térbeli felületeket?
142. *Mit nevezünk vonalfelületnek?*
143. Mit nevezünk Coons-foltnak?
144. *Hogyan hozhatunk létre felületeket görbék mozgatásával?*
145. *Határozza meg a Bézier-felületeket!*
146. *Határozza meg a B-spline felületeket!*
147. Mit nevezünk kontrollhálónak?
148. *Definiálja a NURBS felületeket!*
149. Milyen tulajdonságuk van a NURBS felületeknek?

IV.

AZ EMBERI LÁTÁS ÉS A FÉNYTERJEDÉS TÖRVÉNYSZERŰSÉGEI, SZÍNTEREK

A különböző, számítógéppel előállított képek minőségének (például felbontás, színárnyalatok száma stb.) megtervezésekor a legfontosabb fiziológiai és fizikai törvényszerűségeket is figyelembe kell vennünk. Az ezzel kapcsolatos ismereteket foglaljuk össze ebben a fejezetben.

4.1. AZ EMBERI LÁTÁS

Az ember számára a fény az elektromágneses sugárzásnak az a speciális része, melyet szemével érzékelni képes és melynek hatására agyában képérzet alakul ki.

Először a fény érzékelésével kapcsolatos legfontosabb biológiai jellemzőket tekintjük át.

4.1.1. Az emberi látás biológiai alapjai, színérzékelés

Az ember szemében kb. 126 millió fényérzékelő receptor található, melyek az elektromágneses sugárzást felfogva a keletkezett ingerületet az idegrendszer útján (a szemben kb. 1 millió idegszál található) az agyba továbbítják. !

A retinában elhelyezkedő receptorok úgynevezett csapok (kb. 6 millió) és pálcikák (kb. 120 millió) lehetnek. Ezek közül a pálcikák (rodes) a fényerősséget vagy világosságot érzékelik, a csapok (cones) viszont az ember színlátásában játszanak fontos szerepet.

Az ember a fényt a 380 nm-es ultraibolya és a 780 nm-es infravörös hullámhossz tartományban képes érzékelni.

Ezen belül a színeket a szemben található

- *P* típusú (vörös fényre érzékeny)
- *D* típusú (zöld fényre érzékeny),
- *T* típusú (kék fényre érzékeny)

színérzékelő csapok különböző erősségű ingerlése alapján látjuk.

Thomas Young angol orvos fedezte fel a XIX. században, hogy az emberi szemben háromféle színérzékelő receptor található. Ezt követően Hermann Helmholz német fizikus mutatta ki, hogy a színérzékelő csapok nem csupán egy konkrét hullámhosszú fényt érzékelnek, hanem ez viszonylag széles spektrumban történik. Ezen alapul a napjainkban is használatos úgynevezett trikromatikus látáselmélet.

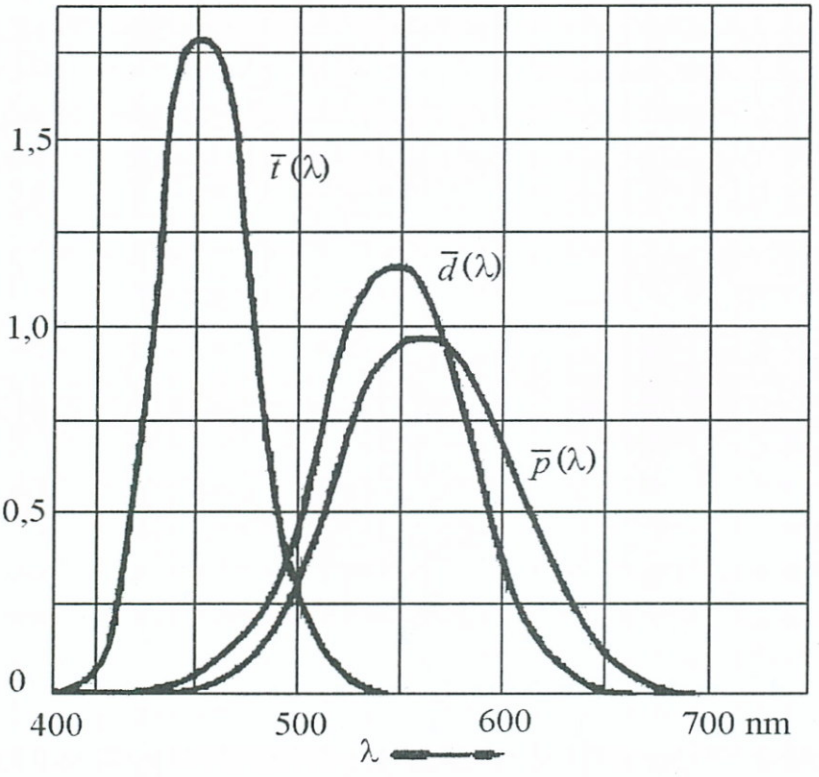
A polikromatikus színelméletet Granit Nobel-díjas amerikai orvos fejlesztette ki az 1960-as években. E szerint léteznek csapok színezeti érzékenység nélkül, ezeket dominátoroknak nevezzük. A csapok másik csoportja a modulátorok, ezek hozzák létre a színérzékelést. Ezekből hétféle különböztethető meg:

- narancs és vörös,
- sárga, zöld és zöldeskék,
- kék és ibolya,

melyek a megfelelő színeket érzékelik. Később ezt az elméletet kiegészítették azzal, hogy a modulátorok működése a fényintenzitástól is függ: így például közepes fényintenzitásnál csak a narancs, a zöld és a kékesibolya modulátor működik. Ez magyarázza, hogy a trikromatikus színelméletből levont következtetések gyakorlati tapasztalatainkkal megegyeznek.

Színérzékelésünket a *P*, *D*, *T* csapok ingerületi állapota egyértelműen meghatározza. Ez azt jelenti, hogy ha egy konkrét színárnyalatot látunk, akkor ehhez a *P*, *D*, *T* csapok egy-egy konkrét ingerületi állapota tartozik. Megfordítva, ha a *P*, *D*, *T* csapok egy konkrét ingerületi állapotban vannak, akkor ennek együttesét (eredőjét) egy konkrét színárnyalatként érzékeljük.

A csapok ingerlésének szintje a hullámhossztól függ, a különböző hullámhosszúságú fény különböző mértékű ingerületet vált ki. Ezt a $p(\lambda)$, $d(\lambda)$ és $t(\lambda)$ függvényekkel fejezhetjük ki (lásd 96. sz. ábra).



96. sz. ábra
A *p*, *d*, *t* csapok ingerszintje a hullámhossz függvényében

Látható, hogy a

- *P* csapokat kb. az 510–610 nm,
- *D* csapokat kb. az 500–590 nm,
- *T* csapokat kb. a 420–480 nm

hullámhosszú fény ingerli jelentősebben. Az ingerület a *T* csapnál kb. 440 nm (kék), a *D* csapnál kb. 540 nm (zöld) és a *P* csapoknál 580 (vörös) hullámhosszokon a legerősebb.

Egy konkrét fény különböző hullámhosszú részekből tevődik össze. Ennek színe attól függ, hogy a *P*, *D* és *T* csapokat ingerlő hullámhosszok milyen arányban találhatók meg a fényben.

Ha a szemünkbe érkező fény energiájának spektrális eloszlását egy $K(\lambda)$ függvény írja le, akkor az érzékelt színt az

$$SZ = P(K(\lambda)) \cdot P + D(K(\lambda)) \cdot D + T(K(\lambda)) \cdot T$$

egyenlet határozza meg.

Itt

$$P(K(\lambda)) = \int_{\lambda_{min}}^{\lambda_{max}} K(\lambda) \cdot p(\lambda) \cdot d\lambda$$

$$D(K(\lambda)) = \int_{\lambda_{min}}^{\lambda_{max}} K(\lambda) \cdot d(\lambda) \cdot d\lambda$$

$$T(K(\lambda)) = \int_{\lambda_{min}}^{\lambda_{max}} K(\lambda) \cdot t(\lambda) \cdot d\lambda$$

ahol λ min (kb. 380 nm) a legkisebb, a λ max (kb. 780 nm) pedig a legnagyobb az ember által még látható fény hullámhosszát jelöli.

Az emberi szem kb. 200 színárnyalat eltérését képes megkülönböztetni. Ez nem független a hullámhossztól, szemünk a legnagyobb érzékenységet az 555 nm környékén (zöld szín közelében) mutatja és ez jelentősen csökken, ahogy a látható színtartomány szélei felé haladunk.

A lényegesen különbözőnek érzékelt színek hullámhosszainak különbsége a színspektrum szélein (400 és 700 nm) kb. 10 nm, a 480 és 580 nm közötti tartományban pedig kb. 2 nm. A 340–400 nm és a 700–780 nm közötti tartományokban gyakorlatilag csak fényérzékelésünk van színérzet nélkül.

Amennyiben a teljes látható spektrumban egyenletes energiával sugároz egy fényforrás, akkor a *P*, *D*, *T* csapok ingerületi állapota azonos lesz. Ezt a fényt akromatikus fénynek nevezzük, melynek színérzete fehér.

Ahhoz, hogy a színeket megfelelően érzékelhessük, elegendő mennyiségű fényre is szüksége van a szemnek. Ezért látásunkban a fényerősségnek vagy világosságnak – melyet a szemünkbe érkező fényenergia mennyisége határoz meg – is fontos szerepe van.

Az emberi szem a nagyon kis teljesítményű, 10^{-6} lumen alatti fényt nem érzékeli, a 10^{4} lumen érték feletti teljesítményű fény pedig elvakítja a szemet. A világosságban szemünk kb. 50 fokozatot tud megkülönböztetni.

A látásunk során érzékelt színes képeket a színárnyalatok és a fényerősség mellett a színtelítettség is befolyásolja. A színtelítettség a szín fehér színnel való „felhígítottságának", fátyolosságának mértéke. A monokromatikus színek nem tartalmaznak fehér összetevőt, ezért ezeket 100%-os telítettségű színeknek is nevezzük. Ha például a vörös szín telítettségét csökkentjük (azaz növeljük a fehér fény színarányát), ez fokozatosan „halványul" és átmegy rózsaszínbe. Szemünk egy konkrét színezeten belül kb. 20 telítettségi fokozatot tud megkülönböztetni.

Összefoglalva az ember színlátásában a következő összetevők játszanak szerepet:

- *a színárnyalat vagy színezet (hue) (ezt a hétköznapokban csak színnek szoktuk nevezni), ami a szemünkbe jutó fény hullámhosszától függ,*
- *a színtelítettség (saturation), ami az érzékelt fényben megtalálható, fehér fény százalékos összetevőjétől függ,*
- *és a fényerősség (brightness), amit a szemünkbe érkező fényenergia mennyisége határoz meg.*

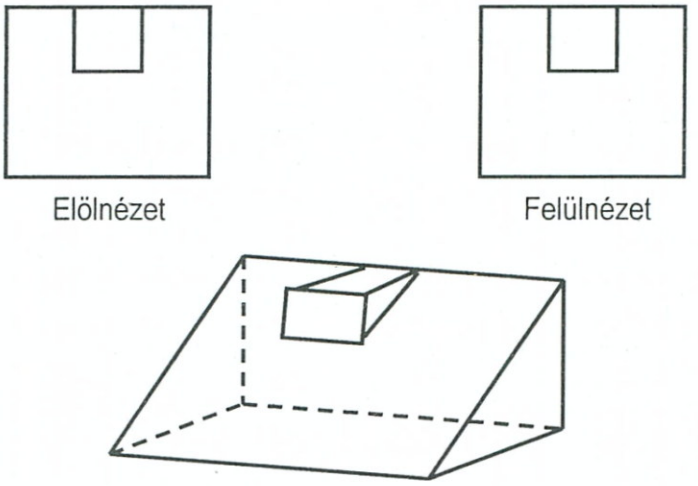
4.1.2. Kontúrlátás, térlátás és mozgóképérzékelés

Látásunk finomsága, részletgazdagsága szempontjából fontos szempont, hogy egy képen két képrészletet mikor látunk még különállónak. Nyilván ez a színárnyalatok megkülönböztethetőségétől is függ (például ha a képrészletek eltérő színűek). Meghatározó azonban a szem felbontóképességének határa, ami kísérletek szerint 0,4 mm. Ezek szerint ahhoz, hogy két pontot szemünk egy képen el tudjon különíteni, ezek között minimum 0,4 mm távolságnak kell lennie. Ennek a határértéknek megkülönböztetett jelentősége van a számítógépes grafikában például akkor, ha azt vizsgáljuk, milyen felbontóképességű monitort használjunk egy grafikus munkahelyen, vagy milyen felbontásban nyomtassunk ki egy képet.

Vizuális emlékezetünk pszichológiai vizsgálatok szerint jórészt a kontúrlátáson alapul, azaz egy alakzat, egy tárgy felismerésében döntő szerepet ennek alakja, határoló vonalainak formája és elhelyezkedése játszik. Valószínűleg ezen alapul az ember alakfelismerő képessége, mellyel képesek vagyunk a képből annak lényegét megtestesítő információkat kiemelni. Gondoljunk csak arra, hogy egy politikus arcát karikatúrája alapján is felismerjük. Ez annak ellenére is megtörténik, hogy a karikatúra és az arc fényképe mint képpontokból álló kép, alapvető eltéréseket mutat.

Kontúrlátásunk és a hozzá kapcsolódó vizuális emlékezetünk esetenként térbeli képzelőerőnk pszichikai gátja is lehet.

Ezt mutatja például a következő egyetemi ábrázoló-geometriai zárthelyi feladat is.

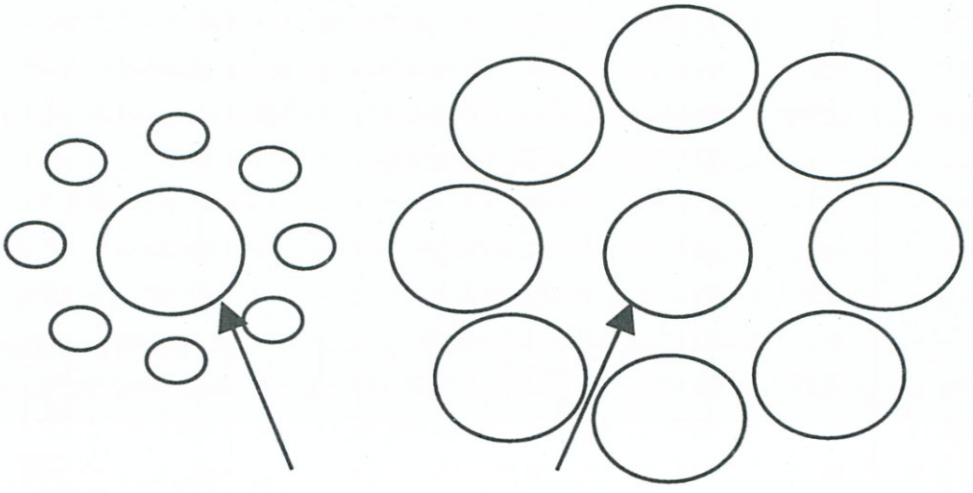


97. sz. ábra
Egyetemi zárthelyi feladat példája térbeli képzelőerőnk pszichológiai korlátaira

A feladat az ábrán nézetekben látható test oldalnézetének meghatározása volt. Ezt a feladatot a hallgatók döntő többsége nem tudta megoldani, mivel vizuális emlékezetük „rabjaiként" csak különböző kockákból álló testekben voltak képesek gondolkodni. A megoldás – mint a 96. sz. ábrán látható – két félbevágott kocka összeillesztéséből következett volna.

Színlátásunk és formaérzékelésünk nem független a tárgyak közelében található más tárgyak képétől, ami miatt szemünk néha „becsap" bennünket.

A látás környezetfüggőségének tehát az a lényege, hogy a különböző alakzatok érzékelése nem független képi környezetüktől. Erre mutat példát a 98. sz. ábra.



98. sz. ábra
Példa a látás környezetfüggőségére

Az ábrán látható, körökkel körbevett két kör azonos nagyságú. Ennek ellenére úgy látjuk, hogy a nagyobb sugarú körökkel körbefogott kör sugara kisebb a másiknál.

Ha különböző távolságokban lévő tárgyakat nézünk, ehhez szemünk a szemlencse fókusztávolságának módosításával automatikusan alkalmazkodik. A szemlencse autofókusza biztosítja tehát, hogy az eltérő távolságú alakzatokról képződő kép a retinán mindig éles legyen. A pupilla nyílásának automatikus változtatása pedig a szemünkbe jutó megfelelő fénymennyiséget biztosítja.

A 3D-s számítógépes grafikák megjelenítése szempontjából lényeges, hogy az ember térlátásával kapcsolatos ismereteket is röviden összefoglaljuk.

Egy kép térhatásúnak tűnik számunkra, ha perspektivikus torzulásokat tartalmaz (például a távolabbi tárgyakat arányosan kisebbnek látjuk), ha különböző tárgyak takarják egymást, árnyékokat látunk, a legtávolabbi tárgyak elmosódottak stb.

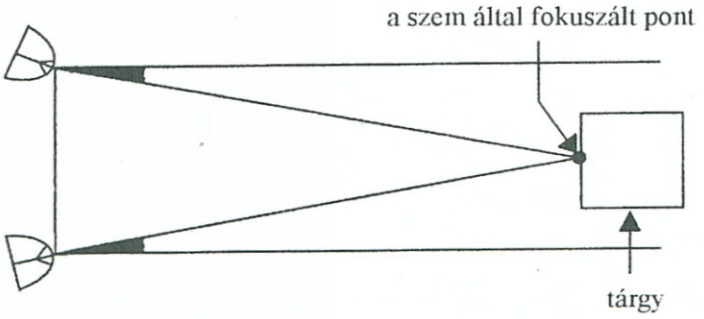
A térbeliség érzékelésének ez a fajtája nyilván vizuális emlékezetünkkel kapcsolatos, korábbi élményeink, tapasztalataink is hozzájárulnak, hogy az így szerkesztett képet térhatásúnak ítéljük meg.

Az ember térlátása azonban nemcsak ilyen hatásokon alapszik.

Leonardo da Vinci írja: „Lehetetlenség, hogy a festmény, még ha a legtökéletesebben utánozza is vonalában, árnyékában, fényében, színében a tárgyat, ugyanolyan plasztikusnak tűnjék, mint a természeti tárgy, kivéve, ha ezt a tárgyat nagy távolságból nézzük, s fél szemmel."

Az előző idézet is kifejezi, hogy a takarás, árnyékolás, perspektivikus torzulás – bár bizonyos mértékig alkalmas plasztikus illúziók keltésére – önmagában még kevés a valódi térhatás kialakulásához. Arra gondolhatnánk, hogy a távolságérzékelés a szemlencse görbületének az éleslátáshoz szükséges automatikus beállásával (akkomodáció) van kapcsolatban. Egy még a XIX. század első felében Ch. Wheatstone által konstruált tükrös sztereoszkóp ezt azonban teljesen megcáfolta. Az eljárás nagyon egyszerű volt: tükrök segítségével a szem előtt a sztereoszkóp olyan képet állított elő, mely megfelelt egy valódi tárgyról a szemben keletkezett képnek. Optikai szempontból azonban mindkét kép gyakorlatilag azonos távolságra volt a szemtől, az összhatás mégis a térbeliség érzete volt.

Még Keplertől származik az a hipotézis, hogy a térélmény kialakulásának alapja a két szem közti távolság miatt a két szemgolyó optikai tengelye között fellépő szögeltérés (lásd 99. sz. ábra).



99. sz. ábra
A két szemmel történő látáskor a szemtengely jobbra és balra eltér a merőlegestől

! *Ezt a legújabb vizsgálatok csak részben igazolták. Ezek szerint a sztereo élmény kialakulásának az a lényege, hogy a két szem retináján a képek részletei egymáshoz viszonyítva kissé eltolódnak és ezt a különbséget az agy az idegrendszer útján észleli.* Ez segített többek között a számítógépes grafikában a 2D-s térhatású képek, az úgynevezett sztereogramok megkonstruálásához (lásd még 5.2.2. fejezet).

Sztereo párosítási probléma (stereo matching problem) alatt a szakirodalom azt érti, hogy a valódi térbeli látvány, vagy ezt szimuláló sztereoszkópos vetítés esetén hogyan találja meg az idegrendszer a bal, illetve jobb retinára vetülő képeken az egymásnak megfelelő, azonos térbeli pontból származó képpontokat. A térbeli pont mélysége (nézőtől való távolsága) ennek alapján meghatározható. E probléma megoldását D. Marr és T. Poggio (1979) a látás számítógépes modelljében adta meg. Ez összhangban van azokkal a kísérleti tényekkel, melyek a Julesz Béla (1971) -féle random sztereogramokkal (Random Dot Stereogram, RDS) kapcsolatosak.

Az elmúlt években létrejöttek olyan számítógépes modellek is, melyek mind a természetes, mind a mesterséges képek esetében jól működnek, és a szemek konvergens mozgását is figyelembe veszik. Az ezekből levont következtetések összhangban vannak a legújabb humán kísérletek tapasztalataival is.

! *Az emberi látásnak az is fontos tényezője, hogy a szemünkben található receptorokat egy kép tudatos észleléséhez mennyi ideig kell fényingernek érnie. Több kísérlet bebizonyította, hogy az 1/15-öd másodpercnél rövidebb időre „felvillanó" képeket nem érzékeljük.* Így például, ha egy filmbe másodpercenként bevágunk 15 darab 1/15 másodpercig látható képet, ezt a film nézése mellett nem vesszük észre.

Érdemes megjegyezni, hogy a tudatalatti képérzékelésünkhöz szükséges időtartam az 1/15 másodpercnél kisebb. Ezt a tulajdonságunkat a reklámban (például a 60-as években az USA-ban) fel is használták, amíg ezt a hatóságok be nem tiltották. Az olyan filmrészletek alá, melyek a nézőben pozitív érzelmeket váltottak ki, bevágták egyes termékek reklámképeit, melyek 1/15 másodpercre villantak fel. Ezzel próbálták elérni, hogy a nézőben az adott termékkel kapcsolatban pozitív képzettársítás alakuljon ki.

A számítógépes grafikával készített filmek és animációk szempontjából meghatározó jelentőségű, hogy másodpercenként hány képet kell mutatni az embernek a mozgófilmhatás kiváltása érdekében.

A filmvetítésnél és a televíziós kép sugárzásánál másodpercenként 25 teljes állóképet jelenítenek meg, ez már elegendő ahhoz, hogy az ember összefüggő, folyamatos, viszonylag villogásmentes mozgóképet érzékeljen. (Megjegyezzük, hogy váltottsoros vagy interlaced televíziós képmegjelenítés esetében ez azt jelenti, hogy másodpercenként 50 „félképet" kell sugározni.)

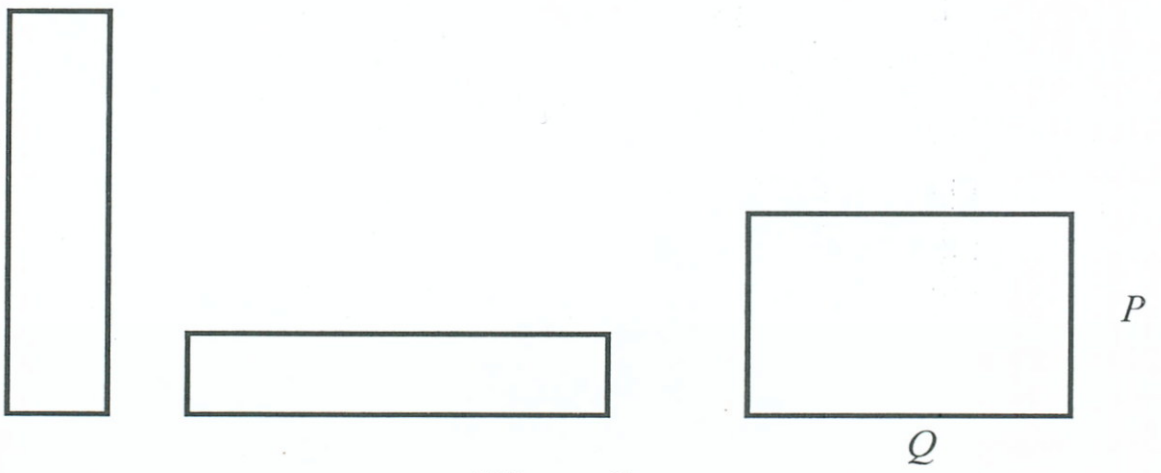
!

4.1.3. Az emberi képérzékelés esztétikája

A számítógépes grafikában a képek megtervezése során célszerű esztétikai és litográfiai szabályokat is figyelembe venni, ha azt akarjuk, hogy a kép „szép" legyen és pozitív hatást váltson ki a nézőből. Természetesen egy számítógépes grafika könyv e szakterület még vázlatos bemutatását sem vállalhatja fel. Ezért csak néhány példát mutatunk be, melyek felhívják a figyelmet az esztétika és litográfia szerepére a számítógépes grafikában. Az e téren részletesebb ismereteket megszerezni kívánó olvasók számára például „David Ogilvy: A reklámról" című könyvét ajánlhatjuk.

Az esztétikai törvényszerűségek bemutatásához példaként az aranymetszést választottuk, mely fontos szerepet tölt be a képzőművészetben is.

A 100. sz. ábrán három téglalapot látunk. Tegyük fel a kérdést, hogy melyik a legszebb téglalap? Vagy kérdezzük meg, hogy egy A4-es lap oldalainak arányát hogyan határozták meg?



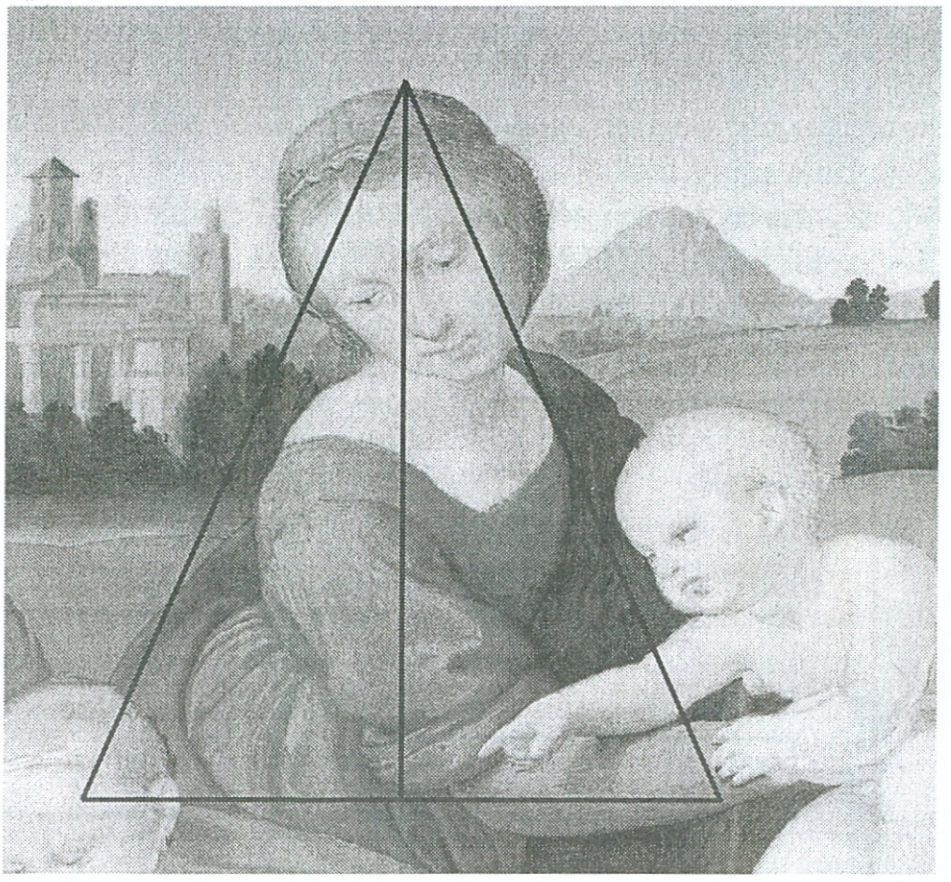
100. sz. ábra
Különböző téglalapok közül a „legszebb" kiválasztása

Az emberek többsége a jobboldali téglalapra szavaz, mely oldalainak az aránya $P/Q = Q/(P+Q)$ $P<Q$. Másképp megfogalmazva, a téglalap rövidebb oldala úgy aránylik a hosszabb oldalhoz, mint a hosszabb a két oldal összegéhez.

Ez az aranymetszés aránya, számértéke $\frac{1}{2}\left(\sqrt{5}-1\right)$.

Ha az előbbi egyenlőség jobb oldalán lévő tört számlálóját és nevezőjét egyaránt elosztjuk Q-val és a $\frac{P}{Q}$ arányt x-szel jelöljük, akkor az $x = \frac{1}{x+1}$ egyenlőségét kapjuk. Ezt a másodfokú egyenletet az x ismeretlenre megoldva adódik, hogy az aranymetszés aránya $\frac{1}{2}\left(\sqrt{5}-1\right)$.

Az aranymetszés arányát a képzőművészetben – például a madonna képek formájánál – is felismerhetjük (lásd 101. sz. ábra).



101. sz. ábra

Itt a kép formai szerkesztését meghatározó egyenlőszárú háromszög magasságának és alapjának aránya pontosan $\frac{1}{2}\left(\sqrt{5}-1\right)$.

Az esztétikai és litográfiai törvényszerűségek szerepének további bemutatásához a példákat a reklámgrafikából választottuk. Nem szabad például a reklámban általában eltérni a képek összeillő, a természetben megszokott színeitől. Ellenérzést szül például, ha egy képen egy emberi kezet vizuális tapasztalatainkkal ellentétesen szürkére színezünk.

A reklámgrafika néhány litográfiai szabálya:

- A reklám címének nagyon jól olvashatónak kell lennie. Negatív hatást érünk el, ha a reklám címében talpas betűket vagy erősen döntött betűket alkalmazunk.
- Nem célszerű a reklám teljes szövegét nagy betűkkel írni, mivel az emberek a hosszabb szöveg olvasásánál a kis betűkhöz szoktak hozzá.
- Érdekes az a statisztikailag bizonyított törvényszerűség, hogy a reklám olvasottságát növeli, ha a reklám szövegét a szövegszerkesztővel nem rendezzük.

4.2. A FÉNYTERJEDÉS FIZIKAI ALAPJAI

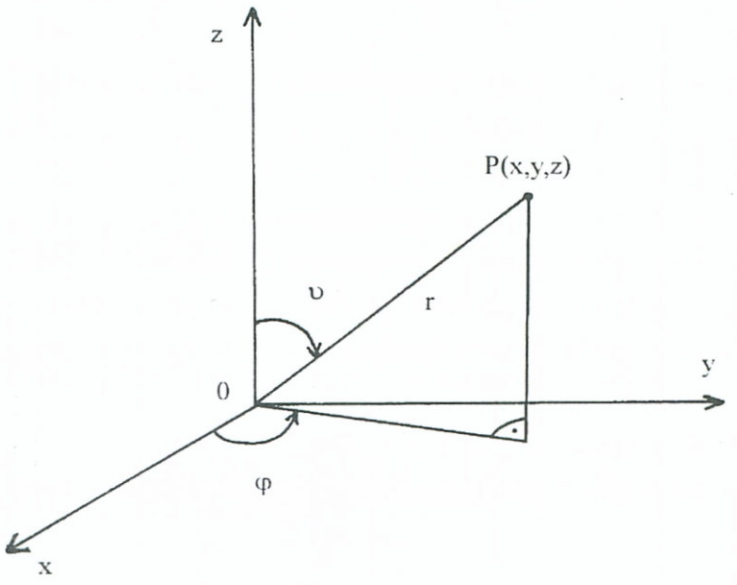
E fejezetben főként a számítógépes grafika megvilágítási modelljeihez szükséges legfontosabb fizikai alapfogalmakat és törvényeket tekintjük át. Ennek során nem törekszünk a tárgyalt témakörök matematikailag precíz kifejtésére, inkább mondanivalónk szemléletességére fektetjük a hangsúlyt.

A fogalmak és törvények képletekben való megfogalmazásakor a differenciális jelölésmódot fogjuk használni, azaz egy f függvény „végtelen kis" (infinitézimális) megváltozását a fizikában szokásos módon df-fel jelöljük. Reményeink szerint az olvasó a matematikából ismeri a differenciál fogalmát. Amennyiben mégsem, a df-fel jelölt mennyiségeket tekintse az adott érték egy „nagyon kicsi" megváltozásának, mely közelítőleg egy számértéknek (konstansnak) tekinthető.

A fejezetben szereplő mennyiségeket egyrészt a sugárzásfizika, másrészt vizuális fotometria szempontjából tárgyaljuk. A két nézőpont között az a lényeges különbség, hogy míg az előbbi az elektromágneses sugárzást a teljes spektrumra vonatkozóan tárgyalja, addig az utóbbi csak az ember által látható tartományban vizsgálja a fényt és figyelembe veszi az emberi látás sajátosságait is. A két tárgyalásmód fogalmai a humán korrekciós tényе-

zőkkel egymásnak megfeleltethetők. A sugárzásfizikai fogalmakat az „*e*" index-szel (energia), a vizuális fotometriai mennyiségeket pedig „*v*" index-szel (vizuális) fogjuk jelölni.

A tárgyaláshoz egyes esetekben szükségünk lesz a térbeli polárkoordináta-rendszerre (lásd 102. sz. ábra).



102. sz. ábra
Térbeli polárkoordináta-rendszer

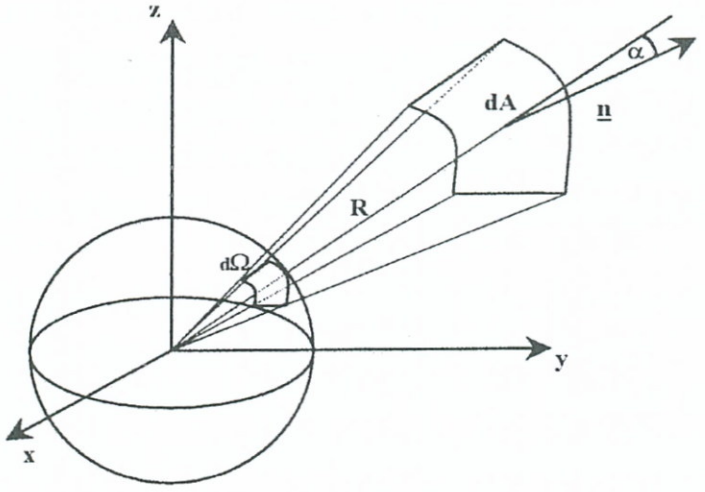
A polárkoordináta-rendszerben egy térbeli *P* pontot az origótól mért *r* távolsággal, az *OP* egyenesnek *xy* síkra való vetülete *x*-tengellyel bezárt *φ* szögével és az *OP* egyenesnek a *z*-tengellyel bezárt *u* szögével írunk le. A szokásos derékszögű *x*, *y*, *z* koordinátákkal a kapcsolatot az

$$x = r\cos\varphi\cdot\sin u \qquad y = r.\sin\varphi\cdot\sin u \qquad z = r\cdot\cos u$$

egyenletek írják le.

4.2.1. Fényforrások sugárzási energiája és teljesítménye, besugárzás-erősség és a Lambert-féle négyzetes távolság- és koszinusztörvény, sugársűrűség

Először bevezetjük a Ω térszög fogalmát. Legyen *dA* egy felületelem és legyen adva a térbeli koordináta-rendszerben egy origó középpontú egységsugarú gömb. A gömb felszínének azt a részét, melyből origó középpontú vetítéssel *dA*-t megkapjuk, jelölje *dΩ* (lásd 103. sz. ábra).



103. sz. ábra

Ekkor $d\Omega$-t differenciális térszögnek nevezzük. Erre

$$d\Omega = \frac{\cos\alpha}{R^2} dA$$

ahol R a dA felületelem távolsága az origótól, α pedig a dA felületelemhez vezető helyvektor és dA felületi normálisa által bezárt szög.

Egyszerű geometriai megfontolásokból következik előző összefüggésünk (hasonló és vetített síkidomok területe).

Ebből az Ω térszöget egy véges felületre az

$$\Omega = \int_A \frac{\cos\alpha}{R^2} dA$$

felületi integrálból kaphatjuk meg.

Sugárzási teljesítmény

Vizsgáljunk meg most egy, az origóban elhelyezett pontszerű fényforrást, mely Q_e sugárzási energiával (Radiant Energy) rendelkezik. Ekkor a $\varphi_e = \frac{dQ_e}{dt}$ mennyiséget, azaz a fényforrás által időegység alatt kisugárzott energiát sugárzási teljesítménynek nevezzük, egysége a watt.

Kisugárzás erősség

Ha a pontszerű fényforrás az egységsugarú gömb egységnyi térszögébe időegység alatt I_e energiát sugároz ki, akkor I_e-t a kisugárzás erősségnek nevezzük (radiant intensity).

A definícióból következőn

$$I_e = \frac{d\varphi_e}{d\Omega}$$

Besugárzáserősség, Lambert-törvényei

A természetes fényforrások nem minden irányban sugározzák be azonos erősséggel a felületet, a felületre eső sugárzáserősség a távolságtól is függ.

Ezt fejezi ki a Lambert-féle négyzetes távolság és koszinusz törvény

$$E_e = I_e \frac{\cos\alpha}{R^2}$$

ahol E_e a felület egységére időegység alatt eső sugárzási energia, R a felület távolsága a pontszerű fényforrástól, α a felület normálvektorának és a fényforrástól a felületig vezető vektor (sugárzás iránya) által bezárt szög. Az E_e mennyiséget besugárzás erősségnek (irradiance) nevezzük.

Az E_e mennyiségre vonatkozó törvényt könnyen beláthatjuk, ha figyelembe vesszük, hogy az energiamegmaradás törvénye szerint a kisugárzott és a felület által befogadott energia azonos, figyelembe véve, hogy $d\Omega = \frac{\cos\alpha}{R^2} \cdot d\mathrm{A} \cdot$

Megjegyezzük, hogy a besugárzás-erősségre vonatkozó Lambert-törvényt pontszerű fényforrásra fogalmaztuk meg, de ez a gyakorlatban kis hibával minden további nélkül kiterjeszthető nem pontszerű fényforrásokra is.

A távolság és koszinusztörvényt szétválasztva is megfogalmazhatjuk. Ekkor:

- *A távolság törvény kimondja, hogy pontszerű fényforrásnál, azonos beesési szögnél a besugárzás erőssége fordítottan arányos a felületnek a fényforrástól mért távolsága négyzetével.*
- *A koszinusztörvény szerint pedig a felület besugárzás-erőssége alfa szög irányában:*

$$I_\alpha = I^0 \cos\alpha$$

A besugárzás erőssége tehát akkor a legnagyobb, ha a fénysugár iránya azonos a felület normálvektorával, ennek értékét jelöltük I^0-al.

Ez a törvény ad magyarázatot többek között arra is, hogy miért ég le először a vállcsúcsunk vagy az orrunk a napon, illetve miért a déli órákban – amikor a nap merőlegesen süt – kell jobban tartanunk az UVA és B sugaraktól.

Most megvizsgáljuk a nem pontszerűen sugárzó testeket és ezzel kapcsolatban bevezetjük a sugársűrűség (radiance) fogalmát.

Ha egy pontszerű testünk van, akkor a megvilágított testeknek éles árnyékhatáruk van. Kiterjedt testeknél viszont azt érzékelhetjük, hogy az árnyékos terület fokozatosan megy át világosból sötétbe. Ennek az az oka, hogy felületformájú kisugárzó testeknél a besugárzás-erősség nem csak a sugárzás irányától, hanem helyétől is függ.

Vizsgáljuk meg most egy kiterjedt fényforrás dA_1 felületelemét, mely a besugárzott felülettől υ_1 irányban helyezkedik el.

Sugársűrűség (radiance)

Ez esetben a sugárzáserősséget υ_1 irányban $\cos\upsilon_1$-val csökkenteni kell. Ezért

$$dI_{e1} = L_{e1} \cdot \cos\upsilon_1 \cdot dA_1$$

Az újonnan bevezetett L_{e1} mennyiséget nevezzük sugársűrűségnek, mely az egységnyi térszögből származó besugárzás erősségét, vagyis ebből a térszögből az időegység alatt az egységnyi felületre eső sugárzási energiát jelenti.

A sugársűrűség fontos tulajdonsága, hogy az értéke a fénysugár útja során nem változik, azaz nem helyfüggő. (Természetesen ez csak vákuumra igaz.) Ez különösen a radiosity és a raytracing algoritmusok alkalmazása miatt fontos a számítógépes grafikában (lásd még 6. fejezet).

Az eddigiek során a sugárzásfizikai fogalmakat alapvetően geometriai tulajdonságok alapján jellemeztük. A teljes körű tárgyaláshoz a frekvenciától való függőséget is figyelembe kell venni. Ezt megtehetjük azzal, hogy minden sugárzásfizikai X_e mennyiséghez egy $X_e(\lambda)$ spektrális sűrűséget rendelünk.

$$X_e(\lambda) = \frac{dX_e}{d\lambda}$$

4.2.2. Vizuális fotometria

A sugárzásfizikai mennyiségeket ebben a fejezetben fotometriai mennyiségekhez rendeljük hozzá. A sugárzásfizika és a fotometria közötti legfontosabb különbség az – mint erre már utaltunk –, hogy az utóbbi az emberi látás pszichológiai tulajdonságait is figyelembe veszi. A fotometriai mennyiségeket sugárzásfizikai megfelelőjükkel fogjuk jelölni, de megkülönböztetésül egy v index-szel fogjuk ezeket ellátni.

A sugárzásfizikai és a fotometriai mennyiségek között a $V(\lambda)$ világosságérzékelési függvény teremti meg a kapcsolatot, mely megadja az emberi szem relatív érzékenységét a hullámhossz függvényében. Ha a szemünk a sötétséghez alkalmazkodik, akkor a látásérzékelésért a szemben elhelyezkedő pálcikák a felelősek, ha viszont megfelelő a megvilágítás, akkor a szem csapjai érzékelik a fényingert. Ezért a fotometriában két érzékelési függvényt kell figyelembe vennünk:

- a sötétben alkalmazkodó szem érzékenységi függvényét $V(\lambda)$,
- a világossághoz alkalmazkodó szem érzékenységi függvényét pedig $V'(\lambda)$

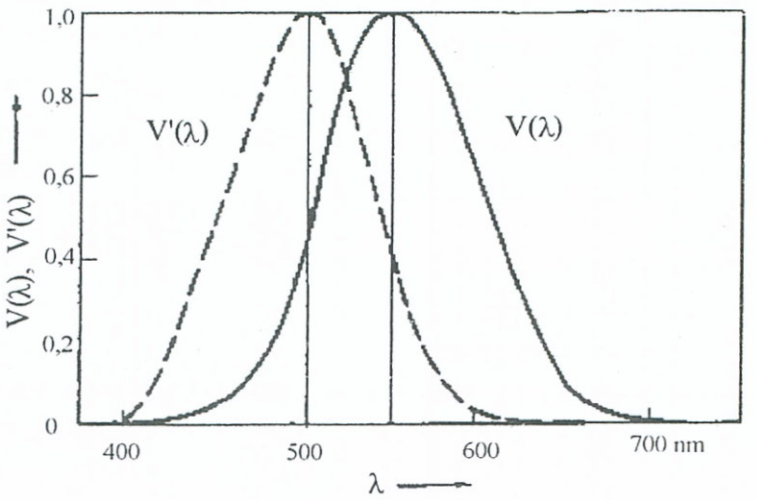
fogja jelölni.

A látásnak a fenti adaptációs képességét Purkinje-jelenségnek nevezzük. A fénysűrűséggel meghatározva:

$L_v \leq 0{,}1$ cd/m² alatt szkotopos,
$0{,}1 < L_v \leq 30$ cd/m² között mezopos,
$30 < L_v$ cd/m² között fotopos

látásról beszélhetünk. A cd = candela a fényerősség egysége.

A $V(\lambda)$ és $V'(\lambda)$ függvényeket ábrázoltuk a 104. sz. ábrán.



104. sz. ábra
Világosságérzékelési görbék

A fotometriai mennyiségeket a sugárfizikai mennyiségekhez a $V(\lambda)$ és $V'(\lambda)$ függvényekből következő korrekcióval rendeljük egymáshoz.

Általánosságban a levezetett fogalmakra ez a következőképpen végezhető el. Ha X_e jelöli a sugárfizikai mennyiséget, X_v pedig a fotometriai megfelelőjét, akkor

$$X_v = K \int_{360}^{830} X_e(\lambda) \cdot V(\lambda) d\lambda$$

és

$$X_v' = K' \int_{360}^{830} X_e(\lambda) \cdot V'(\lambda) d\lambda ,$$

ahol $X_e(\lambda)$ az adott mennyiségnek a spektrális eloszlását kifejező függvény, K és K' egy konstans. A K és K' értékét például a megvilágítás erősség egységének (candela, lásd alant) definíciójából vezethetjük le.

$$K = 683 \frac{cd}{W} \qquad K' = 1725 \frac{cd}{W}$$

Az I_v fényerősség az I_e sugárzáserősség megfelelője a fotometriában. Ennek egysége a candela (rövidítve: cd).

Definíció: *Egy candela a fényerőssége egy fényforrásnak, ha 540 THz-es (megfelel 555 nm-es hullámhossznak) monokromatikus fényt bocsát ki, melynek sugárzás erőssége 1/683 Watt• sr^{-1}.*

A besugárzás-erősségnek a fotometriai megfelelője a megvilágítás-erősség, melynek egysége a lux (rövidítve lx).

Annak érdekében, hogy megfelelő képünk alakuljon ki arról, hogy egy luxnyi megvilágítás-erősség mit jelent, néhány példát mutatunk be:

- éjszakai holdsugárzás kb. 0,2 lx,
- déli napfény kb. 70.000 lx,
- az olvasáshoz szükséges minimális megvilágítás-erősség kb. 100 lx,
- normál munkavégzéshez kb. 500 lx, precíziós munkához kb. 1000 lx szükséges.

Végezetül megadjuk a sugárzásfizikai és fotometriai mennyiségek összefoglaló táblázatát:

Sugárfizikai mennyiségek		
Megnevezés és jelölés	Egysége	Angol megnevezés
Sugárzó energia Q_e	J	radiant energy
Sugárzási teljesítmény φ_e	W	radiant power
Kisugárzáserősség I_e	Wsr^{-1}	radiant intensity
Besugárzáserősség E_e	Wm^{-2}	irradiance
Sugársűrűség L_e	$Wm^{-2}sr^{-1}$	radiance

[sr = sztereo radian]

Fotometriai mennyiségek		
Megnevezés és jelölés	Egysége	Angol megnevezés
Fénymennyiség Q_v	lumen sec.	quantity of light
Fényáram φ_v	lumen	liminous flux
Fényerősség I_v	candela	liminous intensity
Megvilágításerősség E_v	lux	illuminance
Fénysűrűség L_v	candela.m^{-2}	luminance

1. sz. táblázat
Sugárfizikai és fotometriai mennyiségek összefoglalása

4.2.3. Fénykibocsátás, fényvisszaverődés és -törés

4.2.3.1. Fényt sugárzó testek (emisszió)

A fényt kisugárzó testeket a sugársűrűséggel vagy a fénysűrűséggel jellemezzük. Általános esetben a fénysűrűség a kisugárzási irány (ezt a φ és ϑ polárszögekkel adhatjuk meg) és a λ hullámhossz függvénye.

$$L = L(\lambda, \varphi, \vartheta)$$

Az összes fénykibocsátást a hullámhosszra, illetve kiterjedt testeknél a teljes felületre való összegzéssel kapjuk meg.

Pontosabban szólva ez

$$L(\varphi, \vartheta) = \int_{\lambda=360nm}^{830nm} L(\lambda, \varphi, \vartheta) d\lambda$$

és a

$$\frac{d\varphi_v(\lambda)}{dA} = \int_{\varphi=0}^{2\pi} \int_{\vartheta=0}^{\pi/2} L(\varphi, \vartheta) \cos\vartheta \cdot \sin\varphi \cdot d\vartheta \cdot d\varphi$$

integrálok kiszámítását jelenti.

Ha a fénysűrűség irányfüggetlen, akkor a felületegységre jutó fényáramot a

$$\frac{d\varphi_v(\lambda)}{dA} = K \cdot L(\lambda)$$

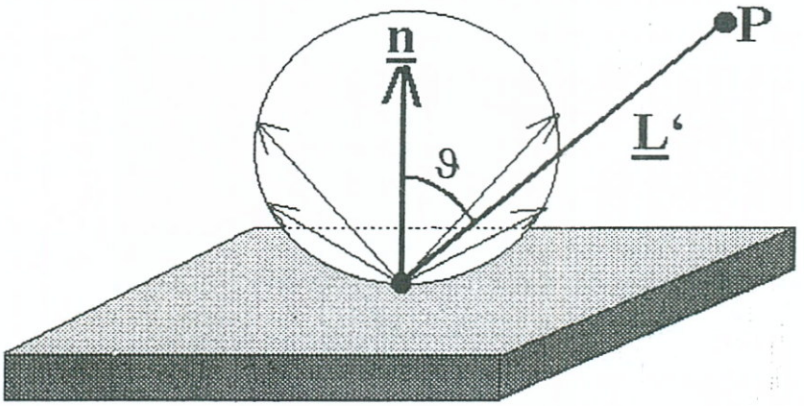
összefüggés határozza meg.

Az irányfüggetlen fénysűrűség a radiosity fotorealisztikus renderelő eljárásnak a legfontosabb alapfeltevése (lásd még 6. sz. fejezet).

Azokat a testeket, melyek ilyen módon sugároznak ki fényt, Lambert-féle fényforrásoknak nevezzük. Ezek jellemzője, hogy az ilyen testek felületét minden irányból azonos világosságúnak érzékeljük.

A fényerőssége a Lambert-féle fényforrásoknak csak a sugárzó felület normálisa és a kisugárzási irány által bezárt szög koszinuszától függ (lásd 105. sz. ábra):

$$dI = L \cdot dA_1 \cdot \cos\vartheta_1$$



105. sz. ábra
Lambert-féle fényforrások kisugárzása

A másik speciális eset az, ha a sugárzó tárgy által kibocsátott fény fénysűrűsége a térben csak a fényforrástól mért távolságtól függ. Ez legegyszerűbben az

$$f(r) = \begin{cases} \dfrac{1}{r^2} & \text{ha } r \neq 0, \\ \text{különben} \quad 1 \end{cases}$$

távolságfüggvénnyel modellezhető.

E modell lényeges javítási lehetőségére Foley mutatott rá. Ezek szerint $f(r)$-nek válasszuk a következő függvényt

$$f(r)=\begin{cases}\dfrac{1}{c_0 r^2 + c_1 r + c_2} & \text{ha} \quad r \neq 0,\\ \text{különben} \quad 1\end{cases}$$

Itt a c_0, c_1, c_2 konstansokat a felhasználó definiálhatja, figyelembe véve, hogy

$$c_0^{\,2} + c_1^{\,2} + c_2^{\,2} \neq 0$$

4.2.3.2. Fényvisszaverődés testek felületéről

A testek felületéről visszaverődő fénysugárzást legáltalánosabb formában a spektrális visszaverődési tényezővel tudjuk leírni, mely a visszavert sugárzás fénysűrűségének és a beeső sugárzás megvilágítás-erősségének az aránya.

$$\rho(\lambda,\varphi_r,\upsilon_r\varphi_i,\upsilon_i)=\frac{L(\lambda,\varphi_r,\upsilon_r)}{\mathrm{E}(\lambda,\varphi_i,\upsilon_i)}$$

A képletben az *i*-index a beeső, az *r* index pedig a visszavert sugárzáshoz tartozó mennyiségeket jelöli. Ezt a függvényt az angol szakirodalomban „bidirectional reflection distribution function"-nak nevezik és BRDF-fel rövidítik.

A számítógépes grafikában a BRDF következő fontos tulajdonságait hasznosítjuk:

- ρ nem változik, ha a beeső és a visszavert fény irányát felcseréljük. (Ezen alapul többek között a raytracing fotorealisztikus renderelő eljárás.)
- ρ általában anizotrop, azaz ha azonos belépési és kilépési irány mellett a felületet a normálisával együtt elforgatjuk, akkor a visszavert fény megváltozik. Erre tipikus példát a fényes anyagú felületek szolgáltatnak. Ilyen például a fém, a lakkfelületű műanyagok stb.
- ha a felület egy pontjába több irányból lép be fénysugár, ezek a visszaverődés során „egymást nem befolyásolják", azaz egyszerűen a fénysűrűség számításánál összegezhetők.

A visszaverődésnek általában 3 típusát szokták megkülönböztetni:

- ideálisan diffúz visszaverődés,
- ideálisan tükröző visszaverődés,
- irányított diffúz visszaverődés.

Ideálisan diffúz visszaverődés

Ez az úgynevezett matt, illetve durva felületek jellemzője. Ez esetben a visszavert fény sűrűsége független a kilépő sugár irányából.

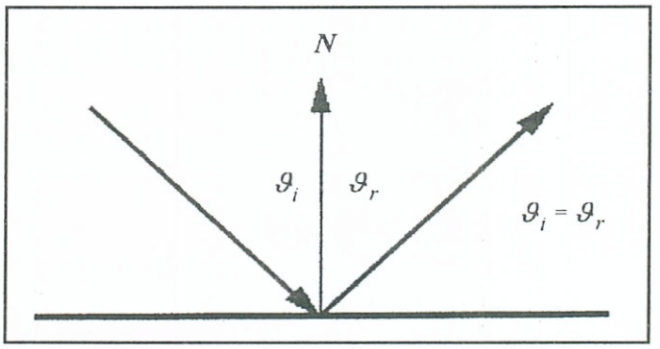
!

A diffúz visszaverődést szemléletesen úgy képzelhetjük el, hogy a felület – erős nagyítás esetén látható – mikroszkopikus tükröző felületelemekből áll, melyeknek síkjai a legkülönbözőbb szögekben állnak. Ezért bármilyen irányban is vizsgáljuk a kilépő sugarakat, mindig találunk több mikroszkopikus felületelemet, mely ebben az irányban tükröz vissza.

Ideálisan tükröző visszaverődés

Az ideálisan tükröző visszaverődést tapasztalhatjuk például egy síktükör esetében. Ebben az esetben a fénysugár a geometriai optikából jól ismert törvénynek megfelelően halad: A belépő és visszavert sugár, valamint a felületi normális egy síkban helyezkedik el és a belépő és visszavert sugár a tükröző felület normálisával azonos szöget zár be (lásd 106. sz. ábra).

!



106. sz. ábra
Ideálisan tükröző visszaverődés

Megjegyezzük, hogy a hagyományos raytracing algoritmusok (lásd 6. fejezet) ezt a visszaverődési törvényt alkalmazzák.

Irányított diffúz visszaverődés (spekulare Reflexion)

A természetben az ideális visszaverődésnek az előbbiekben tárgyalt két szélsőséges esete rendkívül ritka. Viszonylag gyakori viszont *az olyan fényvisszaverődés, melynek van egy kitüntetett iránya, mely irányban a felület a legtöbb fényt veri vissza, majd ahogy ettől az iránytól távolodunk, a visszavert fény mennyisége ennek megfelelően csökken.* Ezt az esetet irányított diffúz visszaverődésnek nevezzük.

!

Az előzőekben tárgyalt visszaverődés-típusokat, a durván tükröző esettel kiegészítve szemlélteti a 107. sz. ábra.

107. sz. ábra
A fényvisszaverődés típusai
a) ideális diffúz b) irányított diffúz c) durván tükröző
d) ideálisan tükröző

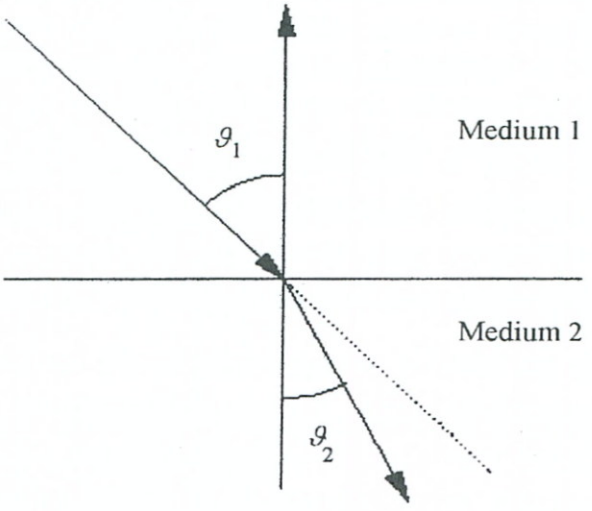
4.2.3.3. Átlátszó testek, fénytörés

A fénysugár törését, amikor egy optikailag ritkább anyagi közeg és egy optikailag sűrűbb közeg határán halad át, a Snellius–Descartes-törvénnyel írhatjuk le.

Ezek szerint a beeső sugár és a megtörő sugár egy síkban van a két közeg határfelületének normálisával. A ϑ_1 beesési és a ϑ_2 törési szögre:

$$\frac{\sin \vartheta_1}{\sin \vartheta_2} = \frac{n_2}{n_1} = \text{konstans}$$

ahol n_1 és n_2 az anyagi közegekre jellemző törésmutatók (lásd 108. sz. ábra).

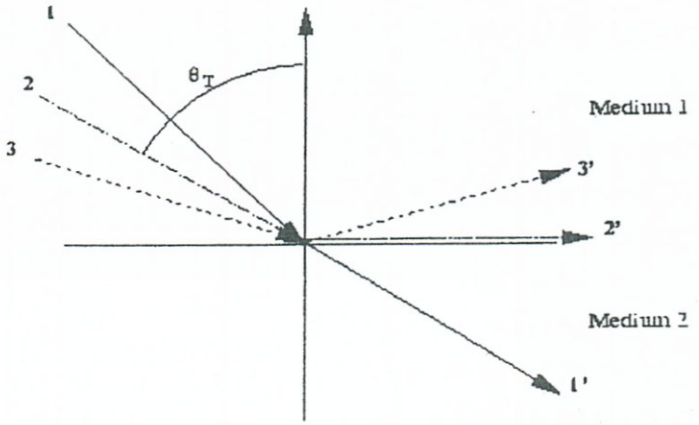


108. sz. ábra
Optikai törési törvény optikailag ritkább közegből optikailag sűrűbb közegbe belépő fénysugár esetén

A két anyagi közeg által meghatározottan létezik egy ϑ_T kritikus szögérték,

$$\sin \vartheta_T = \frac{n_2}{n_1}$$

mely esetében a törési szög 90°-os lesz. Ezt a teljes visszaverődés határszögének nevezzük. Ha ezt átlépi a beeső fénysugár, akkor a belépés az optikailag sűrűbb közegből a ritkább közegbe már nem lehetséges. Ezekben az esetekben a fény a két közeg határfelületéről visszaverődik (total reflexio). (Lásd 109. sz. ábra.)

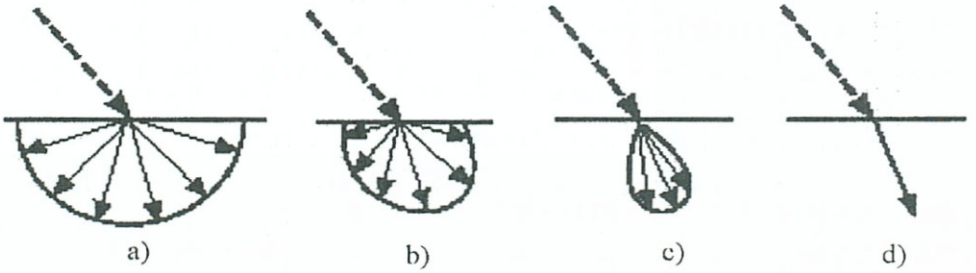


109. sz. ábra
A teljes visszaverődés esete. A 2-es sugár alatti fénysugarak a határfelületről visszaverődnek

Az átlátszó testeken történő fényáthaladásnak is megkülönböztethetjük négy esetét:

- *diffúz áthaladás,*
- *irányított diffúz áthaladás,*
- *durva törés,*
- *ideális törés.*

Ezeket az eseteket mutatja a 110. sz. ábra



110. sz. ábra
Fény áthaladása átlátszó testeken
a) diffúz áthaladás b) irányított diffúz áthaladás c) durva törés
d) ideális törés

Az átlátszó közegek fontos jellemzője a ρ visszaverő és a τ áteresztőképesség. Ezeket kifejezhetjük a törésmutatókkal:

$$\rho = \frac{(n_1 - n_2)^2}{(n_1 + n_2)^2}$$

$$\tau = \frac{4 \cdot n_1 \cdot n_2}{(n_1 + n_2)^2}$$

Példaként megemlítjük, hogy levegő és víz esetén

$$\rho = 2\% \quad \text{és} \quad \tau = 98\%$$

4.3. SZÍNTEREK

A számítógépes grafikában az is fontos célunk, hogy a különböző megvilágítási modellek és algoritmusok szerint meghatározott fénysűrűségeket valós és élethű módon megjelenítsük egy raszteres képernyőn. Ehhez a következő kérdéseket kell megválaszolni:

- Hogyan lehet a monitor egy képpontjának színét megadni?
- Milyen összefüggésben van a sugárzási spektrum a színekkel?

A kérdéseinkre a választ részben a színkeverés különböző eljárásai adják meg.

4.3.1. A színkeverés alapjai

A monitoron található képpontok három részből állnak, melyek egy-egy színösszetevőnek felelnek meg. Minden egyes képpontrészt különböző intenzitással lehet felgyújtani és ezek általában a vörös, zöld és kék színeknek

felelnek meg. Ezek az úgynevezett RGB monitorok, melyek a Reed, Green, Blue színek angol kezdőbetűjéről kapták a nevüket. A színeknek megfelelő képpontrészeket a mai monitoroknál 256 intenzitásszinten lehet felvillantani. Ezeket szemünk önállóan nem érzékeli, ezek hatása szemünkben egy egységes színingerré áll össze. Az intenzitási szintek alapján összesen 256^3 színárnyalat megjelenítésére van lehetőség, ez összesen kb. 16 millió különböző színárnyalatot jelent.

Önként adódik a kérdés: elegendő-e a 3 alapszín az összes elképzelhető színárnyalat előállításához?

Erre a választ a színkeverés Grassmann-féle törvényei adják meg:

- A keverékfény színezetét az összetevők szabják meg.
- Bármely színinger előállítható három szín additív keverésével, ezért minden színezet megadásának szükséges és elégséges feltétele három független színezeti jellemző megadása.
- A színérzet a világossággal nem változik. Ez tehát azt is jelenti, hogy a színérzetet elvonatkoztatva a világosságtól új fizikai jellemzőt nyerünk: ez a színezeti jellemző.

A második törvényből az is következik, hogy a színek egy háromdimenziós matematikai struktúrát alkotnak, azaz megfeleltethetők egy 3D-s vektortér vektorainak. Ezt a vektorteret színtérnek hívjuk, a tér egyes vektorait színvegyértéknek. A vektorok hossza megfeleltethető a fénysűrűségnek (színérték) a vektorok iránya pedig a színtípust határozza meg.

Megjegyezzük, hogy a színtér 3D-s jellege abból is következik, hogy szemünk 3 fajta színérzékelő receptorral rendelkezik: a *P*, *D*, *T* típusú csapokkal.

A színtér három alapszínét elsődleges színvegyértékeknek nevezzük, ezek felelnek meg a vektortér alapvektorainak. Mint már láttuk, a monitoroknál a vörös (*R*), zöld (*G*) és kék (*B*) színhármas egy elsődleges színvegyértékhármast alkot.

Ezek szerint egy tetszőleges színvegyérték előállítható a következő színegyenlettel:

$$F = r \cdot R + g \cdot G + b \cdot B$$

Itt r, g, b az egyes alapszínek arányát jelenti.

Ahhoz, hogy a színtér 3D-s vektortér voltát belássuk, igazolnunk kell ennek lineáris jellegét is. Így például egy *k* skalárral való szorzás esetén, ha

$F = U+V+W$	teljesül, akkor abból
$kF = kU+kV+kW$	is következik.

Ezt az összefüggést kísérletekkel igazolták.

A színtér lineáris vektortérként való kezelése különösen azért hasznos, mert a különböző elsődleges színvegyértékhármasok (azaz alapszínhármasok) közötti transzformációkat egyszerű és logikus matematikai formában, mátrixszorzással elvégezhetjük.

A színvegyértékeket úgynevezett baricentrikus (azaz súlyponti) koordinátákkal lehet jellemezni. Ezek szerint, ha

$$F = R_F \cdot R + G_F \cdot G + B_F \cdot B$$

akkor az F szín baricentrikus koordinátái

$$r_F = \frac{R_F}{R_F + G_F + B_F}$$

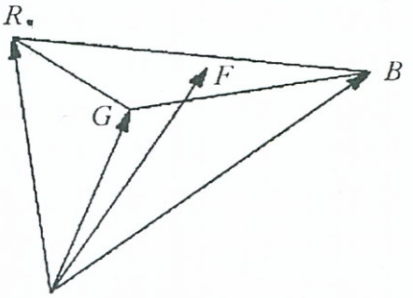
$$g_F = \frac{G_F}{R_F + G_F + B_F}$$

$$b_F = \frac{B_F}{R_F + G_F + B_F}$$

Ebből rögtön következik, hogy

$$r_F + g_F + b_F = 1$$

Ez azt jelenti, hogy minden színvegyértéket a három elsődleges színvegyérték vektor által meghatározott háromszög síkjának egy pontjával tudjuk reprezentálni (lásd 111. sz. ábra).



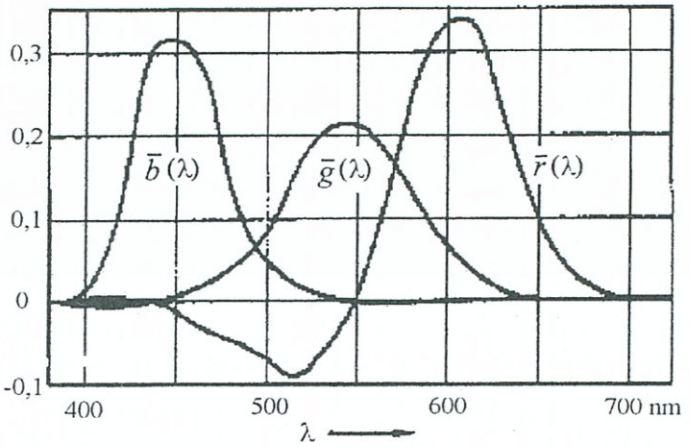
111. sz. ábra
Színvegyérték vektorok baricentrikus koordinátái

Az elkövetkezendőkben arra a kérdésre keresünk választ, hogy egy meghatározott $\Phi(\lambda)$ spektrális eloszláshoz tartozó színérzetet hogyan lehet az alapszínekből kikeverni.

Ehhez osszuk fel a látható hullámhossz spektrumsávokra, és vizsgáljuk, hogy az így keletkező λ_k ...λ_e spektrumsávok milyen színingert állítanak elő. Ekkor egy színingerhez tartozó $f_{\lambda i}$ színvegyértékre

$$f_{\lambda i} = r_{\lambda i} \cdot R + g_{\lambda i} \cdot G + b_{\lambda i} \cdot B$$

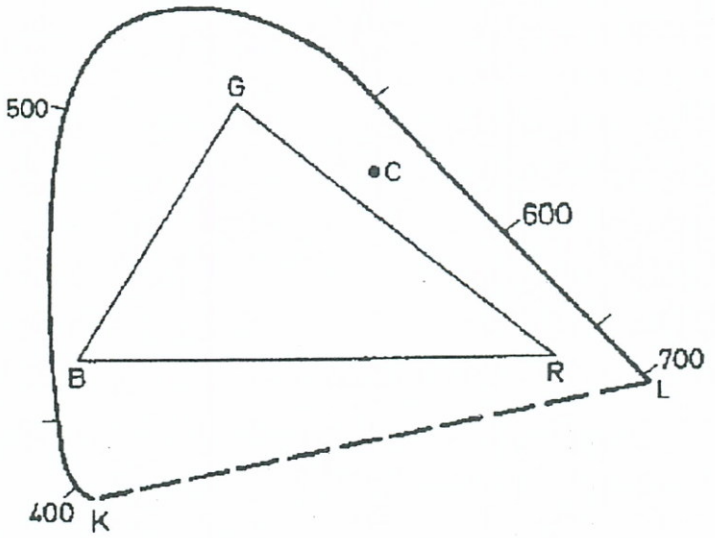
Itt az $r_{\lambda i}$, $g_{\lambda i}$ és $b_{\lambda i}$ együtthatókat a λ_i-re és az R, G, B alapszínekre vonatkozó spektrumértékeknek nevezzük. Ezeket a megfelelő mennyiségű kísérlet eredményei alapján függvényformában is előállíthatjuk (lásd 112. sz. ábra).



112. sz. ábra
Spektrumérték függvények R = 700 nm, G = 546 nm és B = 436 nm alapszínek esetében

Ha a spektrálérték görbéket egy konkrét alapszínhármasra már meghatároztuk, akkor ebből a spektrálértékgörbéket tetszőleges alapszínhármasra is kiszámíthatjuk.

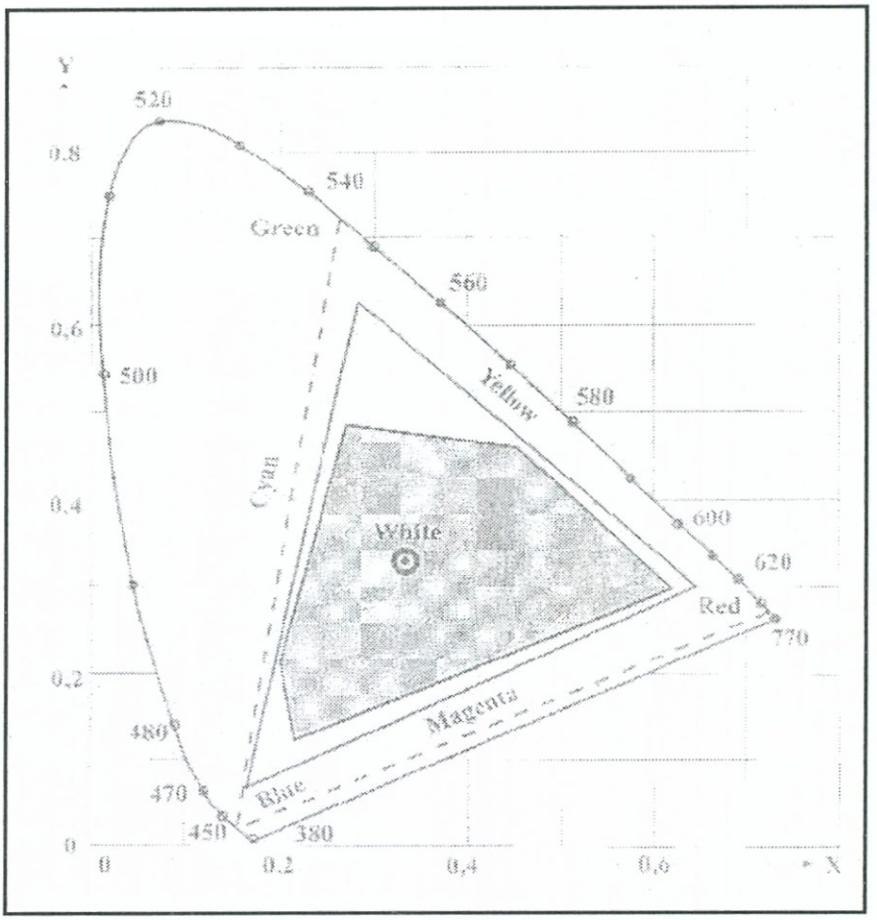
Ha a spektrális színezeteket a λ hulámhossz $f(\lambda)$ függvényének tekintjük, akkor egy folytonos görbét kapunk a 113. sz. ábrán bemutatottak szerint.



113. sz. ábra
Spektrális színezet függvény

Az összes ábrázolható szín a görbe „belsejében" található és minden ilyen szín egy frekvenciaspektrumnak felel meg. Így az összes valós alapszínhármas is a görbén belül található. Egy RGB monitor esetén az ábrázolható színek „spektruma" attól függ, hogy a monitor alapszíneinek megfelelő háromszög a görbén belül milyen területű. Annak érdekében, hogy ne legyen szükség minden egyes alapszínhármasra, új spektrálérték függvényeket kiszámolni a Commision International de l'Éclairage (CIE) az alapszínek rendszerére már 1931-ben egy szabványt javasolt.

A CIE 1931-ben elfogadott normalizált, kétdimenziós színdiagramját mutatja a 114. sz. ábra.



114. sz. ábra
A CIE által javasolt szabvány

A szaggatott vonal jelzi a szabványosított, az alapingerszínek által határolt területet. A folytonos vonal a legjobb minőségű számítógép monitorok által megjelenített kép színtartományát mutatja. A szürke terület a nyomdatechnikában használható színek választéka.

4.3.2. Az RGB, a CMY és CMYK, valamint a HSB színtér

A fénysugárzó testek elsődleges és másodlagos fényforrások lehetnek. Az első csoportba tartoznak azok, amelyek önállóan képesek fény kibocsátására. Ilyenek például a nap, az izzólámpák vagy a számítógép monitora. A másodlagos fényforrások közé sorolhatjuk azokat a testeket, melyek önállóan nem bocsátanak ki fényt, csak visszaverik a rájuk eső fényt. Erre példaként gyakorlatilag az összes tárgyat felsorolhatnánk, de a számítógépes grafikában különösen a nyomtatóval vagy nyomdatechnikával, esetleg plotterrel előállított papíralapú adathordozók érdekesek számunkra.

A színkeverésnek két alapvető módja van, aszerint, hogy elsődleges vagy másodlagos fényforrást modellezünk.

- *Az összeadó, vagy additív színkeverésnél a vörös, zöld és kék alapszínekből vett meghatározott mennyiségeket adunk össze és így kapjuk a különböző színárnyalatokat. Ezzel az úgynevezett elsődleges fényforrások színeit tudjuk előállítani.* !
- *A szubsztraktív vagy kivonó színkeverésnél az alapszínek komplementereiből (ciánkék, bíborvörös, sárga) állítjuk elő a színeket. Ezzel lehet modellezni a különböző tárgyak által visszavert fényt.* (Kivonó (szubsztraktív) színkeverés például színszűrőkkel hozható létre. Ha egy lámpa elé piros, illetve sárga színszűrőt helyezünk, piros, illetve sárga fényt látunk. Ha mindkét színszűrőt feltesszük, a keverékszín narancs lesz, a fényintenzitás csökken.) !

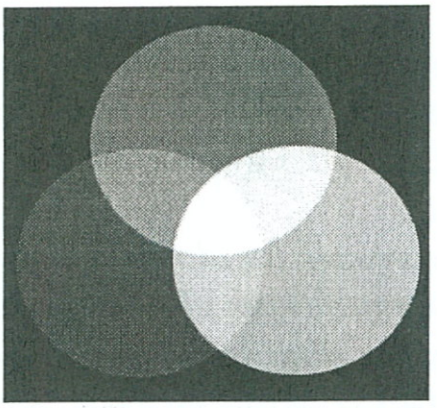
A színkeverés törvényei alapján definiálható a komplementer (kiegészítő) szín fogalma. Eszerint két szín komplementer-párt alkot, ha a keverékük akromatikus színérzékletet hoz létre (additív színkeverésnél tehát fehéret).

Komplementer párnak nevezünk két színt akkor is, ha szubsztraktív színkeverést alkalmazva feketét (sötétet) kapunk eredményül. Ha például 650 nm-es vörös és 492 nm-es zöldeskék szűrőt használunk, ez együttesen nem engedi át a fényt. Másként fogalmazva: ha egy 650 nm-es vörös felületet 492 nm-es zöldeskék szűrőn át figyelünk, a felületet feketének látjuk. Ha egy felület a színtelen (fehér) fényt teljes egészében visszaveri, akkor fehérnek látjuk. A színes, például piros felület a fehér fényből csak a piros fényt veri vissza, a többi más hullámhosszú fényt elnyeli (abszorbeálja).

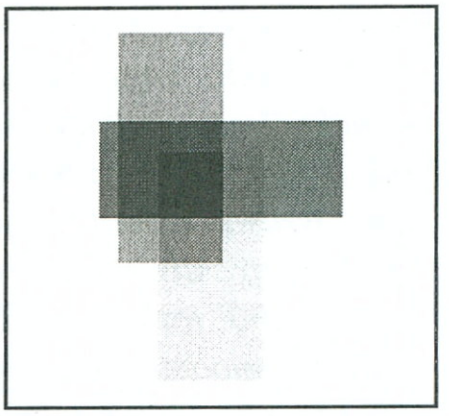
Ennek megfelelően képezhetjük a számítástechnikában legelterjedtebben használt RGB, CMY és CMYK színképeket:

- *Az RGB színtér vörös, zöld, kék (red, green, blue) alapszínekből kikeverhető színeket tartalmazza, az additív színkeverés modellezéséhez használjuk.* !
- *A CMY színtér a ciánkék, bíborvörös, sárga (cyan, magenta, yellow) alapszínekből kikeverhető színeket tartalmazza, a szubtraktív színkeverés modellezéséhez használjuk.* !
- *A CMYK színtér megegyezik a CMY színtérrel, azzal a különbséggel, hogy a CMY színtér alapszíneihez még hozzáadjuk a „tiszta" fekete színt is. Ennek az az oka, hogy a CMY alapszínek keverésével csak sötétszürke színt tudunk előállítani és a nyomdatechnikában a teljesen fekete színre is szükségünk van.* !

Az RGB és CMYK színtérre mutatnak példát a 115. és 116. sz. ábrák.



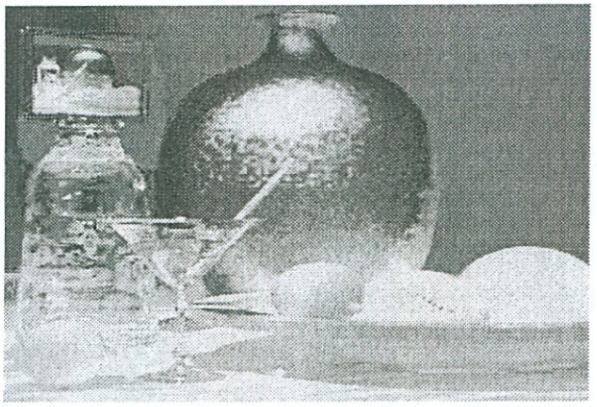
115. sz. ábra
A RGB színtér



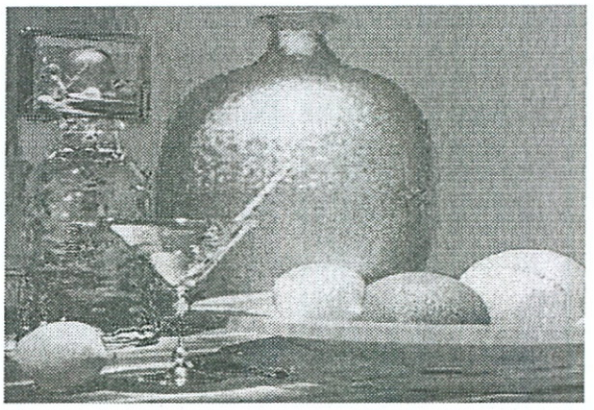
116. sz. ábra
A CMY színtér

Látható, hogy az RGB alapszínek összege a fehér szín, és két RGB alapszín keveréke a CMY színtér egy alapszínét adja ki.

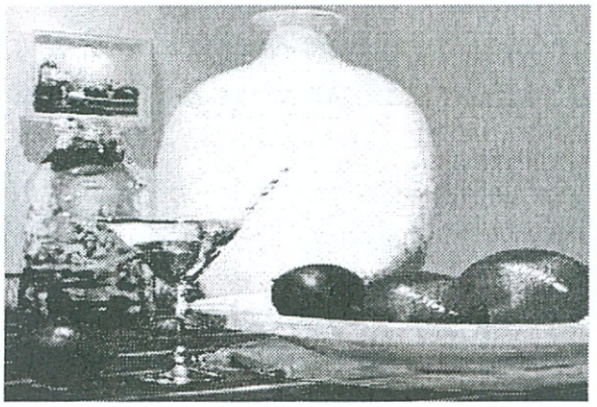
A CMYK színtér 4 fóliájára mutatnak példát a 117–121. sz. ábrák.



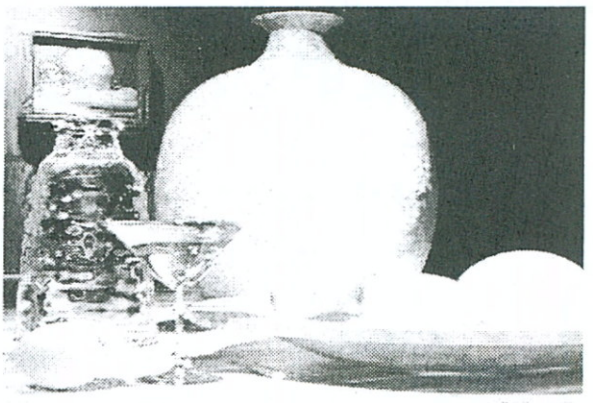
117. sz. ábra
A nyomdatechnikában használt ciánkék fólia



118. sz. ábra
A nyomdatechnikában használt bíbor fólia



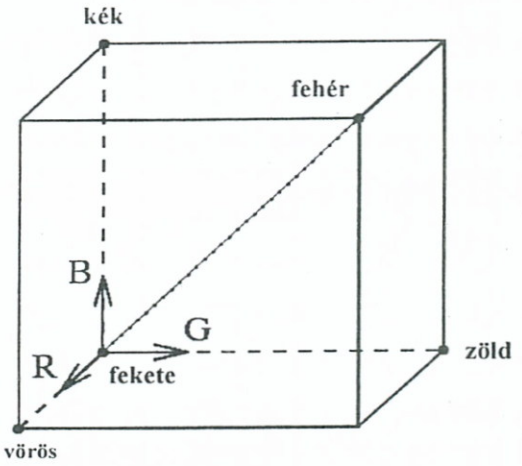
119. sz. ábra
A nyomdatechnikában használt sárga fólia



120. sz. ábra
A nyomdatechnikában használt fekete fólia

121. sz. ábra
A CMYK fóliák egyesítésével kialakuló színes kép

Az RGB és CMY színtereket egy egységkockával szokták ábrázolni (lásd 122. sz. ábra).



122. sz. ábra
Az RGB színtér szemléltetése egységkockával

Az RGB és CMY színterek közötti átszámítás kölcsönösen egyértelmű:

$$[C, M, Y] = [1, 1, 1] - [R, G, B]$$
$$[R, G, B] = [1, 1, 1] - [C, M, Y]$$

!

Az RGB és CMYK színterek színei között viszont nem lehetséges kölcsönösen egyértelmű megfeleltetés. Ezt könnyen beláthatjuk a következő gondolatmenet alapján:

Az RGB alapszínek intenzitását a számítástechnikában 0, ..., 255 közötti értékkel adjuk meg, a nyomdatechnikában pedig a CMYK alapszíneket általában 0–100 közötti fedettségi értékekkel jellemezzük.

! *Emiatt vannak olyan RGB színek, melyek a CMYK alapszínek keverésével nem nyomtathatók ki* (ez leginkább a kék szín környékére jellemző).

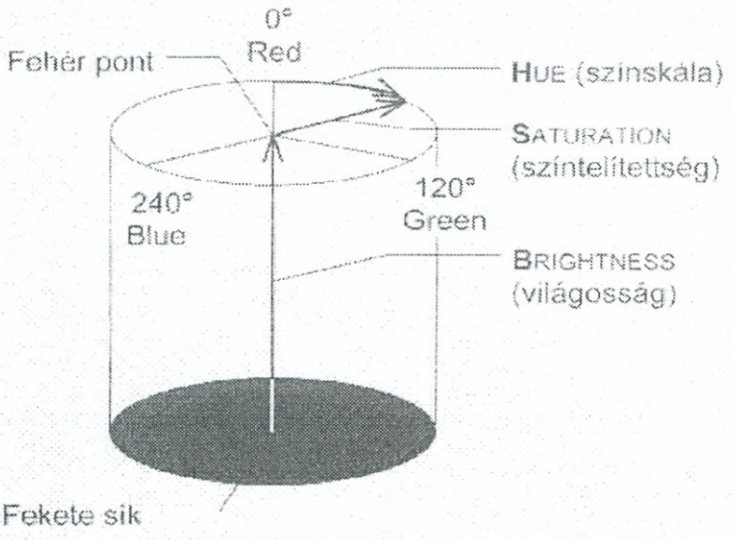
Azt az eljárást, amellyel a monitoron megjeleníthető színeket közelítjük a nyomtatott színekhez, kalibrálásnak nevezzük. Ennek legelterjedtebb megoldása az úgynevezett γ kalibráció. Ennek lényege: a képernyőn 50%-os intenzitással kigyújtott képpontok színe egyezzen meg az 50%-os fedettségű nyomdai raszter színével. (Ehhez a professzionális programcsomagok a szükséges szoftvertámogatást általában biztosítják. Így például lehetővé teszik a C, M, Y, K alapszínű képek külön-külön való nyomtatását és ennek során egy kalibrációs színmintát is nyomtatnak a kép mellé.)

Az RGB, a CMY és CMYK színterek felépítését alapvetően technikai szempontok határozták meg. Ezért olyan színtereket is kialakítottak, melyek jobban alkalmazkodnak az emberi érzékeléshez.

Egy ilyen a *HSB színtér (nevét a HUE = színárnyalat, SATURATION = színtelítettség, BRIGHTNESS = világosság, angol kifejezések rövidítéséből kapta), melyben az RGB alapszínek mellett a színek előállításához a színtelítettség és a megvilágítás erősség értékeit is felhasználhatjuk.* Megjegyezzük, hogy a HSB színtért esetenként HLS-nek is szokták nevezni.

A HSB színtért egy hengerkoordináta-rendszerben ábrázolhatjuk a 123. ábra szerint.

A HSB színtérnél a kör 360°-jából egy konkrét szögértékkel jellemezhetjük az RGB színek közötti átmeneteknek megfelelő színárnyalatokat. A kör középpontjától mért távolsággal fejezhetjük ki a színtelítettséget, és a henger alsó alapkörétől mért távolság adja meg a fényerősséget.



123. sz. ábra
A HSB színtér

4.3.3. Színek kódolása a számítógépes grafikában

Indexelt színkezelés palettával

A számítógépes grafikában a színek kódolására kezdetben 8-bitet használtak, mely összesen 256 fajta színárnyalat kezelését tette lehetővé. Ezt általában az úgynevezett színpaletta módszerrel oldották meg. *Ekkor egy színnek a kódját egy 256 elemű színtáblázatra való hivatkozás (index) jelentette. A programcsomagok a színpalettás indexelt színkezelésnél általában több paletta definiálását is lehetővé tették* és biztosították, hogy a palettát a színárnyalatokkal a felhasználó a képernyőn is megjelenítse és itt válassza ki a megfelelő színárnyalatot.

!

Többcsatornás színkódolás

A hardver fejlődésével egyre inkább a *többcsatornás színkódolás* vált általánossá. Ez *alatt azt értjük, hogy a színtér alapszíneinek intenzitás értékét külön-külön adjuk meg.*

!

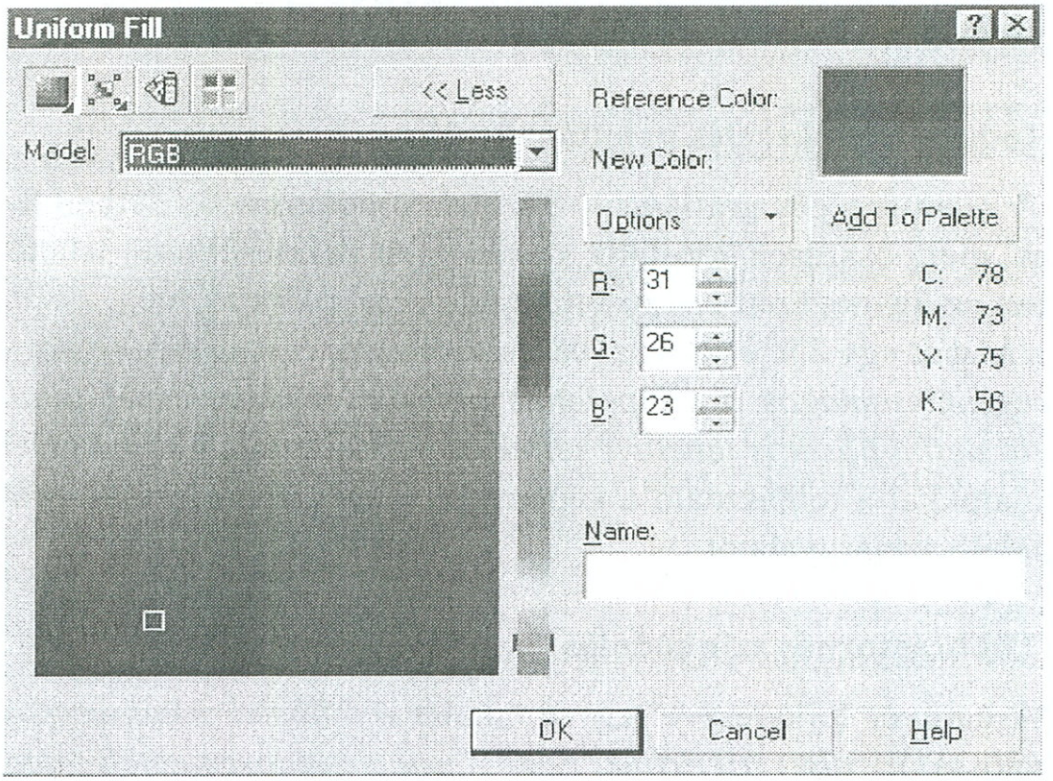
Ha három alapszínünk van – mint például az RGB színtérben –, akkor *csatornánként meg kell határozni, hogy hány biten kódoljuk az adott alapszín intenzitásértékét.*

Az úgynevezett HIHG-COLOR színmegjelenítés esetén a három RGB csatornát összesen 16-biten kódoljuk. Ez összesen 2^{16}*, azaz közelítőleg 32000 különböző színárnyalat megjelenítését teszi lehetővé!*

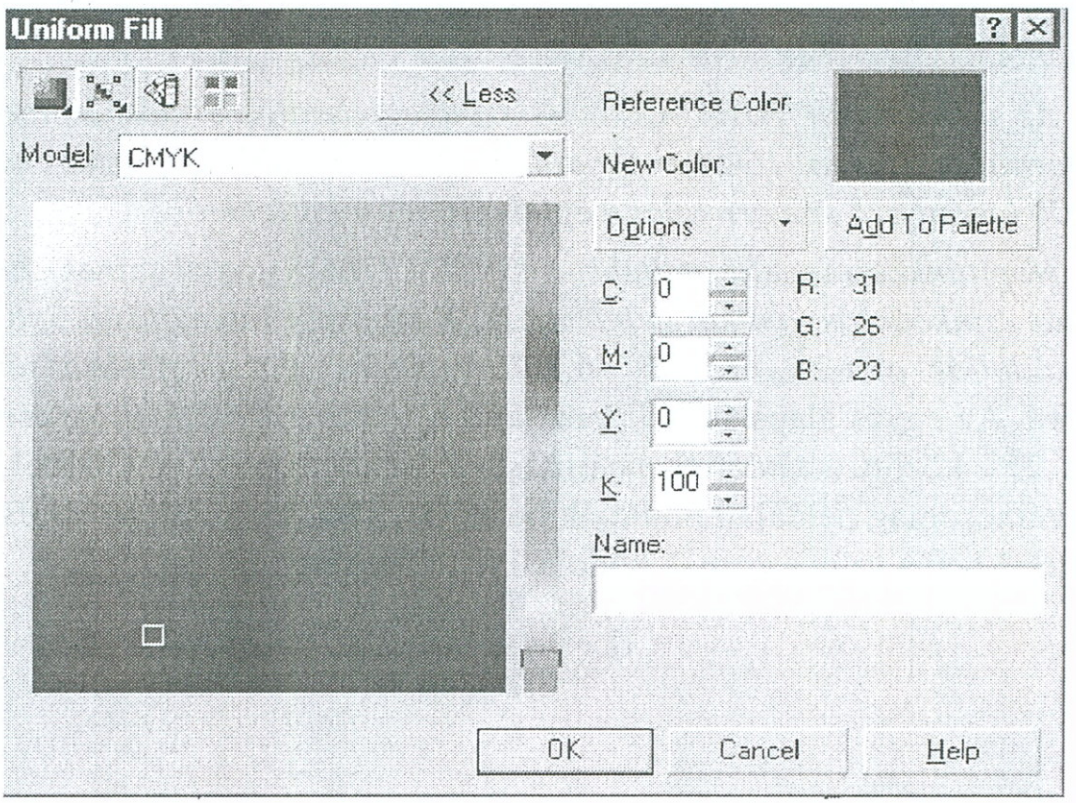
!

Napjainkban legelterjedtebb a TRUE COLOR (igazi színek) háromcsatornás színkódolás, amikor a három RGB alapszín intenzitását 3x8 = 24 biten kódolják, ami 2^{24}*, azaz kb. 16 millió színárnyalat megjelenítését teszi lehetővé.* Az egyes alapszínek intenzitását a felhasználó ebben az esetben egy 0 és 255 közötti számérték beállításával adhatja meg. A COREL DRAW programcsomag példáján mutatják be a 124, 125. sz. ábrák az RGB, CMYK színterekben a színcsatornák kódolási lehetőségeit.

!



124. sz. ábra
Az RGB színtér 3 csatornájának beállítása



125. sz. ábra
A CMYK színtér 4 csatornájának beállítása

ELLENŐRZŐ KÉRDÉSEK ?

150. *Hogyan érzékeli az ember a szemével a fényt?*
151. Jellemezze a P, D, T csapokat!
152. Mit nevezünk akromatikus fénynek?
153. *Milyen határok között képes az emberi szem a fényerősséget érzékelni?*
154. Mit jelent a színtelítettség és hány fokozatát képes az ember megkülönböztetni?
155. *Melyek az emberi színlátás legfontosabb összetevői?*
156. Milyen az emberi szem felbontóképessége?
157. *Milyen szerepe van a vizuális emlékezetnek a tárgyak felismerésében?*
158. Mit értünk a látás környezetfüggősége alatt?
159. *Mi biztosítja, hogy a különböző távolságú tárgyakat élesen lássuk?*
160. *Milyen tényezők segíthetik, hogy egy kép térhatású legyen?*
161. Mi a térlátás alapja?
162. *Hogyan keletkezik a látás során sztereoélmény a legújabb kutatások szerint?*
163. *Mennyi idő szükséges, hogy egy képet tudatosan érzékeljünk?*
164. *Hány állóképet kell megjeleníteni másodpercenként, hogy az ember mozgóképet érzékeljen?*
165. Mi az aranymetszés? Mondjon egy példát az aranymetszés hasznosítására a képművészetben!
166. Sorolja fel a reklámgrafika néhány litográfiai szabályát!
167. Mi a különbség a fényterjedés sugárfizikai és fotometriai tárgyalása között?
168. Mi a sugárzási teljesítmény?
169. Mi a kisugárzás erősség?
170. Mi a besugárzás erősség?
171. *Mit mond ki a Lambert-féle távolság- és koszinuszfüggvény?*
172. *Definiálja a sugársűrűség fogalmát!*
173. Milyen fontos tulajdonsága van a sugársűrűségnek a grafikus algoritmusok szempontjából?
174. Mi a legfontosabb különbség a vizuális fotometria és a sugárfizika között?
175. *Definiálja a candela fogalmát!*
176. *Mi a megvilágításerősség egysége?*
177. Sorolja fel a sugárfizikai mennyiségeket!
178. Sorolja fel a fotometriai mennyiségeket!
179. Mitől függ a fénysűrűség?
180. Mi jellemzi a Lambert-fényforrásokat?

181. A fényvisszaverődésnek milyen típusai vannak?
182. *Mi az ideálisan diffúz visszaverődés?*
183. *Mi az ideálisan tükröző visszaverődés?*
184. *Mi az irányított diffúz visszaverődés?*
185. Hogyan modellezzük a fénytörést?
186. *Az átlátszó testeken való fényáthaladásnak milyen esetei vannak?*
187. Mi az átlátszó közegek fényvisszaverő és áteresztőképessége?
188. Hogyan képezik a színeket az RGB monitorok?
189. Ismertesse a színkeverés Grassmann-törvényeit!
190. Jellemezze a színtereket mint vektortereket!
191. Jellemezze a CIE normalizált kétdimenziós színdiagramját!
192. *Mit értünk additív színkeverés alatt?*
193. *Mit értünk szubsztraktív színkeverés alatt?*
194. *Mi az RGB színtér?*
195. *Mi az CMY színtér?*
196. *Mi jellemzi a CMYK színteret?*
197. *Adja meg az RGB és CMY színterek közötti átszámítás szabályait!*
198. *Mi jellemzi a HSB színteret?*
199. *Mi jellemzi az indexelt palettás színkezelést?*
200. *Mi a HIGH-COLOR színkódolás?*
201. *Mi a TRUE COLOR színkódolás?*

V.

RASZTERGRAFIKA

E fejezetben a képpontokból (pixelekből) felépített képek számítógépes feldolgozásának alapjaival ismerkedhetünk meg. Ennek részeként először a rasztergrafika legfontosabb jellemzőit mutatjuk be, tárgyaljuk a rasztergrafikus primitívek kezelését, külön figyelmet fordítva a szöveges információk raszteres feldolgozására. Néhány speciális képtípus és a rasztergrafikus animáció megismerését követően a rasztergrafika legfontosabb algoritmusaival foglalkozunk és röviden bemutatjuk a digitális képfeldolgozás és a képtömörítés alapjait.

5.1. A RASZTERGRAFIKA LEGFONTOSABB JELLEMZŐI

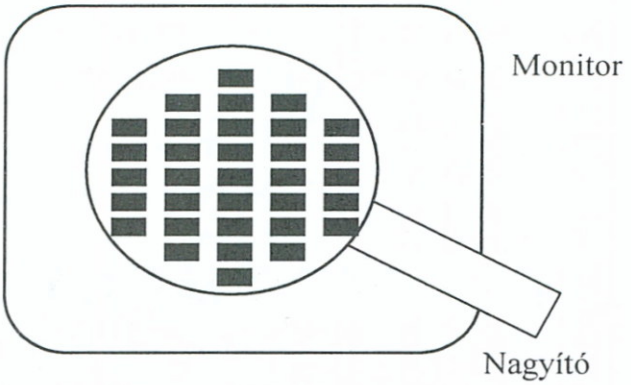
A számítógépes grafika gyakorlati alkalmazásainak egy részénél (például egyszerűbb kiadványszerkesztés) már az is megfelelő, ha lehetőségünk van a monitor képernyőjén látható képek „szerkesztésére", egyszerűbb képi objektumok generálására és a szabadkézi rajzolásra. Ehhez elegendő *a számítógépen feldolgozott képeket képpontonként ábrázolni, egy-egy képpontra megadva a képpont pozícióját (koordinátáit) a képen és a pont színét.*

!

A képpontokból (pixelekből) felépülő képet raszteres képnek nevezzük, ezek számítógépes feldolgozását pedig rasztergrafikának.

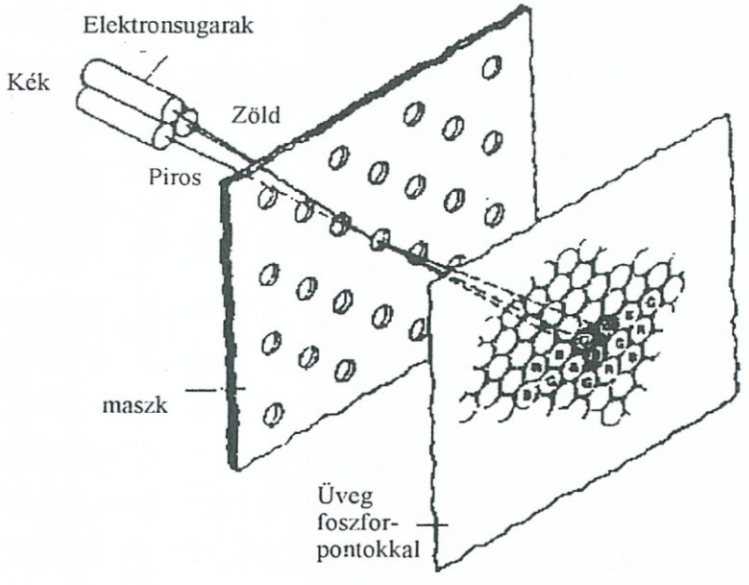
5.1.1. A raszteres kép felépítése

! *A pixel (picture element) magyarul a kép elemi, tovább fel nem bontható részét jelenti.* Ezek például a monitor képernyőjén erősen nagyítva a következőképpen helyezkednek el:

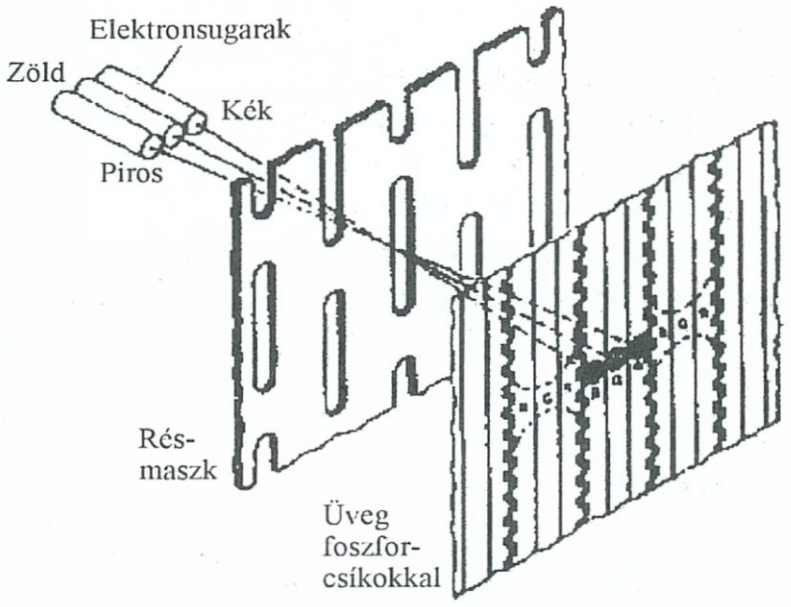


126. sz. ábra
Pixelekből álló raszteres kép

A pixelek tulajdonképpen a monitor foszforrétegén három darab, az RGB alapszíneknek megfelelő részpontból állnak, melyet a megfelelő elektronsugár „gyújt fel". Ezek elrendezése deltaformátumú vagy soros formátumú lehet, utóbbi esetben in-line képcsőről beszélünk (lásd a 128. és 128. sz. ábrát).



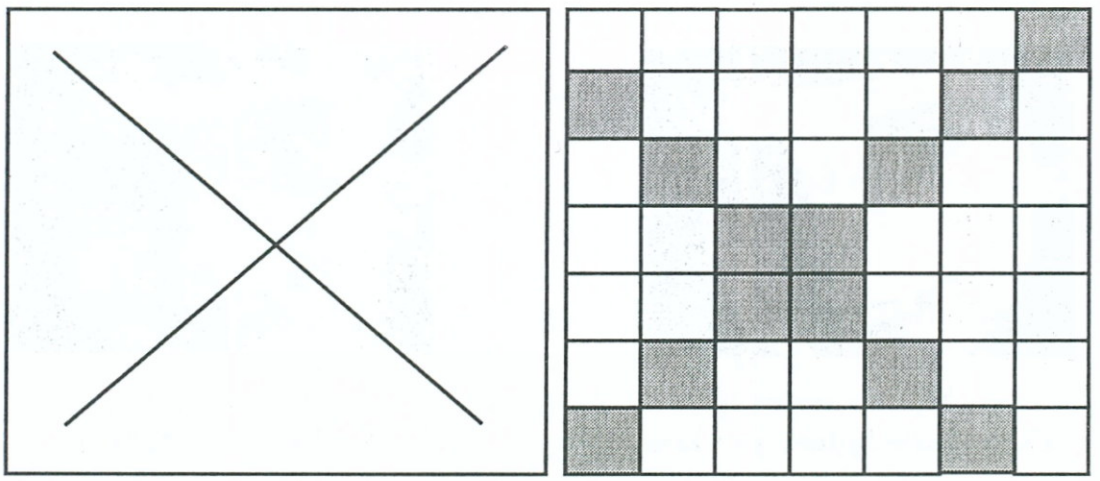
127. sz. ábra
A monitorképernyő foszforpontjainak deltaelrendezése



128. sz. ábra
A monitorképernyő foszforpontjainak in-line elrendezés

Megjegyezzük, hogy a „képpont" megfogalmazás esetenként félvezető lehet, mivel – helytelenül – a geometriai értelemben kiterjedés nélküli pontra is gondolhatunk.

Ha raszteres képernyőn a képpontok elég „sűrűn" helyezkednek el és „gyorsan" frissítik a képpontokat, szemünkkel nem pontokat, hanem összefüggő képet látunk. Ennek ellenére a képernyőn két, látszólag egymást egy pontban metsző egyenesnek egy vagy több közös képpontja is lehet, és az is előfordul, hogy a két metsző egyenesnek nincs közös pontja.



Két metsző egyenes Erősen nagyítva

129. sz. ábra
Egyenesek metszése raszteres képen

Az egyenesek a raszteres képen tehát nem „pontszerű"-en találkoznak, mint a geometriában. Ezért például a metszéspont és az érintő a geometria szabályai szerint raszteres képernyőn nem értelmezhető és geometriai szerkesztésre sincs (vagy csak látszólagosan van) lehetőség.

Mivel a rasztergrafikában elméletileg tökéletes, kiterjedés nélküli vonalakat, pontokat nem tudunk ábrázolni, csak az lehet a célunk, hogy ezekhez legjobban hasonlító és „legszebb" eredményt adó képpontsorozatokat jelenítsünk meg.

5.1.2. Színkezelés

A megjeleníthető színek mennyisége alapján négyfajta raszteres képtípust különböztethetünk meg. Ezek lehetnek:

- *bittérképes képek* (bitmapped image),
- *szürkeárnyalatú képek* (grayscale image),
- *színpalettával indexelt képek* (indexed color image),
- *valódi színezetű képek* (true color image).

Bittérképes kép esetén minden egyes képponthoz tartozó színinformációt 1 biten (1 = fekete, 0 = fehér) kódoljuk, így ezek a képek fekete-fehérek (lásd a 130. sz. ábra).

A szürkeárnyalatú kép – képpontonként 8 biten kódolva – 256 féle fekete-fehér átmeneti „színt" tartalmazhat. Ezzel már jobb minőségű képet állíthatunk elő, ez például már egy igazolványképnek megfelelő lehet (lásd a 131. sz. ábra).

130. sz. ábra
Fekete-fehér bittérképes kép

131. sz. ábra
Szürkeárnyalatú kép

A színpalettával indexelt képek pixeljeihez egy színindex értéket rendelünk hozzá, mely egy 256 elemű színtáblázatra hivatkozik, melyet palettának nevezünk. Ezt a rasztergrafikus rendszerek általában a képernyőn is megje-

lenítik és így biztosítják a felhasználó számára a szín interaktív (például egérrel való rámutatás) kiválasztását. *A színpaletta minden 8 bites indexe tehát egy konkrét színárnyalatot határoz meg egy pixel számára.* A rasztergrafikában általában mód van arra, hogy színpalettával indexelt képeknél képenként különböző színpalettát alkalmazzunk (lásd 132. sz. ábra).

A valódi színezetű (True color) képek esetében a színtér alapszíneinek megfelelő színcsatornánként adjuk meg az alapszínek intenzitását. Ez RGB vagy CMY színtér esetén 3x8 = 24 bit, a CMYK színtér esetén pedig 4x8 = 32 bit megadását jelenti. Így például RGB alapszín intenzitások keverésével 2^{24}*, azaz több mint 16 millió (16 777 216) különböző színárnyalatot tudunk megkülönböztetni.*

132. sz. ábra
Színpalettával indexelt kép

133. sz. ábra
TRUE COLOR raszteres kép

5.1.3. Raszteres képek tárolása és visszakeresése

A raszteres képfájlok a számítógép tárolóeszközein általában a következő részekből épülnek fel:

- fejléc – megadja a kép formátumát, méretét pixelekben (esetleg a palettát),
- adatrész – pixelenként tartalmazza a színkódokat.

Ebből azonnal következik a rasztergrafika egyik legfontosabb tulajdonsága:

A raszteres kép csak teljes egészében kereshető vissza és csak felülírással módosítható. Ez azt jelenti, hogy a raszteres képen lévő elkülönült grafikus objektumokat egyedileg nem tudjuk visszakeresni. Ha egy objektum egy részletét módosítjuk, akkor a teljes képet meg kell változtatni.

Így például, ha egy raszteres ábrán az egyes rajzelemek feliratát módosítani akarjuk, akkor a teljes képet vissza kell keresnünk és megváltoztatva a szöveget, „új" képként kell letárolnunk.

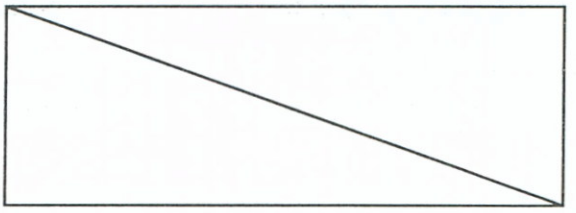
5.1.4. Rasztergrafikus primitívek

A rasztergrafikus primitívek olyan, a programcsomagokba beépített rajzelemgenerátorok, melyekkel a felhasználó tipikus rasztergrafikus objektumokat hozhat létre. Ilyenek például:

- *vonalak,*
- *sokszögek* (háromszög, négyzet, téglalap stb.),
- *kör és ellipszis,*
- *szövegek* (betűcsalád, betűtípus, például: vastag, dőlt, keskeny, széles, árnyékolt stb.).

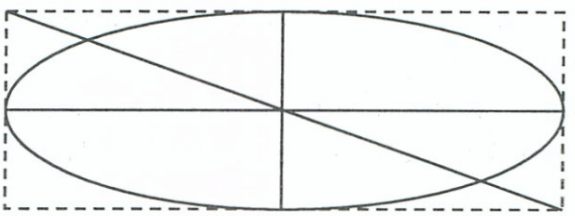
A primitív generátornak megfelelő programrutin számára a felhasználó megadja az igényelt primitívek paramétereit – például egy kör esetében a kör középpontjának koordinátáit és sugarát -, mely alapján a program legenerálja a megfelelő képponttömböt.

Primitívek megadására és kirajzolására jellemző példa egy téglalap definiálása. Ezt a téglalap bal felső és jobb alsó csúcsának megadásával tehetjük meg (koordináta értékekkel, vagy a csúcsoknak megfelelő átló egérrel történő meghúzásával, lásd a 134. sz. ábra).



134. sz. ábra
Téglalap primitív definiálása

Egy másik példa egy ellipszis megadása lehet, melyet bennfoglaló téglalapjának definiálásával határozhatunk meg (lásd a 135. sz. ábra).



135. sz. ábra
Ellipszis primitív megadása

Azokat a primitíveket, melyek a raszteres képen egy területet körbezárnak (azaz a határoló görbe bezáródik) terület-meghatározó primitíveknek nevezzük. Ezek lehetnek vonalasak vagy kitöltöttek (utóbbihoz a határolóvonal nem tartozik hozzá).

5.1.5. Rajzelemek, vonalstílus, vonalvastagság, kitöltő mintázat

A rasztergrafikában rajzelemeket „szabadkézzel" (egérrel) vagy primitívekkel hozhatunk létre.

A rajzelemekhez tulajdonságokat rendelhetünk hozzá. Ezek lehetnek:

- *vonalstílus,*
- *vonalvastagság,*
- *szín és*
- *területmeghatározó primitívek esetében kitöltő szín, illetve mintázat.*

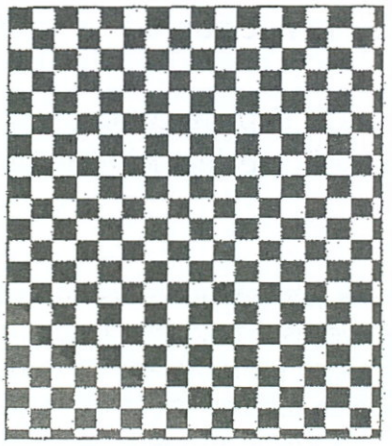
A rajzelemekhez rendelt tulajdonságokat a rasztergrafikus programcsomagok mindaddig megőrzik, míg azokat a felhasználó meg nem változtatja.

A vonalstílus és vastagság tulajdonképpen a számítógépen belül egy speciális pixelekből álló, előre definiált maszk alkalmazását jelenti, mely vezérli a képpontok megjelenítését a képernyőn.

A kitöltő mintázat (lásd a 136. sz. ábra) egy előre letárolt pixeles maszk, mellyel a raszteres programcsomag a kijelölt területmeghatározó primitívet kitölti. Ennek négy típusát szokták megkülönböztetni:

- solid: egy kiválasztott színnel állít elő egy kitöltést,
- bitmap pattern opaque: a mintázat minden képpontja felülírja az eddigi mintázatot,
- bitmap pattern transparent: a mintázat csak bizonyos színű képpontokat ír felül,
- pixmap pattern: ekkor a mintázat többszínű maszkot tartalmaz, mely teljesen felülírja a kitöltendő területet.

A mintázatot általában egy táblázatból választhatja ki a felhasználó.



136. sz. ábra
Kitöltő mintázat

A rasztergrafikus rendszerekben kitüntetett szerepe van a háttérszínnek, mely általában a „0" értékű színkódhoz van rendelve. (Ez a magyarázata annak, hogy ha egy alakzatot elmozgatunk a képernyőn, akkor a helyén a háttérszín marad vissza.)

5.1.6. Szövegek kezelése raszteres képernyőn

A szöveges információk grafikus felhasználói felületen való megjelenítésének általánossá válásával a számítógépes grafikának fontos részterületei kapcsolódnak a szöveg- és kiadványszerkesztéshez.

A szöveg és kiadványszerkesztő programok között az alkalmazott eljárások és módszerek vonatkozásában egyre inkább elmosódik a határ. Ezeket ma már csak céljuk szerint különböztethetjük meg:

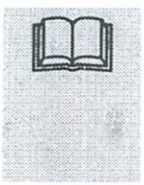
- a szövegszerkesztő programcsomagok célja a szöveges információk számítógépen való, különösebb szakértelmet nem igénylő feldolgozása,

- *a kiadványszerkesztő programcsomagok célja, hogy a felhasználó számára a szöveg szerkesztése mellett olyan funkciókat is biztosítson, mellyel a kiadványok nyomdai előállításához szükséges előkészítő munka teljes körűen elvégezhető.* Ilyenek például: a nyomdai minőségű szedés, tördelés, több hasáb (kolumna) kezelése, fotók, grafikák négyszínre bontással történő nyomtatása, a szövegelemek helyének tipográfiai pontosságú meghatározása.

Az előbbiekre figyelemmel a nyomdai kiadványszerkesztést (Desk Top Publishing = DTP) úgy definiálhatjuk, mint nyomdai anyagok előállítását számítógéppel és a megfelelő szoftverrel.

Ezek szerint a szövegek szerkesztése, tördelése része a DTP-nek, de ebbe a fogalomba még bele kell értenünk a számítógépes képfeldolgozás és grafika egy részét is.

Ennek megfelelően a DTP programcsomagok eszközkészletében a szövegelemeket kezelő eljárások mellett különböző rajzi elemeket generáló részeket is megtalálhatunk.



A DTP szoftverek jelzőjeként sokszor megjelenik a WYSIWYG (What You See Is What You Get) megnevezés. Ez azt jelenti, hogy a számítógép képernyőjén látott színek, méretek, vonalvastagságok pontosan azonos módon fognak megjelenni nyomtatásban is. (Ez azonban sok esetben részben reklámfogás. Mielőtt ezt a kijelentést szó szerint vennénk, célszerű igazságát a gyakorlatban is letesztelni.)

A szövegek grafikus megjelenítéséhez első lépés a megfelelő betűkészlet megválasztása. A formák meghatározásához a rasztergrafikus rendszerek a következő lehetőségeket biztosítják:

- *betűcsalád* (Times, Helvetica stb.) *meghatározása,*
- *betűtípus* (vastag, dőlt stb.) *kiválasztása,*
- *betűnagyság rögzítése.*
-

5.1.6.1. Betűcsaládok

A magyar számítástechnikai szakzsargonban a „font" megnevezést gyakran betűcsalád szinonimájaként is szokták használni, ami nem teljesen pontos.

Font alatt mindig egy jelkészletet értünk, melyben lehetnek betűk, számok, matematikai szimbólumok stb. Ha szöveges információk kezeléséről van szó, akkor a font mindig egy betűkből álló jelkészletet jelent.

Betűcsaládnak nevezzük az azonos grafikus jellemzőkkel és formai sajátosságokkal rendelkező betűk összességét. Ha a betűcsaládot kiválasztjuk, akkor egy betűformátumnak megfelelő fontot és egyúttal az ezt generáló programcsomagrészt is rögzítettük.

A nyomdászatban korábban a betűcsaládokat felhasználásuk szerint csoportosították, így például voltak „cím" betűk. Az e téren mutatkozó sokszínűség (a betűcsaládok elnevezése és ennek értelmezése például országonként különbözött) okozta zavarok miatt hozták létre a nyomdászatban az ún. „tizedes osztályozás" szabványt, mely – ólom-, fényszedő és számítógépes betűtípusokra egyaránt érvényesen – leírja a betűcsalád grafikus jellemzőit. Például ilyen osztályok:

- Talp nélküli lineáris antikva: egyenlő vonalvastagságú, talpnélküli betűcsalád. Ide tartozik például a Helvetica és az Avantgard (lásd 138/a. sz. ábra).
- Talpas lineáris antikva: talpas, egyenlő vonalvastagságú antikva, mint például a Candida (lásd 138/b. ábra). (Ezzel kapcsolatban érdekes, hogy pszichológiai vizsgálatok alapján az ember talpas betűformákat könnyebben olvassa.)
- Írott betűformák: Az emberi kézíráshoz hasonló forma, általában stiláris díszítőelemként alkalmazzák (lásd 138/c. ábra).
- Idegen betűformák: Ide tartoznak például a héber, görög, arab, kínai stb. betűk (lásd 138/d. sz. ábra).

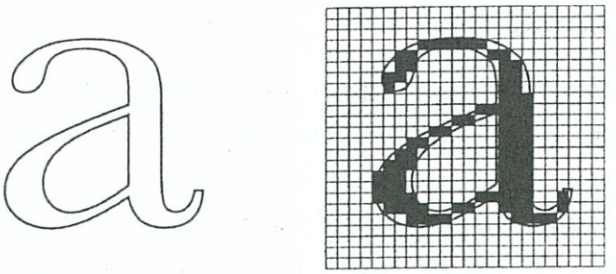
a) b) c) d)

137. sz. ábra
Példák a nyomdászatban használatos betűcsaládokra

A betűcsaládok – generálásuk elve szerint – *vagy vektorosak vagy raszteresek.* A Windows operációs rendszerben például megtalálhatjuk

- *TrueType, méretezhető vektorbetűket (kiterjesztésük .ttf) és a*
- *bittérképes rendszerfontokat (kiterjesztésük .fon)*

A TrueType font a betűk körvonalát görbékkel írja le, míg a raszteres (bittérképes) fontok a betűket egymásmelletti pixelekkel közelítik meg (lásd 138. sz. ábra).



138. sz. ábra
A vektoros és raszteres betűk képzésének elve

Természetesen mind a vektoros, mind a raszteres betűk képernyőn vagy nyomtatásban való megjelenítésükkor képpontokból állnak. A lényeges különbség közöttük:

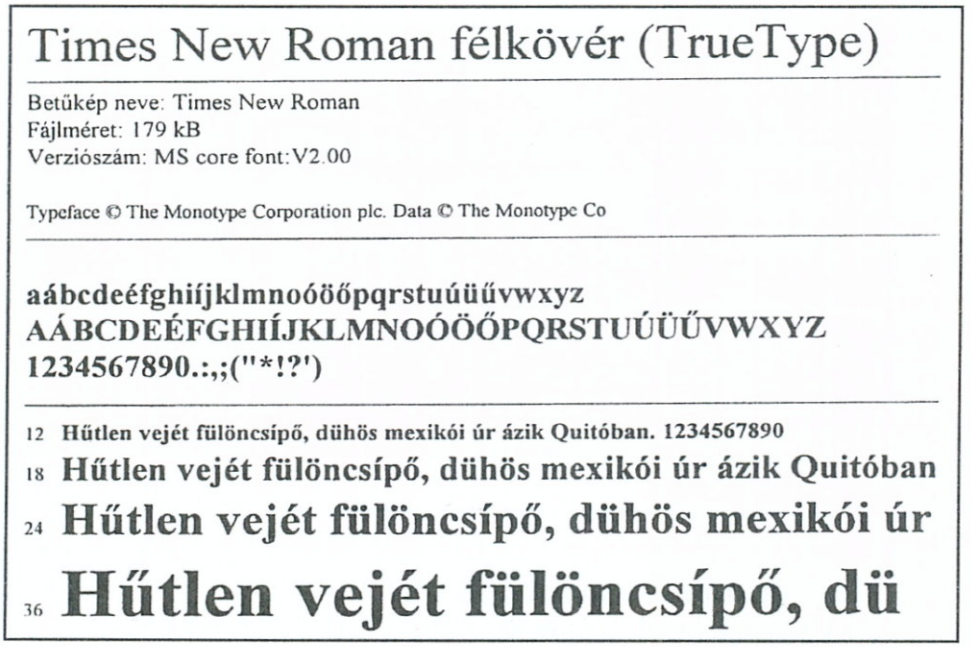
- a raszteres betűk megjelenítése egy az egyben a letárolt képpontinformációk alapján történik,
- a vektoros betűk méretezhetők, azaz ha egy vektoros betű méreteit megadjuk, akkor a betűcsaládnak megfelelő programrész először kiolvassa a betű körvonalinformációit és kiszámítja, hogy az adott méretben a vonalak hol helyezkednek el, így megkapja a betű határoló körvonalait, ebből határozza meg a megjelenítéskor érintett pixeleket.

A gyakorlatban elterjedt betűcsaládok a következők:

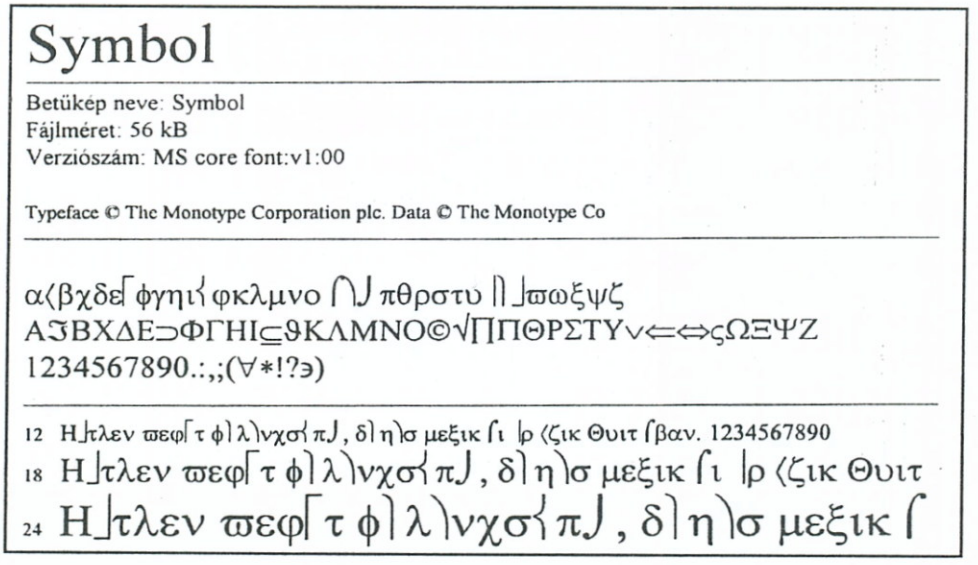
- dekoratív,
- modern,
- roman (talpas),
- script (írott),
- swiss (egyenes, talpnélküli),
- idegen betűk.

Sajnos – mint ez vélhetően az olvasónak is feltűnt – a számítástechnikai gyakorlatban használt betűcsalád osztályozás tartalmában és kategóriában egyaránt eltér a nyomdatechnikában bevezetett „10-es osztályozás"-tól.

A WINDOWS-95 Fonts mappájában található fontfájlok közül egy TrueType típusút mutat be a 139. sz. ábra, és egy matematikai jelkészletet tartalmazó raszteres fontot a 140. sz. ábra.



139. sz. ábra
TruType vektoros betűk a Roman betűcsaládból



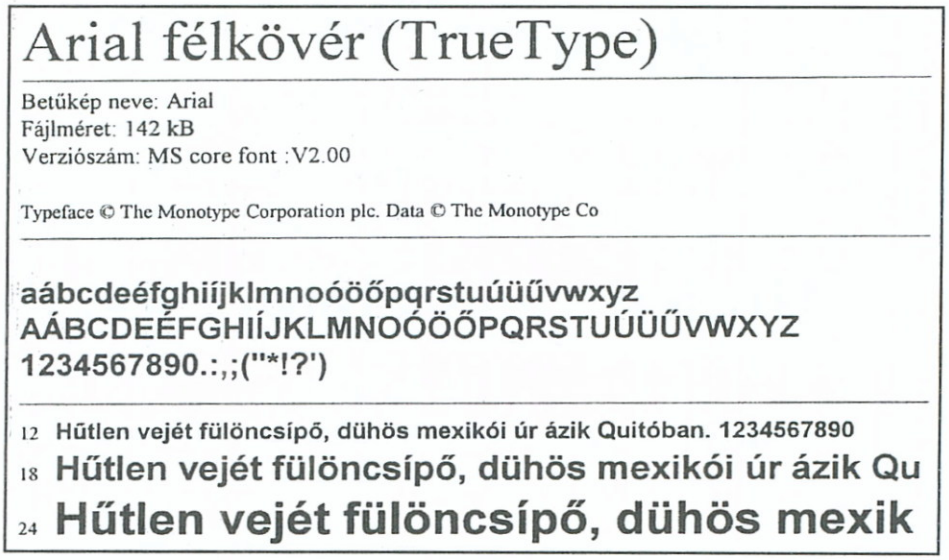
140. sz. ábra
Raszteres matematikai jelkészlet

5.1.6.2. Betűtípusok

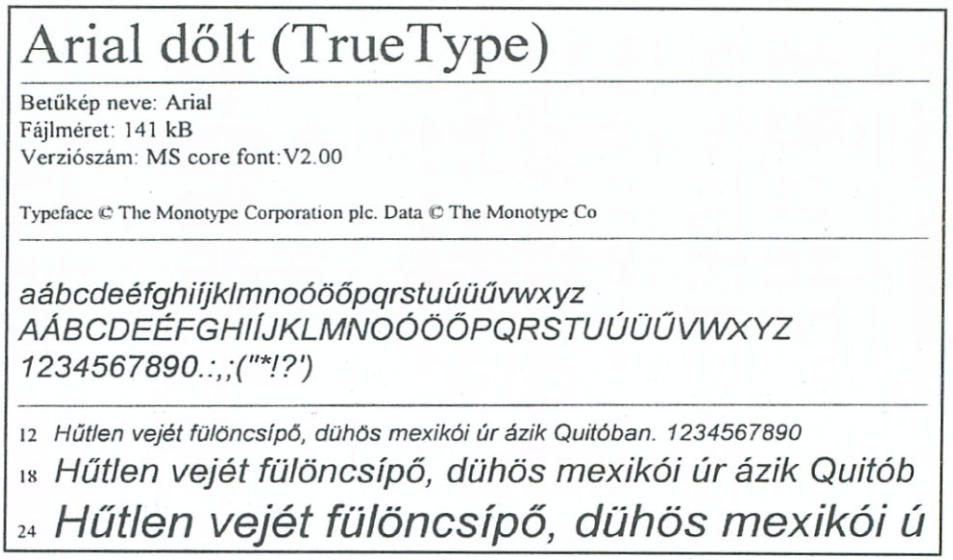
A betűcsaládokon belül betűtípusokat különböztethetünk meg. *A betűtípusok megtartják a család általános grafikai jellemzőit, de néhány tulajdonságukban eltérhetnek egymástól. Ezek:*

- *a betűkép sötétebb vagy világosabb megjelenése,*
- *a betűkhöz tartozó vonalak vastagsága (vékony, normál, félkövér, kövér),*
- *egyenesállású vagy döntött a betű képe.*

Az ARIAL gyakran alkalmazott betűcsaládon belül a különböző betűtípusokra mutat példát a 142. és 142. sz. ábra.



141. sz. ábra
Arial félkövér betűtípus



142. sz. ábra
Arial dőlt betűtípus

Néhány gyakran használt betűtípus angol megnevezése és rövidítése a következő:

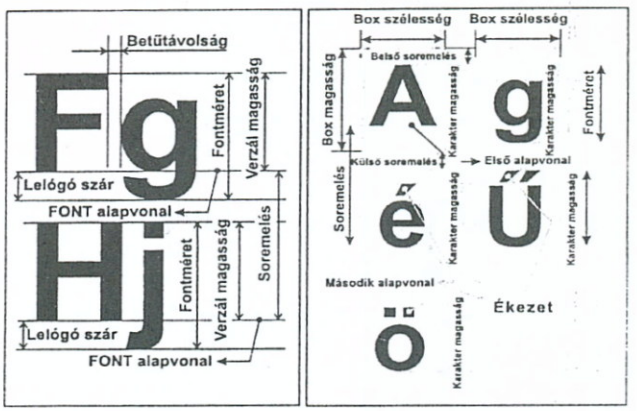
Bd	Bold	félkövér
Bk	Book	normál
Blk	Black	kövér
Cn	Condensed	keskeny
Dm	Demi	félkövér
El	Elongated	nyújtott
Ex	Extended	széles
Hv	Heavy	kövér
It	Italic	dőlt
Md	Medium	kövérebb
Oul	Outline	körvonalas
Rm	Roman Normal	normál talpas
Rnd	Rounded	kerekített
XBd	Extra Bold	kövér
XCn	Extra Condensed	nagyon keskeny

5.1.6.3. A betűk mérete

A harmadik fontos jellemzője a betűknek a betűnagyság, vagy betűméret.

A betűk méreteit tipográfiai pontokban, mint hosszmértékben szokás megadni: egy tipográfiai pont értéke Magyarországon 0,376 mm, angolszáz mértékegységben viszont ez csak 0,351 mm.

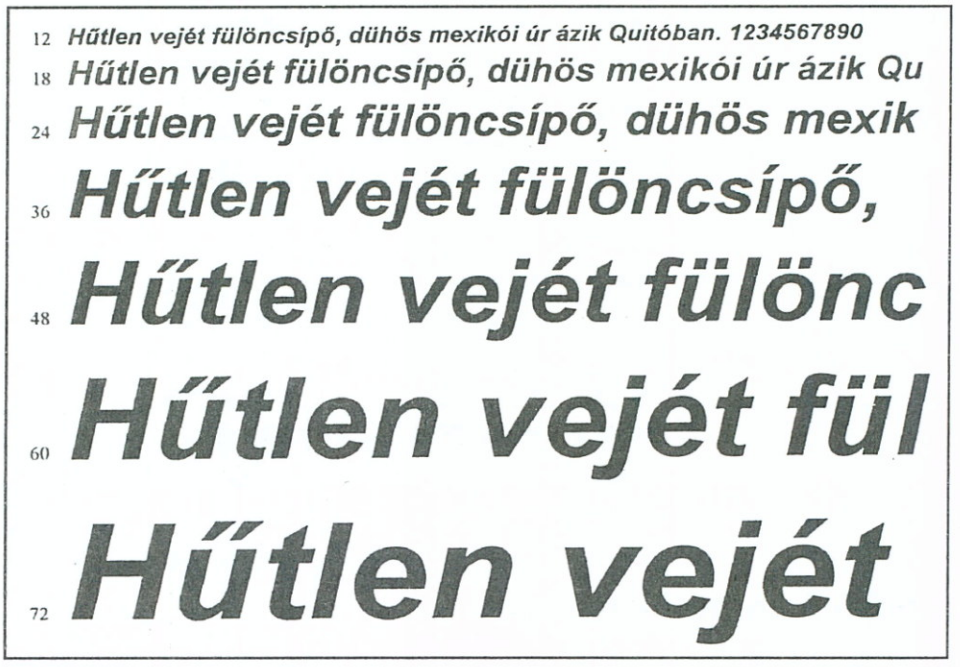
A betűk méretezéséhez használt legfontosabb fogalmakat a 143. sz. ábra mutatja be.



143. sz. ábra
A betűk méretezéséhez használt fogalmak

Az ábrán látható méretek döntő többsége külön magyarázatot nem igényel. Verzál = nagy betű magasság. Boksz = a betűt tartalmazó téglalap, melyet karaktercellának is szoktak nevezni.

A különböző nagyságú betűkre, az Arial bold italic típus esetén a 144. sz. ábra mutat példákat.



144. sz. ábra
Különböző nagyságú Arial félkövér dőlt betűk

5.1.7. A rasztergrafikus megjelenítést biztosító hardver funkcionális elemei

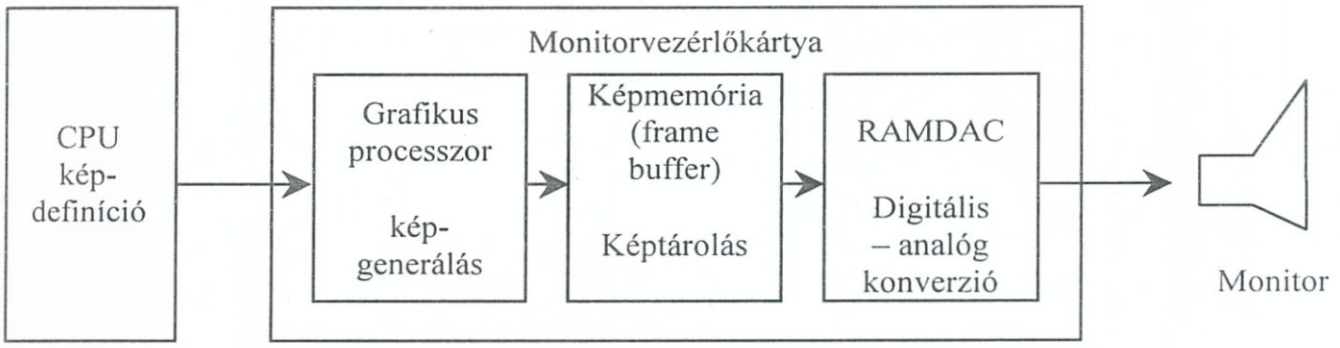
A rasztergrafika legfontosabb perifériái:

- a monitor, a monitorvezérlő (2D-s és 3D-s gyorsító) kártyával,
- a szkenner,
- a különböző típusú nyomtatók,
- nyomdai levilágítók.

Ezeket részletesen fogjuk tárgyalni a VIII. fejezetben. Az elkövetkezendőkben csupán – a monitor és monitorvezérlőkártya példáján bemutatva – a rasztergrafikus hardver alapvető feladatait tekintjük át.

A raszteres képernyőn történő képmegjelenítés feladatainak egyre jelentősebb része napjainkban már a monitorvezérlő kártyára hárul, ezáltal tehermentesítve a processzort.

Az együttműködő hardver egységek közötti, a raszteres kép megjelenítése során létrejövő feladatmegosztást mutatja a 145. sz. ábra.

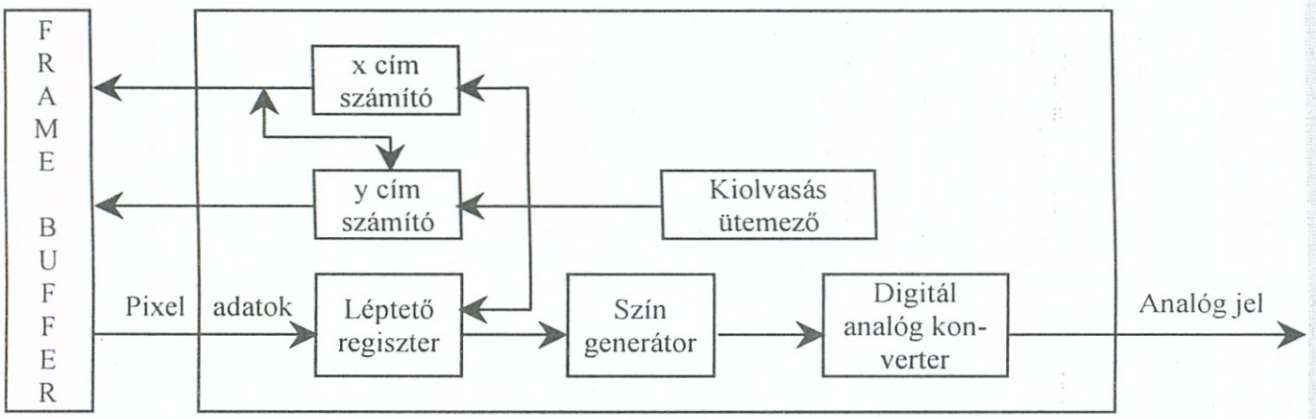


145. sz. ábra
A hardver egységek közötti feladatmegosztás
a raszteres kép megjelenítésében

A képpontok kiszámítása raszteres primitívekkel történik meg, és ennek eredményét a grafikus processzor a képmemóriába (frame buffer) írja be. A képmemória két porttal rendelkezik, ezt a grafikus processzor és a digitális-analóg konverter konkurens módon használja.

A RAMDAC digitális-analóg konverter a képmemóriát olvasva vezérli a monitor kirajzoló elektronágyúit és így jelenik meg a kép a monitor képernyőjén.

A Video-controller a képmemóriát a következő séma szerint olvassa:



146. sz. ábra
A képmemória kezelése a monitorvezérlőkártyában

Absztrakt logikai eszköztípusok

A grafikus programcsomagok tervezésének egyik kulcskérdése, hogy milyen módon lehet biztosítani a programcsomag függetlenségét a konkrét hardver eszközöktől.

A rasztergrafikus programcsomagok esetén a szabványok erre a problémára az úgynevezett logikai eszközök bevezetésével adják meg a választ. (SRGP szabvány, lásd a VI. fejezet)

A rasztergrafika két logikai eszközt alkalmaz. Ezek:

- a billentyűzet, mely alapvetően karakteres input adatok bevitelére szolgál, de koordinátaadatokat is közölhetünk begépeléssel,
- a lokátor eszközök, melyeknél az eszköz helyzete hozzárendelhető az absztrakt 2D-s egészértékű koordináta-rendszer egy pontjához.

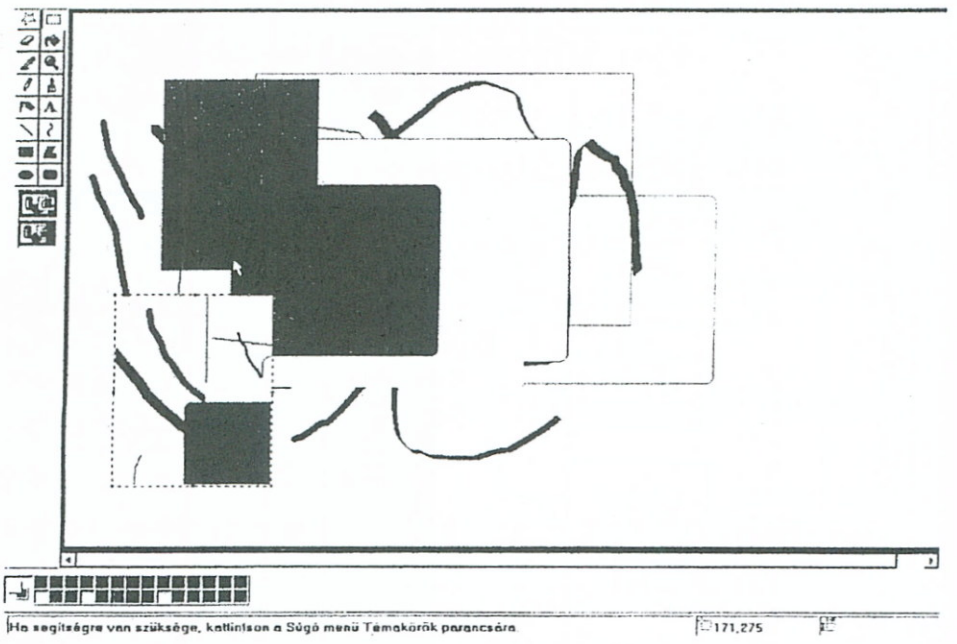
Lokátor eszközök lehetnek: egér, joystick, tablet és a touch-screen.

A lokátor eszközökkel a kapcsolattartás történhet időszakos lekérdezéssel, vagy a megszakítási rendszer segítségével.

5.1.8. A raszteres felhasználói felület funkcionális lehetőségei

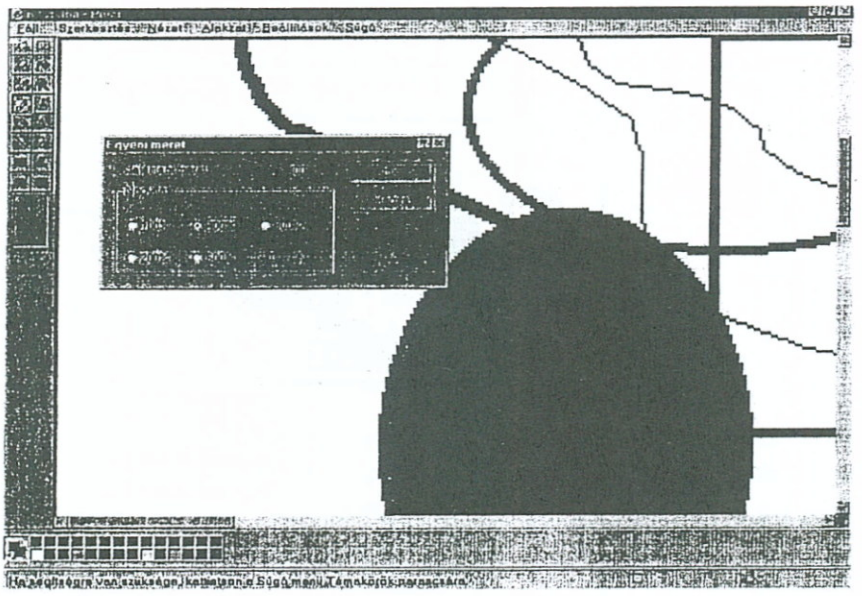
Az elkövetkezendő ábrasorozattal – a PAINT program alapján – áttekintjük a raszteres felhasználói felületen rendelkezésre álló grafikus lehetőségeket.

A rasztergrafikában a kép egy adott részét át lehet helyezni és ekkor az alakzat helyén a háttérszín marad vissza (lásd 147. sz. ábra).



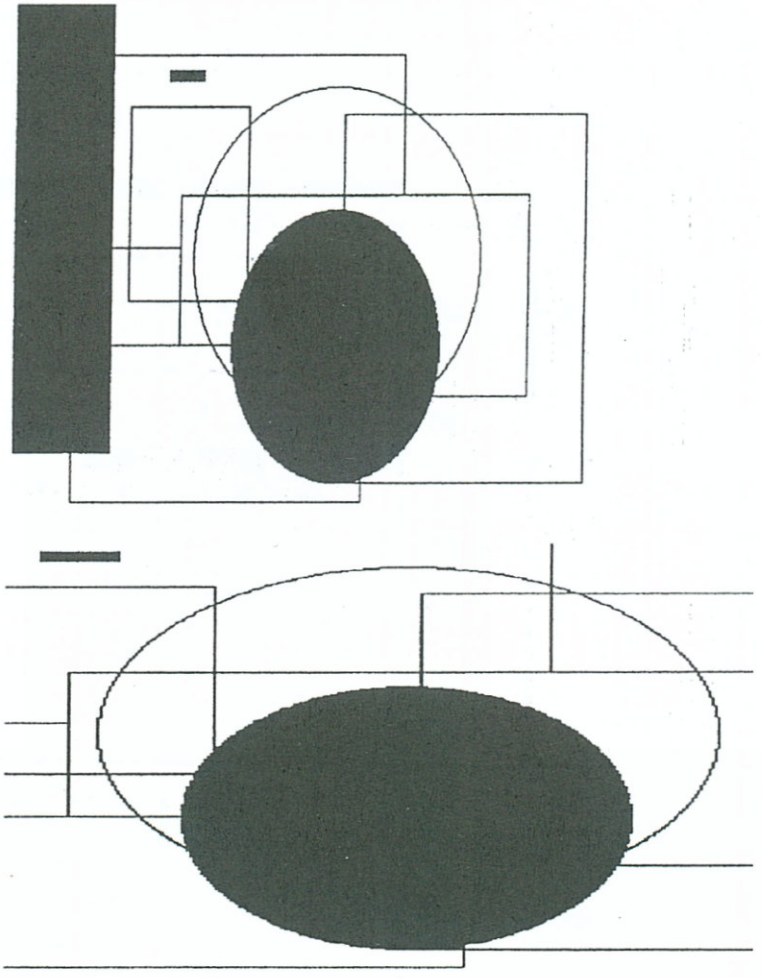
147. sz. ábra
Képrészlet elmozgatása a képernyőn

A rasztergrafikában a nagyításnak csak egész értékeket lehet megadni és az erősen kinagyított kép eldurvul (lásd 148. sz. ábra).



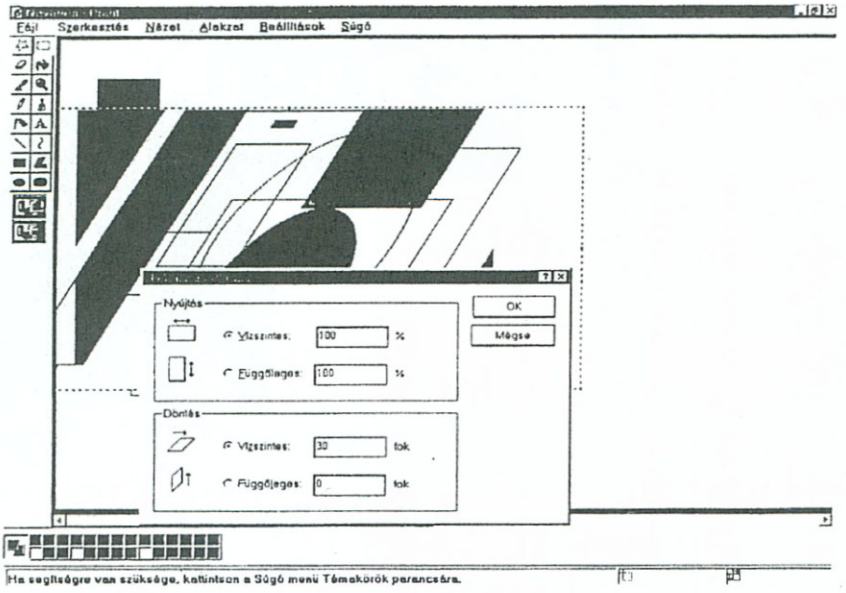
148. sz. ábra
Nagyításkor a kép eldurvul

Egy kijelölt ablak tartalmát összenyomhatjuk vagy megnyújthatjuk az X és Y tengely irányában, különböző értékekkel (lásd 149. sz. ábra).



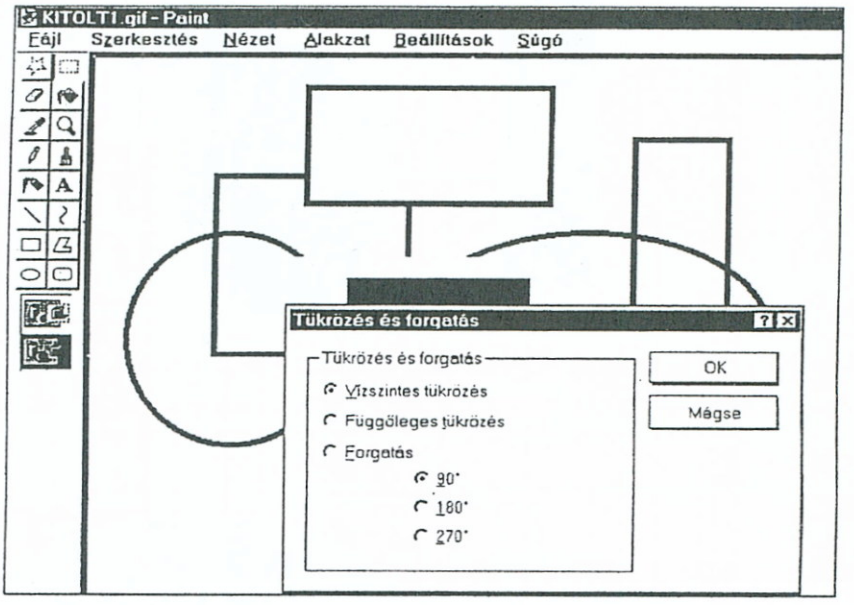
149. sz. ábra
Objektumok nyújtása és összenyomása raszteres képernyőn

Az összenyomás, nyújtás mellett szöget is megadhatunk a kép torzításának (lásd 150. sz. ábra).



150. sz. ábra
Alakzatok megdöntése raszteres képernyőn

Lehetőség van a teljes rajzelem vagy annak egy kijelölt részének tükrözésére és 90° többszörösével való elforgatására is (lásd 151. sz. ábra).



151. sz. ábra
Alakzatok forgatása raszteres képernyőn

A professzionális jellegű grafikus programcsomagok az előzőeken kívül még jónéhány funkcionális lehetőséget biztosítanak a felhasználó számára.

Ezek közül néhány:

- A nyomdai előkészítéshez nélkülözhetetlen color separation (színrebontás) során a CMYK színtér színeinek arányát határozhatjuk meg. Az egyes színek intenzitását a nyomtatón a szürke megfelelő árnyalataiként jeleníthetjük meg. Ez a nyomdában az egyes színnyomatok elkészítését segíti.
- A képkezelést többszintűen végezhetjük: a grafikus objektumokat mint rétegeket (layers) gyakorlatilag korlátlan mennyiségben egymásra helyezhetjük. A képek megfelelő szintre való küldésével a rétegek között tetszőleges sorrendet alakíthatunk ki, de mindig csak a legelső (legfelső) olyan réteg látszik, amely nem átlátszó (és az alatta lévő rétegből a nagyságának megfelelő méretű területet takar el).
- Selection (kiválasztás): segítségével a kívánt képrészleteket kiválaszthatjuk. Az ezt követő módosítások már csak ezekre a képrészletekre vonatkoznak.
- Floating selection (lebegő kiválasztás): A lebegő kiválasztással kijelölt terület, az alatta lévő területek felett, önálló rétegként helyezkedik el. Az alatta lévő területekre egészen addig nincsen hatással, míg a kiválasztást nem szüntetjük meg.

Crop (képek körülvágása): Ezzel a funkcióval tetszőleges területet vághatunk ki a képből.

Dithering (árnyalás): Az árnyalás segítségével két szín közötti átmeneteket adhatunk meg.

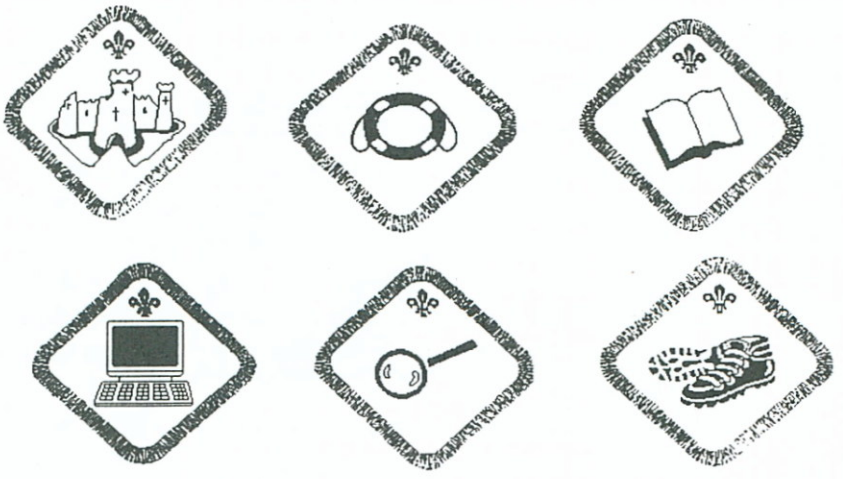
Feather Edge (lágy perem): E funkció két egymással határos, de színben jelentősen eltérő felület közötti kontraszthatásnak a csökkentésére alkalmas: a határoló éltől mindkét objektum irányában néhány pixelnyi távolságban fokozatos színátmenetet biztosít.

5.2. CLIPART, SZTEREOGRAMM, MORPHING

E fejezetben néhány speciális rasztergrafikus képtípust mutatunk be vázlatosan. Ezek a clipartok, az animált gif-ek és a sztereogrammok. Röviden foglalkozunk a moprhing-al, mely eljárás raszteres képek folytonos egymásba való átalakítását teszi lehetővé. A morphing-ot különösen a 3D-s grafikával készített filmek különböző trükkjeleneteinél alkalmazzák előszeretettel.

5.2.1. Clipartok

Clipartoknak nevezzük azokat a kisméretű képeket, melyeket általában szimbólumonként, emblémaként, logoként vagy egyszerűen csak díszítőelemként alkalmazunk egy képen. Ezek formai megjelenésüket tekintve lehetnek sematikus rajzok (lásd a 152. sz. ábra), rasztergrafikával generált kis képek vagy beszkennelt (esetleg átalakított) képek.



152. sz. ábra
Clipartok sematikus rajzok formájában

A clipartokat nagyon gyakran egyszerű animációs eljárással készítjük el, ekkor a kép egyes elemei rajzfilmszerűen mozognak. Egy forgó földgömbre mutat példát a 153. sz. ábra.

153. sz. ábra
Forgó földgömb clipartként

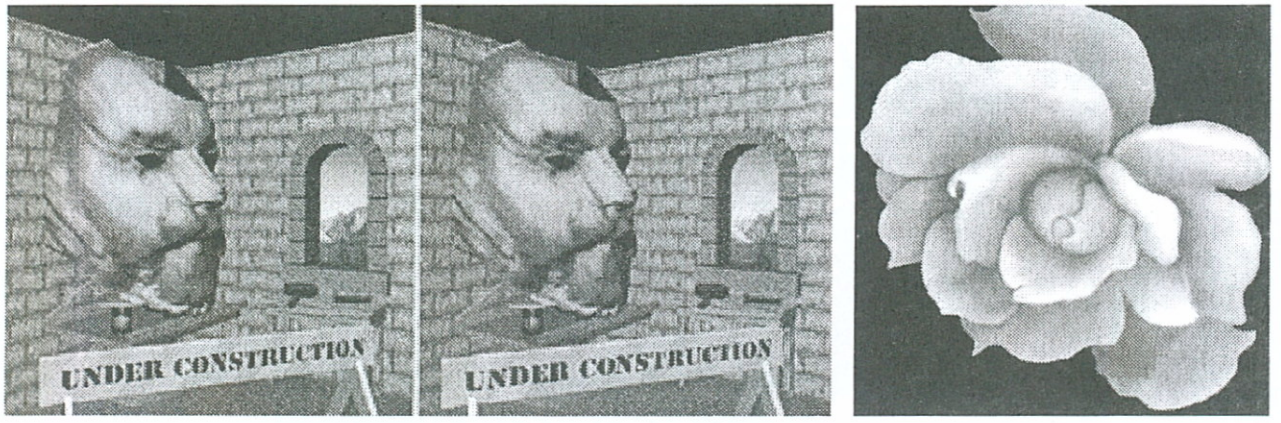
Az animációval készített clipartok az eljárás technológiáját tekintve legtöbbször animált gif-ek (lásd 5.4.4. fejezet).

A clipartok egyik legjellemzőbb alkalmazási területét a WEB lapok szerkesztése jelenti. Például, aki már szörfölt az Interneten, gyakran találkozhatott az „UNDER CONSTRUCTION" felirattal és a szorgalmasan „lapátoló" figurát ábrázoló animációval.

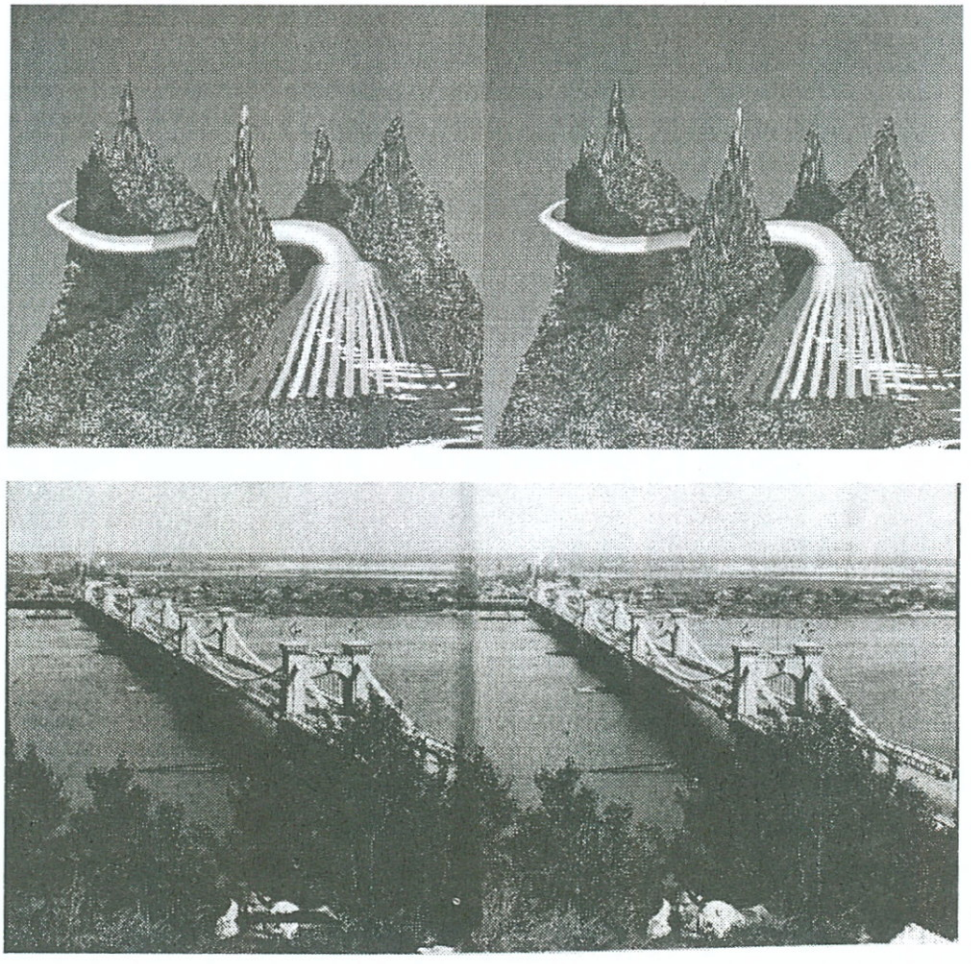
5.2.2. Sztereogrammok

! *A sztereogrammok olyan 2D-s képek, melyek a nézőben olyan érzést keltenek, mintha 3D-s képet nézne.*

Feltehetjük a kérdést: ezt a térhatást egy kép milyen módon érheti el? Régebbi elméletek szerint a képek nézése során a korábban már megismert térbeli alakzatokra emlékezünk. Ezért a képek perspektivikus ábrázolása, árnyékolása, a takart testek látványa a térhatás élményét kelti a nézőben. Erre mutatnak példákat a következő ábrák:

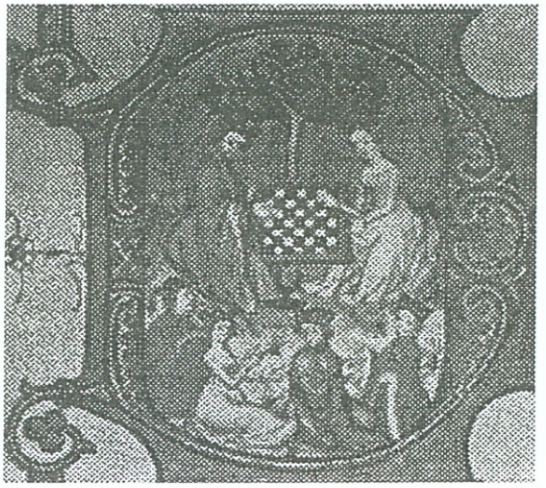


154. sz. ábra
Térhatás elérése takarással és árnyékolással



155. sz. ábra
Térhatás elérése perspektivikus ábrázolással

A térhatást azonban más eszközökkel is elérhetjük, erre a képzőművészetben találhatónk példákat (lásd a 156. sz. ábrán a XV. századból származó miniatúrát). E kép esetén a térhatás részben az alakok, formák kitakarásából, részben a háttér felől az előtérbe folyamatos tónusátmenet révén simuló alapsík által, részben pedig a vizuális mező kicsiny, környezetétől elszigetelt „kulcslyukhatása" révén jön létre.

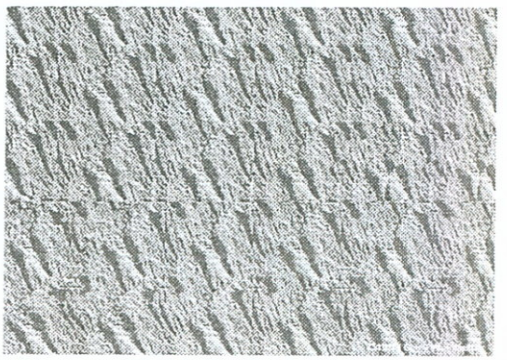


156. sz. ábra

A periodikus, egyszerű eltolásos (transzlációs) szimmetriát mutató mintázatot (lásd a 157. sz. ábra) pl. a terítőn, kétdimenzióként érzékeljük.

Térlátásunk új biológiai elmélete szerint ha a kép mikroszerkezetében ez a periodicitás sérül, akkor agyunk ebből a térbeli motívumot konstruálj (lásd a 158. sz. ábra).

157. sz. ábra
Egyszerű eltolásos szimmetria esetén a képet 2D-snak érzékeljük

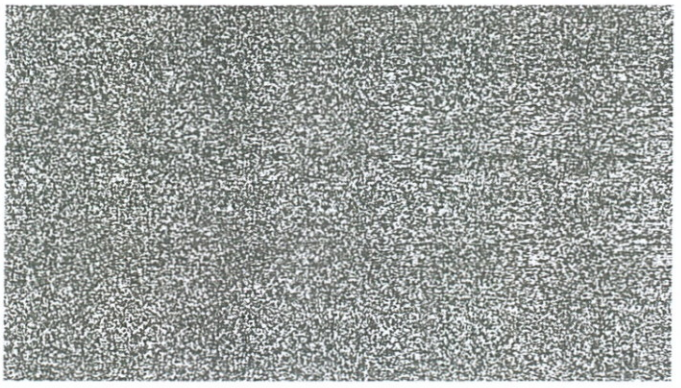


158. sz. ábra
A nem teljesen periodikus mikroszerkezetű kép térhatású

Julesz Béla a látáskutatás nagynevű, az Egyesült Államokban élő tudósa, még 1960-ban megmutatta, hogy a két szemünkben keletkező két retinális kép kicsiny különbségei már önmagukban is elegendők a térlátáshoz. Korábban ugyanis úgy vélték, hogy az alakok, árnyékok, textúrák, takarások feldolgozása agyunkban megelőzi a térbeli kép kialakulását. Julesz

frappáns ellenpéldája a véletlenszerű ponthalmazból konstruált sztereogram volt: ezen sem alak, sem árnyék, sem textúra, sem takarás nincs, mégis térben látjuk az elrejtett motívumokat a retinális képek különbözősége miatt.

Julesz random sztereogramja a kétdimenziós ábrán „valódi" háromdimenziós illúziót ad (lásd 159. sz. ábra).



159. sz. ábra
Julesz Béla féle random sztereogram

5.2.3. Morphing

Morphing alatt a számítógépes grafikában azt a transzformációt értjük, melynek során egy kép alakját folytonosan változtatva „átfolyik" egy másikba.

!

Jellemző példa erre a filmek világából a Terminátor II. „megtestesülése" a kövezetből.

160. sz. ábra
Morphing alkalmazása a Terminátor II. filmben

Ezek az algoritmusok lényegüket tekintve úgy működnek, hogy egy munkahálót feszítünk fel a forrás és a célképre, ezáltal kijelöljük, hogy melyik képrész melyik képrészbe transzformálódjon.

5.3. A RASZTERGRAFIKA LEGFONTOSABB ALGORITMUSAI

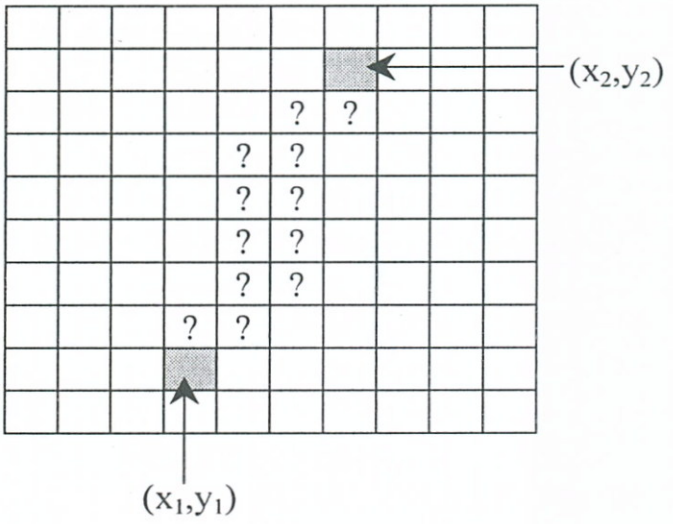
Az elkövetkezendőkben azokat a fontosabb algoritmusokat tekintjük át, melyekkel raszteres képen különböző alakzatokat képpontokkal lehet közelíteni.

5.3.1. Modellezés 2D-s egészkoordinátás rendszerben (Bresenham-algoritmus)

A rasztergrafika modelltere a 2D-s egészkoordinátákból „felépülő" koordináta-rendszer. Ezt négyzetes hálóval szokás szemléltetni, ahol egy négyzet egy raszterpontnak felel meg. A modellezés során arra a kérdésre keressük a választ, hogy hogyan lehet folytonos geometriai alakzatokat képpontokkal közelíteni. Ennél az a legfontosabb szempont, hogy a raszteres képernyőn megjelenített ábra esztétikus legyen, és a kép hatását tekintve – megfelelő felbontású eszköz esetén – megfeleljen a geometriában megszokottaknak.

A probléma érzékeltetésére vizsgáljuk meg a következő példát. Kapjon a grafikus képernyőt kezelő eszközszintű szoftver (vagy egyre jellemzőbben a monitorvezérlőkártya hardvere) egy olyan parancsot, hogy a képernyő (x_1,y_1) koordinátákkal jellemzett képpontjából egy egyenest kell húzni az (x_2,y_2) koordinátájú ponthoz.

Ekkor arra a kérdésre kell választ keresni, hogy melyik képpontok tartozzanak ehhez az egyeneshez (lásd 161. sz. ábra).



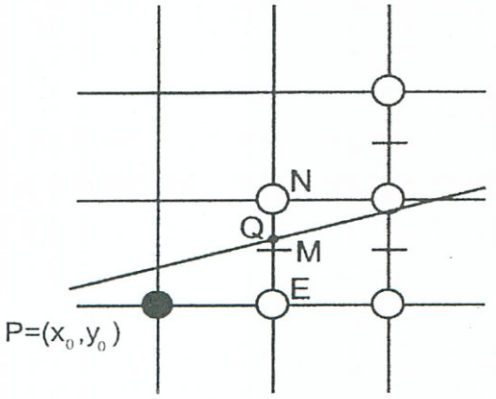
161. sz. ábra
Két pixelt összekötő pixelek meghatározása

Azokat az algoritmusokat, melyek kérdésünkre megadják a megfelelő eljárást, digitális differencia elemző (DDA) algoritmusoknak nevezik.

Ezek az algoritmusok a raszteres képeket reprezentáló 2D-s egészkoordinátás rendszerben működnek, ahol a képpontoknak a koordináta-rendszer rácspontjait feleltetjük meg.

Az egyik legismertebb algoritmus *a Bresenham-féle középpontos vonal-algoritmus*. Ezt az eljárást az egyenes példáján mutatjuk be, de alkalmazható görbe vonalak rajzolására is. *Ennek lényege, hogy a raszteres képen „oszlopirányban" haladva minden egész értékű „x"-pontban a matematikai egyeneshez függőlegesen legközelebbi pontot választjuk.*

Tegyük fel, hogy a 162. sz. ábrán látható egyeneshez a $P(x_0,y_0)$ pontot már kiválasztottuk az egyenes közelítő pontjaként (fekete kör). Feladatunk az algoritmus alapján eldönteni, hogy a következő lépésben az E vagy az N pontot kell-e választani közelítő pontként. Az ábrán az M pont az $\overline{N,E}$ szakasz felező pontját jelöli, Q pedig az egyenes és az $\overline{N,E}$ szakasz metszéspontja. (Felhívjuk a figyelmet arra, hogy mind az M, mind a Q pont a 2D-s valós koordináta-rendszerben értelmezett és nem a képernyő egészértékű koordináta-rendszerében.)



162. sz. ábra

Eljárásunk alapgondolata a következő:

- ha a Q pont az $\overline{N,E}$ szakaszon van, akkor N-t választjuk közelítő pontként,
- ha Q az $\overline{ME}$ szakasz része, akkor pedig az E-t választjuk közelítő pontként.

Legyen most

$$F(x,y) = ax + by + c = 0$$

az egyenes egyenlete.

Amennyiben az egyenes két rácspontot köt össze, akkor *a*, *b*, *c* egész számok (vagy az egyenlet szorzásával azzá tehetők).

A koordinátageometriából ismeretes, hogy a síknak azon pontjaira, melyek

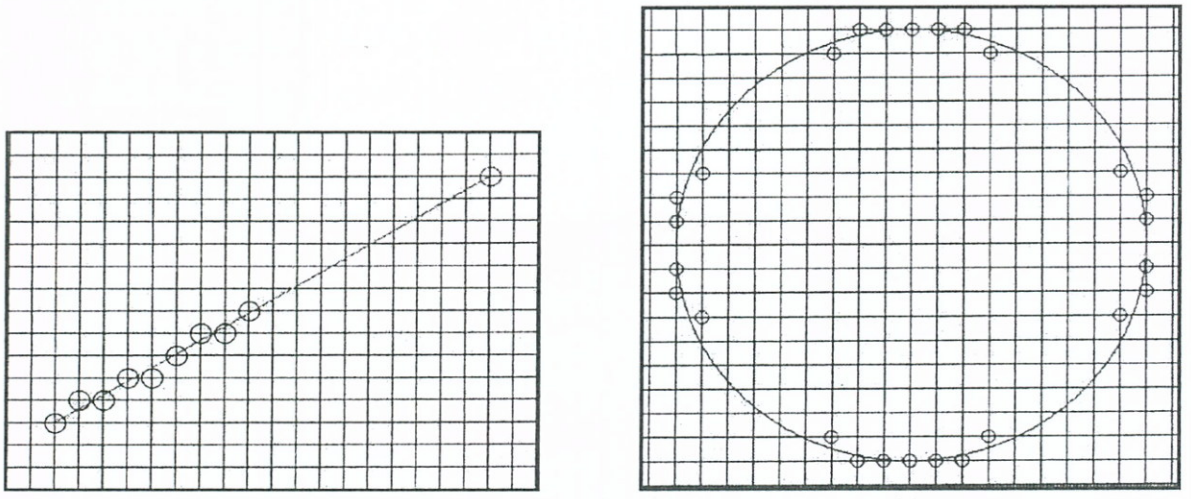
- rajta vannak az egyenesen $F(x,y) = 0$
- az egyenes egyik oldalán vannak $F(x,y) > 0$ és
- az egyenes másik oldalán vannak $F(x,y) < 0$.

Ezt a tulajdonságot használhatjuk fel az algoritmusban a helyes döntés alternatíváinak tesztelése során.

Az M pont koordinátái nyilván $(x_0 + 1, y_0 + \frac{1}{2})$ és ha ezt a pontot behelyettesítjük az $F(x,y)$ egyenletbe – attól függően, hogy negatív vagy pozitív értéket kapunk – el tudjuk dönteni, hogy az E vagy az N pont választása az optimális. (Vegyük észre, hogy ha a behelyettesítéskor az egyenletet 2-vel is megszorozzuk, az egész eljáráshoz csak egészértékű aritmetikai műveleteket kell használnunk.)

A Bresenham-algoritmussal való egyenes rajzolást raszteres képen mutatja be szimbolikus formában a 163. a) sz. ábra.

A Bresenham-algoritmus görbék közelítésére is alkalmazható. A kör közelítő raszterpontjait láthatjuk a 163. b) sz. ábrán.

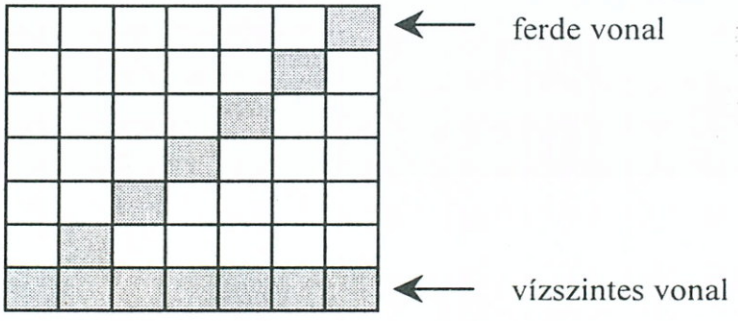


163. sz. ábra
Egyenes (a) és kör (b) közelítése a Bresenham-algoritmussal

5.3.2. Anti-aliasing

Az eddigiek során azzal foglalkoztunk, hogy egy 2D-s geometriai alakzatot nem tudunk általában kirajzolni olyan képpontokkal, melyek pontosan az alakzaton helyezkednek el és ezért közelítő megoldásokat kell alkalmaznunk.

Egy másik problémával találjuk magunkat szembe, *ha nemcsak vízszintes és függőleges vonalakat húzunk a képernyőn. Nézzük meg például egy ferde egyenes erősen kinagyított képét, amikor az úgynevezett lépcső effektus fellép* (lásd a 164. sz. ábra). Ezek a ferde vonalak már relatíve nem túl nagy felbontásnál is észrevehetők. !



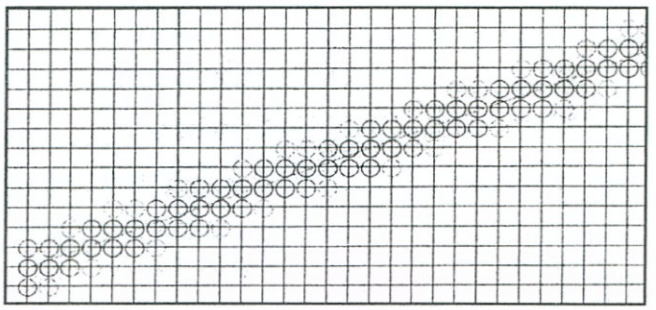
164. sz. ábra
A vonalak „lépcsőzetes" képe raszteres képernyőn

! *A raszteres egyenesszakaszok ábrázolásának további gondja, hogy a ferde és vízszintes vonalak fényereje is eltér egymástól.*

Ennek oka például a 164. sz. ábra példáján bemutatva, hogy

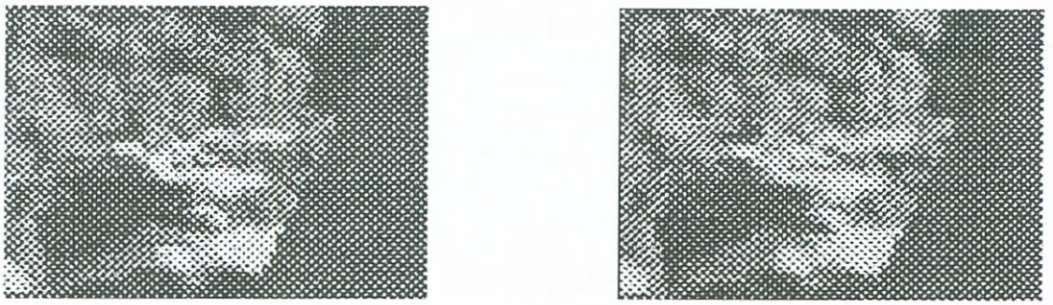
- a vízszintes vonal 7 egységnyi hosszú és 7 pixelt tartalmaz,
- a ferde vonal $7 \cdot \sqrt{2} \approx 10$ egységnyi hosszú és szintén 7 pixelt tartalmaz.

! E problémák kiküszöbölését célozza az *anti-aliasing eljárás, melynél a vonal melletti és a vonal szélein lévő pixelek színét átlagolják, és ezzel a vonalat tulajdonképpen egy „téglalappal" közelítik.*



165. sz. ábra
Anti-aliasing egyenes ábrázolása

Az anti-aliasing-nak a teljes képre gyakorolt hatását a 166. sz. ábra mutatja.



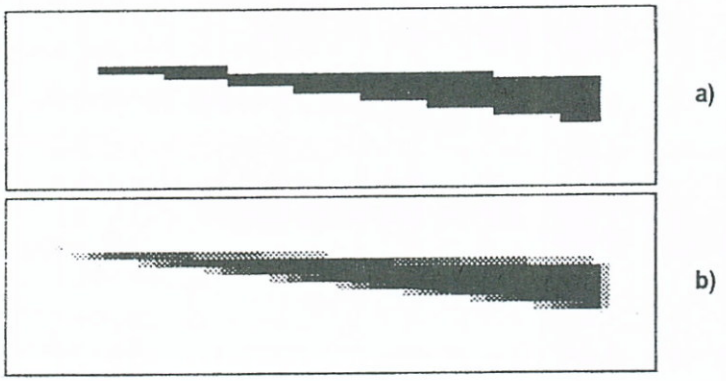
166. sz. ábra
Az anti-aliasing hatása a teljes képre

5.3.3. Super-sampling

! *Az élsimításnak egy másik, nagyon gyakran használt eljárása az úgynevezett super-sampling. Ennek lényege, hogy az éleken elhelyezkedő pixeleket felbontják 4x4=16 db további részre, melyet subpixelnek nevezünk.*

Ezek természetesen csak az algoritmusban léteznek, nem valódi képpontok. *Ez lényegében azt jelenti, hogy a raszteres képpontokat elvileg egy nagyobb felbontásnak megfelelően számítjuk ki.*

Egy megjelenített pixel színe vagy szürkeségértéke ezt követően a hozzá tartozó részpixelekhez rendelt értékek összeadásával kerül kiszámításra (lásd 167. sz. ábra).



167. sz. ábra
A super-sampling eljárás élsimítási hatása
a) simítás nélkül; b) super-sampling eljárással

5.4. A KÉPFELDOLGOZÁS ÉS TÖMÖRÍTÉS ALAPJAI

E fejezetben a számítógépes képfeldolgozással kapcsolatos legfontosabb fogalmakat és eljárásokat foglaljuk össze. Így tárgyaljuk a számítógépes rendszerekbe történő képbevitel, javítás és felismerés alapfogalmait, a képi redundanciákat és megszüntetésükre alkalmazható képtömörítési eljárásokat, a leggyakoribb képfájl-típusokat és ezek konvertálását.

Következetes szakmai terminológiát használva e fejezet mondanivalójának döntő része a digitális képfeldolgozás tárgykörébe esik. A gyakorlati munkában természetesen a számítógépes grafika és a digitális képfeldolgozás ennyire élesen nem választható szét. Erre két rövid példa:

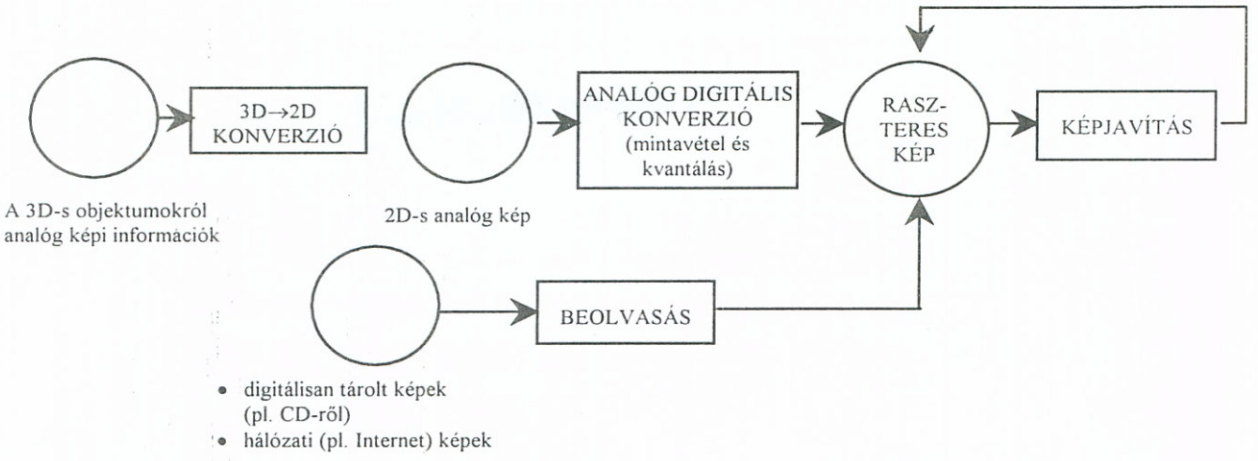
- Ha egy vektorgrafikus rendszerrel, például COREL DRAW-al előállítunk egy reklámgrafikát, akkor a következő kérdéssel is találkozhatunk: milyen képfájl-formátumban adjuk át a grafikát nyomdai sokszorosításra.
- Ha egy ház tervrajzával elkészülünk egy vektorgrafikus rendszerben, és ezt egy fotominőségű képen valós természeti környezetben be akarjuk bemutatni a megrendelőnek, akkor valamilyen formában

„egyesíteni" kell a vektorgrafikában renderelt képet a környezet fényképével.

Mindezek a példák is indokolják, hogy a képfeldolgozással kapcsolatos alapismereteket e fejezetben röviden összefoglaljuk.

5.4.1. Képbevitel a számítógépes rendszerbe

Elsőként a képi információk számítógépes rendszerbe történő bevitelének lépéseit vizsgáljuk meg, melyet a 168. sz. ábra szemléltet.



168. sz. ábra
A képi információk számítógépes rendszerbe történő bevitelének lépései

A valós világ objektumai háromdimenziósak a fényt önállóan kisugározzák vagy visszaverik. Ha ezt valamilyen érzékelő eszközzel (például TV kamerával) rögzíteni kívánjuk elsőként a 3D-s objektumokról kapott analóg képi információt két dimenziós analóg információvá kell átalakítani. Ahhoz, hogy számítógéppel feldolgozható digitális információt kapjunk, a következő lépésben egy *analóg-digitális konverziót* kell végrehajtanunk. Ennek *első lépése a mintavételezés. Ezt úgy képzelhetjük el, hogy például az analóg képi információkat hordozó téglalap alakú képre egy kisméretű négyzetekből álló hálót borítunk és minden négyzetben található folytonos képi információt valamilyen módszerrel átlagoljuk. Ezt követően a négyzetben lévő analóg értékeket ezzel az átlagértékkel helyettesítjük.*

Ez az átlagérték egy meghatározott maximum és minimum tartományon belül gyakorlatilag tetszőleges értékeket fel vehet. Ezért a *digitalizálás utolsó lépéseként – amit kvantálásnak neveznek – a maximum és minimum tartományt intervallumokra osztjuk fel, és ha az átlagérték egy intervallumba*

esik, akkor a fényinformációt ennek az intervallumnak a bináris kódjával helyettesítjük. Ennek az eljárásnak az eredményeként egy raszteres képet kapunk, melyet közvetlenül feldolgozhatunk a rasztergrafikában. Az előbbiekben leírt módon működnek a raszteres képbeviteli eszközök például a szkenner vagy a digitális kamera.

5.4.2. A képjavítás eljárásai

A képfelvétel során és a digitalizáló berendezés minőségi paraméterei miatt torzulások, zajok léphetnek fel. A negatív hatásokat képjavító eljárásokkal próbálhatjuk meg csökkenteni.

A képjavító eljárások egyik fontos eljárása a világosságkód transzformáció, mellyel a nem megfelelő megvilágításból és az érzékelő eszköz műszaki korlátaiból adódó fényveszteségből fakadó kontrasztszegénységet próbálhatjuk meg kiküszöbölni.

Így például a kontrasztkiemelés egyik módszere, hogy a világosságkódok részhalmazait a transzformáció során a 169 sz. ábra szerint egy kóddal helyettesítjük. Ezáltal a világosságkódok lassú átmenetét felgyorsítják, ezáltal az árnyalatok átmenete élesebb lesz.

169. sz. ábra
Kontrasztkiemelés világosság-kódtranszformációval

A zajelnyomáshoz a képpontok véletlenszerűen megváltozásából adódó hibákat is ki kell küszöbölni. Ezt elérhetjük például azáltal, hogy minden egyes képponthoz rendelt értéket a pont és környezete átlagolt értékével helyettesítünk.

Az élkiemeléssel a képen található objektumok határainak elmosódottságát küszöbölhetjük ki. A világosságkódokban meglévő „ugrások" élkiemelő eljárásokkal való erősítése az emberi alakfelismerés szempontjából is előnyös, mivel ez jórészt a kontúrlátáson alapul.

5.4.3. Szegmentálás és osztályozás, alakfelismerés

Szegmentálásnak nevezzük mindazokat az eljárásokat, melyek lehetővé teszik, hogy a számítógépes algoritmusokkal a képen lévő fontos objektumokat a kevésbe fontosaktól (például háttér) elkülönítsük.

Ezeknek az eljárásoknak az elvi modellje a következő:

- az objektumok lényeges tulajdonságait úgynevezett sajátságvektorokkal fejezzük ki,
- döntésfüggvényeket alkotunk és ezzel kiértékeljük a képpontok sajátságvektorait és így meghatározzuk, hogy egy képpontot melyik objektumhoz kell besorolni.

A szegmentálás két alapesete a foltkeresés és az élkeresés.

A foltkeresésnél a szomszédos, a döntésfüggvény szerint hasonló képpontokból képezzük a foltokat. Az éleket ott keressük, ahol a sajátságvektor nagyot változik. (Ezek általában a foltok határoló görbéi)

A képen lévő alakzatok osztályozásának célja a kép (illetve részeinek) számítógépes felismerése. Ehhez úgynevezett alakfelismerési algoritmusok szükségesek.

Ezek két fő típusba sorolhatók:

- Mintaalakzatokhoz való hasonlóság eldöntését segítő algoritmusok
- Klaszterezési algoritmusok, melyek a képelemek osztályba-sorolását valószínűségszámítási módszerekkel végzik el.

Speciális osztályozási lehetőséget ad a textúraelemzés, mely lényegét tekintve a kép mikrostruktúrájának vizsgálatát jelenti. A mikrostruktúrát vagy textúrát az elemzésekkel a következő kategóriákba sorolhatjuk:

- ismétlődő vagy azonos mintázat;
- statisztikusan ismétlődő, azaz lényegét tekintve azonos mintázat (például fraktál);
- véletlenszerűen eltérő mintázat (a textúrában nem lelhető fel szabályszerűség).

Ennek eredményét szintén fel lehet használni a képen lévő objektumok azonosítására.

5.4.4. Képtömörítés

A számítógépes képek tömörítését a redundanciák teszik lehetővé és szükségessé.

Egy közleményben az alkalmazott jelsorozat egyes elemei (pl. a kódkészlet egyes kódjai) $A_1, A_2, \ldots A_{N-1}, A_N$ meghatározott $P_1, P_2, \ldots P_{N-1}, P_N$ valószínűséggel fordulnak elő. Ezek segítségével definiálhatjuk egy A_i jel egyedi információtartalmát:

$$I(A_i) = -LOG_2 P_i$$

Ezek szerint minél ritkábban fordul elő egy jel (azaz kisebb a valószínűsége), az általa hordozott információ tartalom annál nagyobb.

Az átlagos információtartalom – vagy más néven az entrópia – összehasonlíthatóvá teszi egy adott jelkészlet átlagos információtartalmát egy másik jelkészletével. Az entrópiát a következő képlettel definiálhatjuk:

$$E = -\sum_{i=1}^{n} P_i \cdot LOG_2 P_i$$

Az entrópia akkor a maximális, ha $P_1 = P_2 = \ldots = P_N$, azaz az összes jel előfordulási valószínűsége azonos.

Most már rátérhetünk a számítógépes képfeldolgozás egyik legfontosabb fogalmának, a redundanciának tárgyalására. A redundanciát a következő képlettel definiáljuk:

$$R = (E_{MAX} - E) / E_{MAX}$$

Itt E_{MAX} az adott n elemű jelkészlethez tartozó entrópiák közül a maximális értékűt jelöli.

Kissé leegyszerűsítve, úgy értelmezhetjük a redundancia fogalmát, hogy egy közleménynek akkor magas a redundanciája, ha „megértéséhez" minimálisan szükséges információt kifejező jeleken túlmenően sok felesleges jelet is tartalmaz.

A képfeldolgozásban a következő redundanciákkal találkozhatunk:

- *A kódolási redundancia lép fel akkor, amikor például a fekete-fehér képpontok ábrázolásához 8 bitet azaz 1 byte-ot használunk fel,*
- *Képi redundanciát jelent például:*
  - *ha nagy kiterjedésű azonos színű objektumok minden egyes képpontját tároljuk a geometriai jellemzők helyett,*
  - *a mozgóképeknél és az animációknál jelentős redundancia léphet fel akkor, ha az egymást követő képek csak nagyon kis mértékben különböznek egymástól és* ahelyett, hogy csak az eltéréseket tárolnánk, *minden képet teljes körűen kódolunk és tárolunk,*
  - *ha a vektorosan is tárolható alakzatokat raszteresen tároljuk* (például egy háromszögnél elegendő a három csúcspont koordinátáit tárolni) és nem kell az összes pixel koordinátáét és színét).
- *Pszihovizuális redundanciát* jelent például, ha
  - *a képminőségbeli eltéréseket* (például sok színárnyalattal nagy felbontás) *az ember már nem tudja megkülönböztetni,*
  - *szubjektíve a képnek csak egy része érdekes számunkra, ennek ellenére a teljes képet tároljuk és feldolgozzuk.*

A számítógépes képfeldolgozás során a képeket egyrészt a tárolóigény csökkentése miatt, másrészt a képek adatátviteléhez szükséges idők csökkentése miatt kell tömörítenünk. A képtömörítési eljárások ezen túlmenően az adatátviteli hibák kiküszöbölését és az adatok titkosítását is szolgálhatják.

5.4.4.1. Veszteségmentes tömörítés

! *A veszteségmentes képtömörítés a kép összes információját megőrzi. Ez úgy lehetséges, hogy ezek az eljárások csak a kódolási és a képi redundanciát szüntetik meg.*

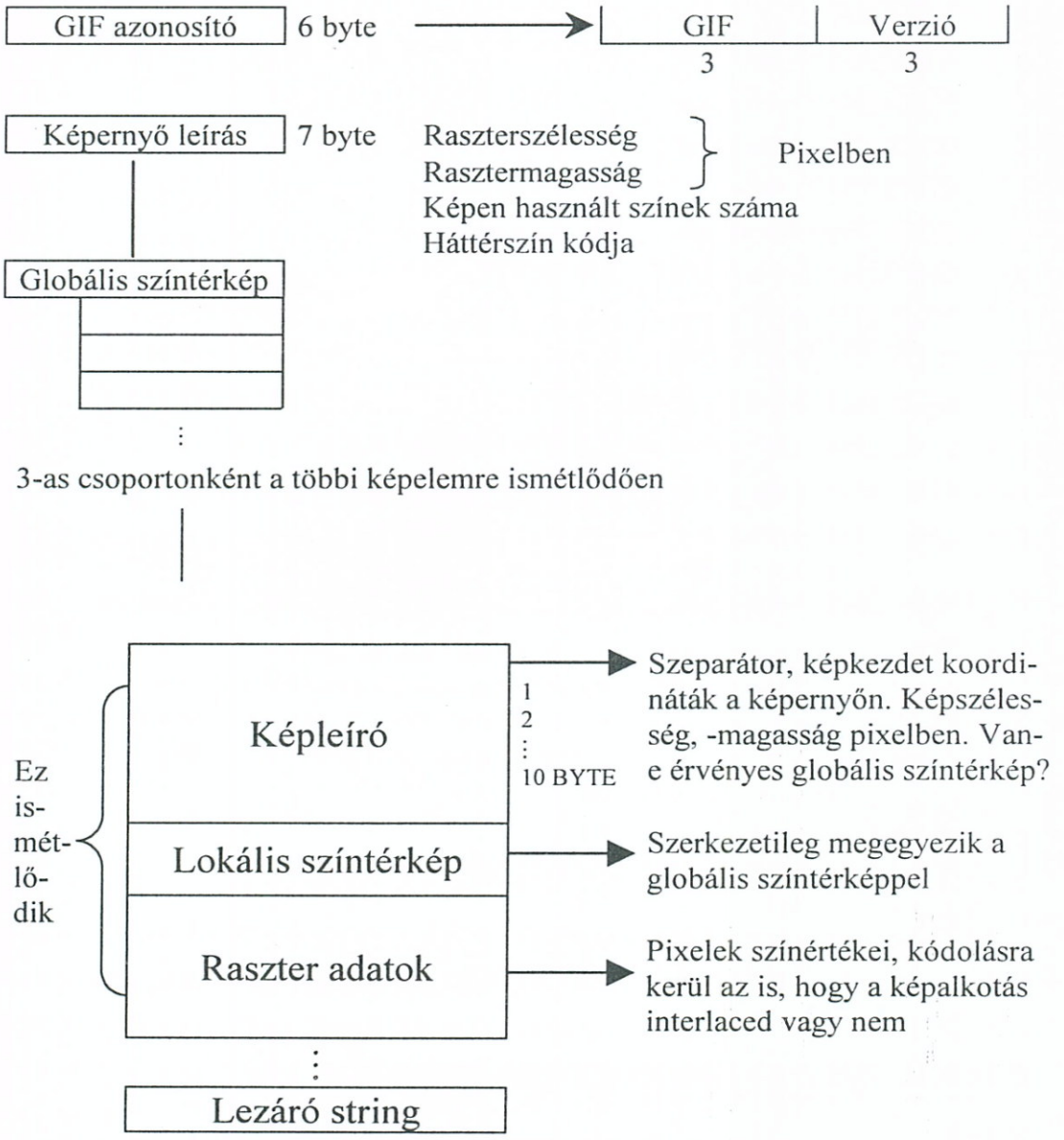
Erre a következő példákat lehet megemlíteni:

- a Huffmann-kódolásnál a nagyon gyakori adatokat kódoljuk a legkevesebb bit-számmal. Így például, ha a magyar nyelvű szövegeket kell kódolnunk a leggyakoribb magánhangzó „E" betűt csak egy bittel jellemezzük.
- A képen azonos jellemzőkkel bíró homogén foltokat kódolhatjuk a geometriai adatokkal és 1 pixel jellemzőivel, ahelyett hogy az összes pixelről tárolnánk az összes információt.
- A mozgóképeknél jelentős tömörítést érhetünk el azzal, ha nem a teljes kép adatait, hanem csak a képpont értékek megváltozott adatait tároljuk.

! *A raszteres képeknek az Interneten általánosan elterjedt veszteségmentes képtömörítő eljárása a GIF (Graphics Interchange Format). Ez kvázi szabvánnyá a compuserve hálózat által vált. A GIF tömörítésű fájlok kiterjesztése .gif.*

A GIF formátum kezdetben csak a 8-bites színkódolást tett lehetővé, de újabban kezelni képes a 24 bites színkódolást is.

A GIF adatszerkezetét a 170. sz. ábra mutatja be:



170. sz. ábra

5.4.4.2. Veszteséges tömörítés

A veszteséges tömörítés a képek pszihovizuális redundanciáját használja ki. A tömörítés mértéke aszerint változhat, hogy a képminőségnek milyen követelményeket kell kielégítenie. A két szélső eset:

- az emberi érzékeléssel ne lehessen észre venni a minőségromlást (ez átlagosan 1:20-tól 1:40 arányú tömörítést tesz lehetővé)

- *csak azt kívánjuk, hogy a kép lényeges objektumai felismerhetők legyenek* (ekkor kb. 1:200 arányú tömörítés is elérhetünk).

A veszteséges tömörítés legfontosabb algoritmusa a DFT, azaz a Diszkrét Fourier Transzformáció. (Ezt szokták DCT-nek, azaz diszkrét koszinusz transzformációnak is nevezni.) Ezt az algoritmust alkalmazza *az Interneten legelterjedtebb JPEG nevű veszteséges tömörítés, mely* a Joint Photographic Experts Group rövidítésből származik. Az eljárás *ISO és CCITT szabvány is. A JPEG tömörítésű fájlok kiterjesztése .jpg.*

Jellemzői :

- *a tömörítéssel átlagosan 1:30-as arányt érhetünk el, de ez az eljárást megvalósító szoftverekben paraméterezhető* (tehát a minőségromlás árán beállíthatunk nagyobb arányú tömörítést is).
- A tömörített raszteres képeket 24 bites színkódolással is kezelni képes az algoritmus.

Az algoritmus lényege a következőkben foglalható össze:

- RGB színtérből a képeket áttranszformáljuk YUV színrendszerbe, mely színek alkalmazása esetén pszichovizuális kísérletek bizonyították, hogy az ember kevésbé érzékeny a kép információveszteségére.
- Alkalmazzuk a DFT transzformációt. (Ez azért előnyös, mert a szomszédos képpontok színeltérése általában minimális, így a frekvenciatartományba transzformálva a képet, ez a kis változás miatt kis frekvenciájú komponenseket tartalmaz.)
- Alkalmazzuk a transzformált képre a gyakoriság szerinti Huffmann-kódolást.

A fraktáltömörítés alapgondolata, hogy a természetben előforduló képek általában önhasonló mintákat tartalmaznak.

Az algoritmus lényege a következőkben foglalható össze:

A kép egyes részeit elemi alakzatokkal, fraktálokkal közelítjük, s csak az ezeket a képeket előállító függvények együtthatóit tárolják. Az adatvesztés ebben az esetben csak a közelítés pontatlanságából adódik. Az alkalmazott módszernek köszönhetően viszont az eredményül kapott kép felbontásfüggetlen lesz.

Mivel várhatóan a fraktáltömörítés a jövőben jelentősen el fog terjedni, fontosabb jellemzőit összefoglaljuk:

- *A tömörítésnek jelentős a számításigénye.* A tömörített kép kifejtése ennél lényegesen gyorsabb.
- *A fraktáltömörítéssel lényegesebben nagyobb tömörítést lehet elérni, mint pl. a JPEG-gel (1:100)*
- *A fraktáltömörítésnek jobb a kontúr és színárnyalat visszaadása a korábban ismert tömörítő algoritmusokhoz képest.*

A fraktáltömörítésű fájlokat a .fif kiterjesztésről ismerhetjük fel.

5.4.5. Képfájl típusok és konvertálásuk

A képi információkat képfájlokban tároljuk. Ezek lehetnek

- Általánosan elterjedt raszteres vagy vektoros fájl típusok,
- Egy programcsomaghoz kötődő specifikus fájlok.

Az általánosan használt (szabványok vagy kvázi szabványok) raszteres képfájl típusok a következők:

BMP = *Windows bitmap*, független a grafikus kártyától és annak kezelőprogramjától, 24 bites szín-mélység kezelését biztosítja.

PCX = *A paintbrush festőprogram fájlformátuma* eredetileg, jelenleg az egyik legelterjedtebb formátum. Az RGB színteret a 24 - bites színmélységig is lehet kódolni (a CMYK-t nem tudja kezelni).

TIF = *Az ún. TIFF = Tagged Image File Format nagyon elterjedten használt, főleg DTP területén.* Fontos jellemzője az operációs rendszer és a hardver függetlenség.

JPG = *A veszteséges képtömörítési szabvány fájlformátum.*

FIF = *Veszteséges fraktáltömörítéses fájlformátum*, elérhető kb. 1:100 arány is minőségromlás nélkül.

GIF = *A WEB-en legelterjedtebb veszteségmentes tömörítésű fájlok formátuma.*

PDF = *Az Adobe cég által kifejlesztett fájlformátum* (Portable Document Format). Alapját Postscript lapleírónyelv képezi.

PCD = *A digitális fényképekre a Kodak által kifejlesztett szabvány.*

PNG = *Portable Network Graphics, a Compuserve hálózat fájltípusa.*

Az általánosan elterjedt vektoros (és esetenként a raszteres képeket is kezelő) fájltípusokat mutatjuk be a következőkben:

EPS (Encapsulated Postscript) = Raszteres és vektoros képek nyomdai munkákhoz való előkészítése során alkalmazhatjuk.

WMF = *Windows Metafile*

WRL = *VRML fájlok formátuma*

DXF = Huzalvázas vektoros (CAD) objektumokkal kapcsolatos adatcserére használatos (Drawing Exchange Format)

IGES = (Initial Graphics Exchange Specifications) vektoros fájlszabvány

Az elterjedt professzionális grafikus programcsomagok fájlformátumai a következők.

CADL = CADKEY programcsomag fájlformátuma,

! *CDR = Corel-Draw fájlformátuma,*

DWG = az AUTOCAD fájlformátuma,

3DS = a 3D Studio MAX fájformátuma,

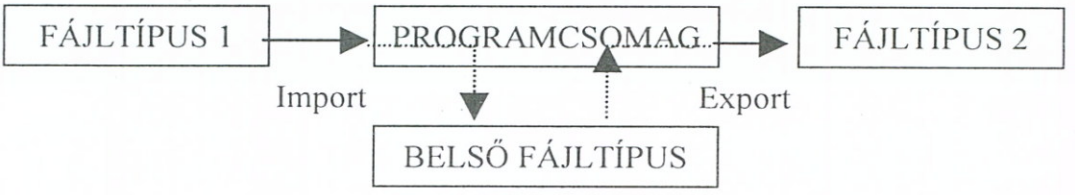
PSF = az ADOBE Photoshop fájlformátuma.

A különböző képfájl típusokat felhasználhatóságuk alapján is megkülönböztethetjük:

- Ez a fájltípusok egy része esetén a nevükből jórészt azonnal adódik. Például az EPS-t DTP célokra használhatjuk, és a DXF fájl formátumot célszerű alkalmazni CAD terveknek például INTERNET igénybevételével való megküldéséhez.
- A JPEG veszteséges tömörítésű fájlformátumot célszerű választani a foto minőségű képek tárolása során, ha a képeket emberek fogják nézni . (Ha egy képpel huzamosabban akarunk dolgozni, ne mentsük el többször JPEG-ben, mert a minősége jelentősen romlik.)

A számítógépes képfeldolgozás során sokszor szükségünk van az egyik rendszerben előállított képfájlnak egy másik típusra történő átkonvertálására. Ehhez az összes jelentősebb grafikai programcsomag tartalmaz fájl export-importot megvalósító eszközöket, amelyekkel a különböző fájltípusokat konvertálhatjuk.

A professzionális programcsomagok ún. szűrőket tartalmaznak, melyek elvégzik a fájl konverziót az egyik típusról a másikra. Ha képfájlokat szeretnénk konvertálni, akkor általában a 171. sz. ábrán látható módszert célszerű választanunk.



171. sz. ábra
Képfájl konverzió professzionális programcsomagokkal

ELLENŐRZŐ KÉRDÉSEK ?

202. *Miből épül fel a rasztere kép?*
203. *Mit értünk pixel alatt?*
204. *Milyen fajta raszteres képtípusok vannak?*
205. *Mit nevezünk palettának?*
206. *Mit jelent a TrueColor és hány színárnyalatot lehet kódolni ezzel a módszerrel?*
207. *Hogyan kereshetjük vissza és módosíthatjuk a rasztere képeket?*
208. *Mit értünk a rasztergrafikus primitívek alatt?*
209. Hogyan lehet primitíveket létrehozni?
210. Mit értünk terület-meghatározó primitív alatt?
211. *Milyen tulajdonságokat lehet hozzárendelni a rajzelemekhez?*
212. Milyen lehetőség van a területmeghatározó primitívek kitöltésére?
213. *Mit értünk DTP alatt?*
214. *Mi a feladata a kiadványszerkesztő programcsomagoknak?*
215. *Hogyan határozhatjuk meg rasztergrafikus rendszerekben a szövegek formáját?*
216. Mit értünk font alatt?
217. *Mit nevezünk betűcsaládnak?*
218. *Milyen vektoros és raszteres betűcsaládokat ismer?*
219. Hogyan modellezzük a TrueType fontokat?
220. Hogyan jelenítjük meg a vektoros és raszteres betűket?
221. *Hogyan kezeli a fontokat a WINDOWS operációs rendszer?*
222. *Miben különbözhetnek egy családon belül a betűtípusok?*
223. Mi a monitorvezérlő kártya részegységeinek feladata raszteres képek megjelenítése során?
224. *Mit nevezünk clipartnak és ezek mire használhatók?*
225. Mire használhatók az animált-gif-ek?
226. *Mit nevezünk sztereogrammnak?*
227. Hogyan kelthetünk térhatás érzetet a nézőben legegyszerűbben?
228. Mi Julesz Béla térhatás elméletének lényege?
229. *Mit értünk morphing alatt?*
230. Mi a morphing algoritmusok lényege?
231. Mire használhatók a DDA algoritmusok?
232. *Mi a Bresenham-algoritmus lényege?*
233. *Mit jelent a lépcső effektus és milyen fényerőproblémák jelentkeznek ferde vonalak képernyőn való megjelenítésekor?*
234. *Mi az anti-aliasing lényege?*
235. *Mi a super-sampling eljárás?*
236. Mit nevezünk mintavételezésnek?

237. Mit nevezünk kvantálásnak?
238. Milyen negatív hatásokat küszöbölnek ki a képjavító eljárások?
239. *Mit nevezünk szegmentálásnak?*
240. Mi a szegmentálási eljárás lényege?
241. *Mit nevezünk alakfelismerésnek?*
242. *Mikor magas a redundanciája egy közleménynek?*
243. *Mondjon példákat a kódolási, képi és pszihovizuális redundanciára?*
244. *Mi alapján lehetséges a veszteségmentes tömörítés?*
245. Milyen kódolási eljárásokkal lehet megszüntetni a kódolási és képi redundanciát?
246. *Mi a GIF?*
247. *Mi teszi lehetővé a veszteséges tömörítést?*
248. *Mi a JPEG és milyen tömörítés érhető el felhasználásával?*
249. Jellemezze a fraktártömörítést!
250. *Melyek a legfontosabb raszteres képfáljformátumok?*
251. *Soroljon fel néhány vektoros fájlformátumot!*
252. Hogyan lehet a grafikus fájlokat konvertálni a professzionális programcsomagokkal?

VI.

VEKTORGRAFIKA

E fejezetben a 3D-s vektorgrafika legfontosabb fogalmait, jellemző tulajdonságait és algoritmusait tekintjük át. A tárgyaltak egy része értelemszerűen vonatkozik a 2D-s vektorgrafikus (általában rajzoló, tervező) rendszerekre is, ahol a különbségek lényegesek, ott azokra külön utalni fogunk.

A vektorgrafika, mint azt az I. fejezetben már bemutattuk, a különböző vektorgrafikus objektumokat egy absztrakt világkoordináta-rendszerben, a modelltérben kezeli.

A felhasználó, amikor egy vektorgrafikus programcsomaggal dolgozik, a modelltérben

- önálló vektorgrafikus objektumokat (például testek, tárgyak modelljei) hozhat létre, illetve új objektumokat generálhat a meglévőkből (például egy kör elmozgatásával egy toruszt),
- meghatározhatja a vektorgrafikus objektumok viszonyát (például alá-fölérendeltség, csoportképzés), ezekből komplex jeleneteket (scene) állíthat össze,
- műveleteket végezhet a modelltér tárgyaival (például elmozgathatja ezeket a világkoordináta-rendszerben),
- fényforrásokat és kamerákat definiálhat, és
- a modelltér jeleneteit megjelenítheti raszteres kép formájában például a monitor képernyőjén.

E fejezetben először a vektorgrafika általános jellemzőit (a raszter és vektorgrafika különbsége, a vektorgrafikus adatbáziskezelés, a geometriai

modellezés és primitívek, a vektorgrafikus objektumokkal való műveletvégzés, a megjelenítés és a felhasználói felület) tárgyaljuk. Ezt követően – a gyakorlati munkavégzés lépései szerint – sorra vesszük a vektorgrafika különböző modellezési eljárásait, a legfontosabb látható kép meghatározási, megvilágítási és árnyékolási algoritmusait. Végezetül pedig a fotorealisztikus képábrázolás két legnépszerűbb algoritmusát (raytracing, radiosity) mutatjuk be valamivel részletesebben. Ennek során a legfontosabb fogalmak és eljárások lényegének a bemutatására törekszünk, nem térünk ki a programalgoritmus szintű részletkérdésekre.

6.1. A VEKTORGRAFIKA ÁLTALÁNOS JELLEMZŐI

Az első vektorgrafikus rendszerek a mérnöki tervezőmunkához szükséges műszaki rajzok interaktív elkészítését segítették. Kialakulásukat többek között a geometriai pontosságú szerkesztések számítógéppel történő elvégzésének igénye motiválta, melyet raszteres képen megfelelő színvonalon nem lehetett megvalósítani. A raszteres rajzolóprogramoknak a mérnöki munkában történő alkalmazásában az is problémát okozott, hogy egy raszteres rajzon ábrázolt alkatrészhez tartozó rajzelemeket önállóan nem lehetett a számítógépen tárolni és ismételten felhasználni.

Az első vektorgrafikus rajzolóprogramok vonalakból álló 2D-s síkidomok előállítását tették lehetővé. A korábbiakhoz képest már e rendszerek alkalmazása is minőségi különbséget jelentett, mivel a kétdimenziós rajzelemekkel tetszés szerinti geometriai transzformációkat (forgatás, tükrözés stb.) lehetett végrehajtani, és a vonalas rajzok adatbázisszerűen tárolásra és újrafelhasználásra kerülhettek.

A vektorgrafikus szoftverek alkalmazási területei – főként a hardver felgyorsult fejlődése (például a lebegőpontos processzorokkal rendelkező mikroszámítógépek tömeges elterjedése) miatt – az elmúlt több mint egy évtizedben erőteljesen bővültek.

6.1.1. A vektorgrafika és a rasztergrafika összehasonlítása

A vektorgrafika gyors fejlődését jórészt a rasztergrafika és a vektorgrafika különbségei magyarázzák:

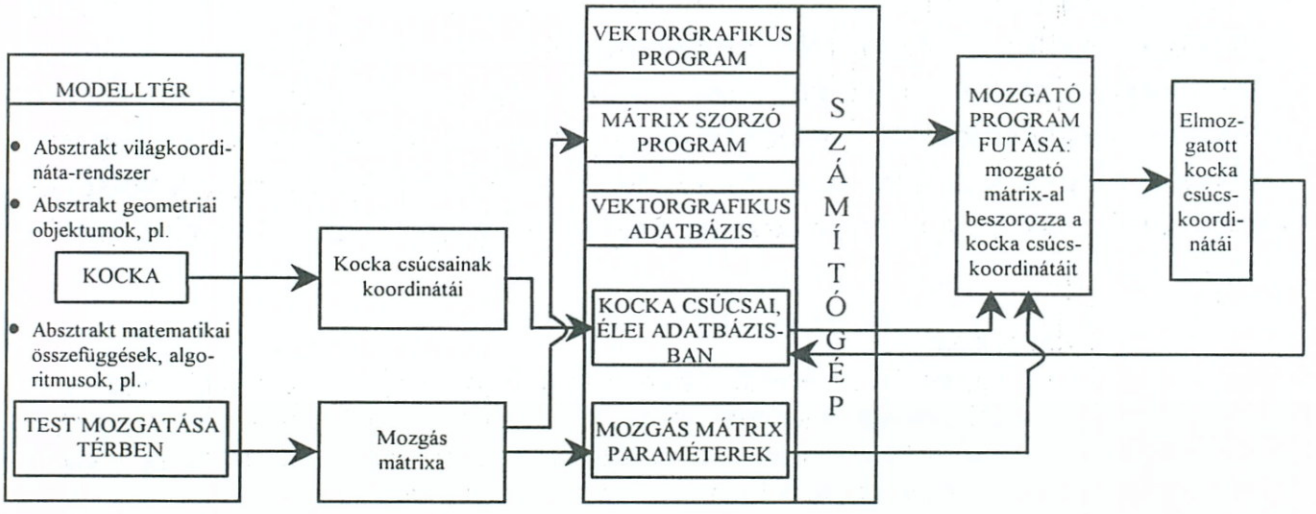
- *A vektorgrafika egy 3D-s (korábbi változatai 2D-s) lebegőpontos világ-koordináta-rendszert használ, ezáltal lehetővé teszi a geometriai pontosságú szerkesztést és transzformációkat* Így sokkal alkalmasabb a rasztergrafikánál a mérnöki és tudományos munka támogatására.

- *A vektorgrafika absztrakt modelltérbeli tárgyakkal dolgozik. Ezek önálló objektumok (entitások), melyekkel műveleteket lehet végezni a képernyőn való megjelenítéstől függetlenül is.* A felhasználói felület, a grafikus programcsomagok és a megjelenítő hardver szabványosítása ugyanakkor lényegében bármely elterjedt számítógépkonfiguráción lehetővé teszi a modelltérbeli vektorgrafikus objektumok raszteres képen való megjelenítését akár több nézőpontból (kameraállásból) is. *Ezzel szemben egy rasztergrafikus képet – lényegét tekintve – csak a kép felülírásával tudjuk módosítani.* !
- *A vektorgrafikában a grafikus objektumokat adatbázisban tárolják, mely lehetővé teszi az egyes testek, tárgyak modelljeinek egyedi visszakeresését és az ezek közötti kapcsolatok rögzítését és kimutathatóságát. A rasztergrafikában viszont nincs lehetőség az egyes képeken belüli grafikus rajzelemek önálló visszakeresésére és kezelésére.* !

6.1.2. A modelltér és a vektorgrafikus adatbázis

6.1.2.1. A modelltér és a vektorgrafikus adatbázis kapcsolata

A vektorgrafika modellterének és a vektorgrafikus rendszer és adatbázisának a viszonyát a 172. sz. ábra szemlélteti.



172. sz. ábra
A modelltér, a vektorgrafikus rendszer és adatbázis kapcsolata

Az ábra alapján is látható, hogy a vektorgrafika modelltere egy absztrakt világkoordináta-rendszer, melyben az absztrakt geometriai objektumokat kezeljük. Ez meghatározott matematikai törvényekkel és ezekből levezethető algoritmusokkal történik. Mindez természetesen a számítógépen fizikailag nem létezik, ezeket csak a vektorgrafikus programcsomag és adatbáziskezelő szoftverek fejlesztői használják fel. A modelltér absztrakt összefüggései alapján készülnek el a számítógépes rendszer programjai, ezek szerint töltjük fel a vektorgrafikus adatbázist. Példánkban, ha a modelltér egy kockáját a térben el akarjuk mozgatni, akkor a vektorgrafikus rendszernek egy olyan programegysége aktivizálódik, mely a mozgásnak megfelelő mátrixszorzó algoritmust végrehajtja és ennek megfelelően az elmozgatott kocka koordinátáival felülírja a kockának megfelelő adatbázisrészt.

6.1.2.2. Vektorgrafikus geometriai objektumok adatbázis adatai

A vektorgrafikus adatbázisban tehát nem a tárgyak, testek képeit, hanem a megfelelő *3D-s világtérbeli geometriai objektumok matematikai és strukturális adatait dolgozzuk fel.* Ezeknek az adatoknak a konkrét tartalma és megjelenési formája (például csak egy poliéder csúcsainak koordinátáit tároljuk, vagy emellett még a lapjainak normálvektorát is) a kiválasztott geometriai modellezési eljárástól függ, mellyel a 6.1.3. fejezetben foglalkozunk részletesebben.

6.1.2.3. A vektorgrafikus objektumok kapcsolatrendszere

A vektorgrafikában feldolgozott geometriai alakzatok csak a legritkább esetekben önállóak, ezek között általában a felhasználói szakterületnek megfelelő kapcsolatokat is ki kell építeni. A geometriai objektumok közötti strukturális kapcsolatokat a rendszerek egy részénél a felhasználó viszonylag szabadon megválaszthatja, más esetekben viszont az objektumok lehetséges kapcsolatai csak egy nagyon kötött, szigorú szabályrendszer szerint alakítható ki.

Így például:

- egy építészeti tervezőprogram esetén a tervező egy ház geometriai részegységeit (ablakok, tetőszerkezet stb.) lényegében korlátozás nélkül megválaszthatja,
- egy elektronikus áramkör tervező program esetén viszont az egyes áramköri részegységek térbeli elhelyezkedése erősen kötött.

A geometriai objektumok közötti kapcsolatok lehetnek:

- *alá–fölérendeltségi hierarchikus viszonyok, melyek jellemző változata a tartalmazás* (például: ház → tetőszerkezet → tetőablak),
- *mellérendeltségi viszonyok.*

A mellérendelt kapcsolatok közül a legfontosabbak a következők:

- a szerkezeti jellegű kapcsolatok, amikor a geometriai objektumok hierarchiában azonos felsőbb szintű objektumhoz kapcsolódnak (például egy ház egyik falán található ablakok),
- az illeszkedési jellegű kapcsolatok, amikor a geometriai objektumok valamilyen formában csatlakoznak egymáshoz (például egy kocka egy csúcsából kiinduló 3 él),
- *a megjelenítés jellegű kapcsolatok, amikor a geometriai objektumok azonos raszteres képhez, jelenethez (scene) tartoznak.*

6.1.2.4. A vektorgrafikus geometriai objektumokhoz rendelt tulajdonságok

A vektorgrafikus adatbázis geometriai objektumaihoz különböző – jellemzően megjelenítésorientált – *tulajdonságokat is hozzárendelhetünk.* Ilyenek például a rasztergrafikához hasonlóan

- *a szín,*
- *a vonalstílus,*
- *a felületi jellemzők*: textúrák, felületre vetített raszteres képek, érdesség stb.,
- *a szövegek.*

Ezek – mint a geometriai objektumokhoz rendelt paraméterek – szintén letárolásra kerülnek a vektorgrafikus adatbázisban.

Fontos kérdés, hogy a geometriai objektumokhoz rendelt tulajdonságokat, az objektumok kapcsolati rendszerében hogyan kezeljük. A vektorgrafikus rendszerek általában lehetővé teszik a hierarchikus struktúrákban az objektumokhoz rendelt tulajdonságok automatikus átöröklését (azaz, ha a „szülő"-höz rendelt tulajdonság megváltozik, akkor ezt a „gyerekek" felhasználói beavatkozás nélkül öröklik).

6.1.2.5. A modelltér geometriai objektumaihoz hozzárendelt mennyiségi és szervezési információk

A vektorgrafikus szoftverek legtöbbje azt is lehetővé teszi, hogy a modelltér geometriai objektumait különböző mennyiségi és szervezési információkkal is összekapcsoljuk. Erre egy jó példa egy vektorgrafikus térkép, melyhez hozzárendeljük az úthálózat gráfját például az egyes útszakaszokhoz tartozó sebességkorlátozással.

Összefoglalva az eddigieket, *a vektorgrafikus adatbázis a következő jellegű adatokat tartalmazza:*

- *a 3D-s világkoordináta-rendszerben meghatározott geometriai objektumok adatait,*
- *a geometriai objektumok közötti viszonyokat meghatározó strukturális kapcsolatok adatait,*
- *a geometriai objektumokhoz rendelt tulajdonságok adatait,*
- *a modelltér geometriai objektumaihoz rendelt mennyiségi és szervezési információk adatait.*

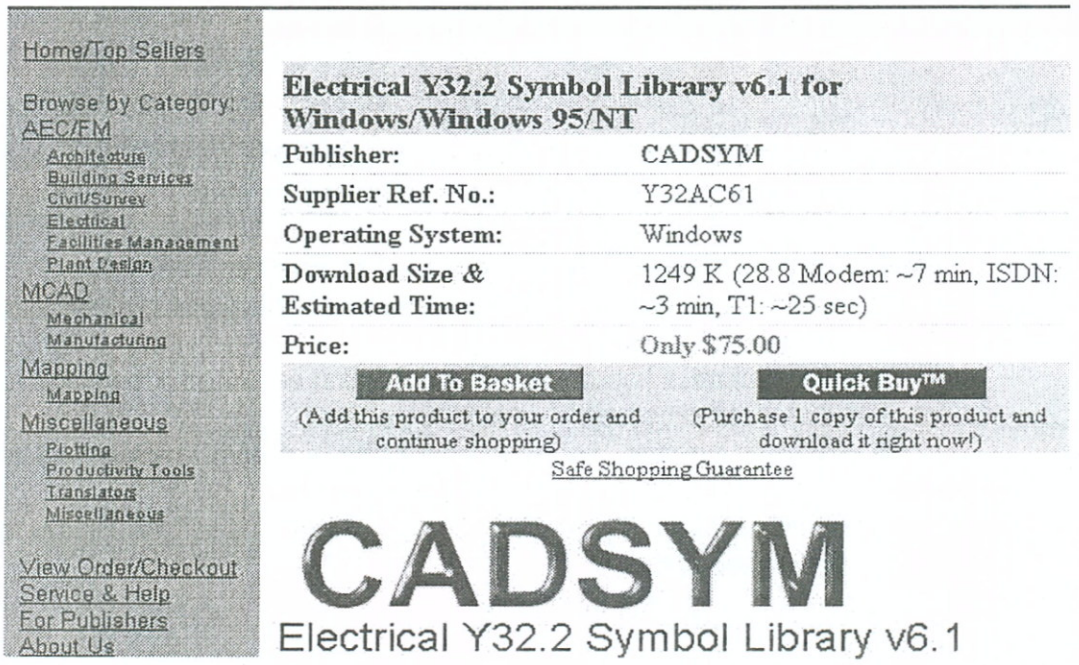
6.1.2.6. A vektorgrafikus alakzatkészlet továbbépíthetősége, a vektorgrafikus információk világméretű felhalmozódása

Mint már kifejtettük, a vektorgrafikus rendszerek lehetővé teszik a vektorgrafikus objektumok adatbázisban történő tárolását, visszakeresését és ezek között a legkülönbözőbb fajta strukturális kapcsolatok létesítését és kezelését. Mindez lehetővé teszi a vektorgrafikus objektumokban megtestesülő szellemi értékek felhalmozódását, ezek többcélú újrahasznosítását. A világméretekben végbemenő folyamat a jelenlegi komplex szoftverrendszerek kialakulásával hasonlítható össze, jellemző állomásai a következők:

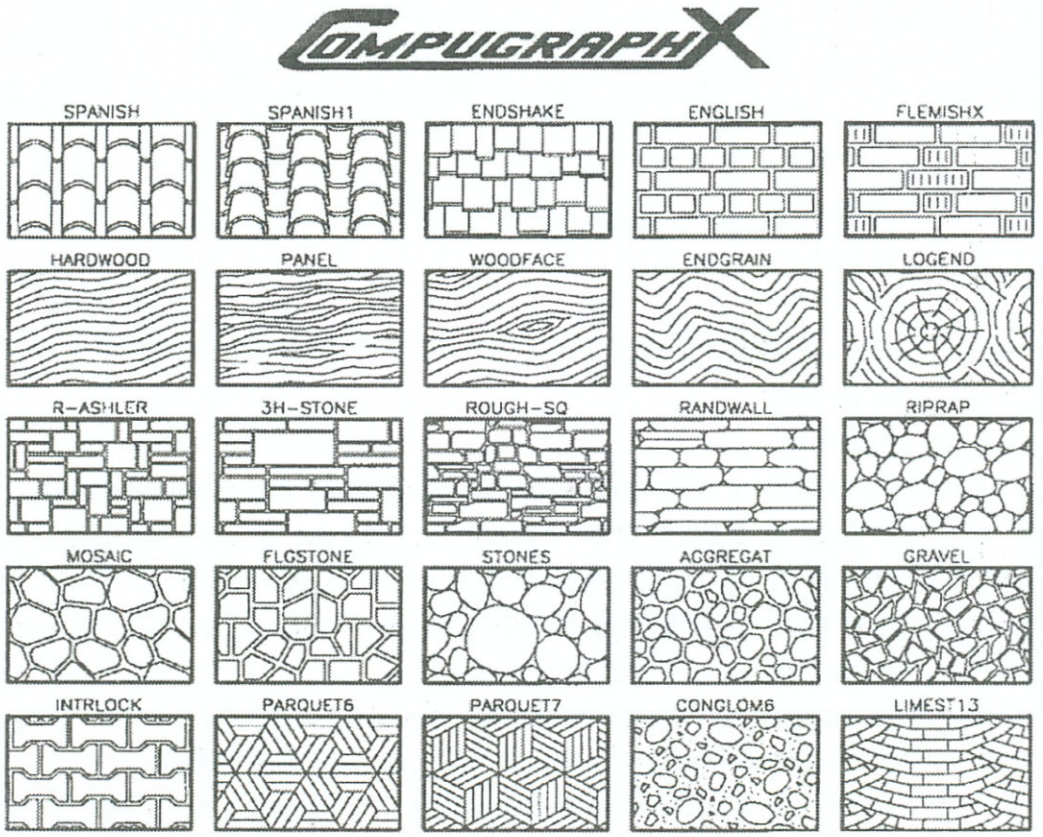
- kezdetben egy-egy reklámgrafikus vagy tervezőmérnök saját számítógépes rendszerében letárolta a számára gyakran szükséges vektorgrafikus építőelemeket (rajzelemek, szimbólumok, alkatrésztervek) és ezeket többször, ismételten felhasználta a munkája során;
- a helyi hálózatok elterjedésével a vektorgrafikus objektumkönyvtárak más szervezeti szinten (például egy építészeti tervezőcégnél a legkülönbözőbb háztervrajzokba beépíthető standard szerkezeti elemek tervei) jönnek létre és hasznosulnak;
- az Internet is elősegítette, hogy napjainkban már megkezdődött a vektorgrafikus adatbázisokban tárolt objektumkönyvtárak hálózati cseréje és hasznosítása, ezzel beindult a vektorgrafikus információk világméretű felhalmozódása.

A vizuális információk országhatárokon átlépő, eredeti szellemi tőke-felhalmozódására néhány jellemző tendencia:

- a mérnöki tervezőrendszerek már lehetővé teszik a hálózati csoportmunkát. Ez azt jelenti például, hogy különböző országokban dolgozó tervezőmérnökök együtt tudnak működni, és egy komplex tervet közösen ki tudnak dolgozni;
- a vektorgrafikus objektumkönyvtárak specializálódnak, az adatcserére szabványokat (például DXF fájlformátum) dolgoznak ki;
- az objektumkönyvtárak jelentős része a WEB-en is hozzáférhető. Erre mutatnak példát a 173. sz. és 173. sz. ábrák.



173. sz. ábra
Elektromos szimbólumok könyvtára a WEB-en



174. sz. ábra
Építészeti tervezésben hasznosítható objektumkönyvtár a WEB-en

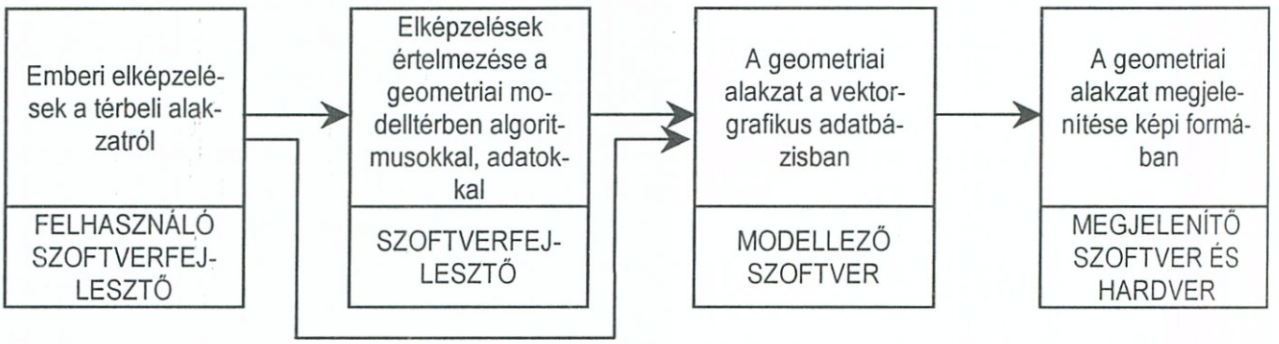
6.1.3. A geometriai modellezés alapjai

A vektorgrafikában a geometriai modellezéssel a valós vagy elképzelt (virtuális) világok térbeli objektumai geometriai jellemzőinek számítógépes kezelését, valamint a jelenetek képi megjelenítését kell megoldani.

A geometriai modellezés ennek megfelelően több lépésben történik:

- az ember megfogalmazza a térbeli alakzattal kapcsolatos elképzeléseit (forma, nagyság stb.),
- a térbeli alakzattal kapcsolatos elképzeléseket értelmezzük a modelltérben matematikai összefüggésekkel és algoritmusokkal,
- a modelltér eljárásai, adatai a számítógép számára értelmezhető formában (utasításkódok, adatok) kerülnek rögzítésre,
- a számítógépen tárolt programok, adatok alapján a térbeli alakzat képi formában megjelenítésre kerül.

A lépéseket a 175. sz. ábra szemlélteti:



175. sz. ábra
A geometriai modellezés lépései a vektorgrafikában

A geometriai modellezés lehet:

- *adatorientált és*
- *eljárásorientált.*

Adatorientált esetben a térbeli alakzat jellemző adatait tároljuk a számítógépes rendszerben (például háromszög esetében a csúcspontokhoz vezető vektorok koordinátáit) *míg eljárásorientált esetben a térbeli alakzat generáló programját* (például körgeneráló rutin a kör egyenlete és a középpontjának és sugarának paraméterei alapján).

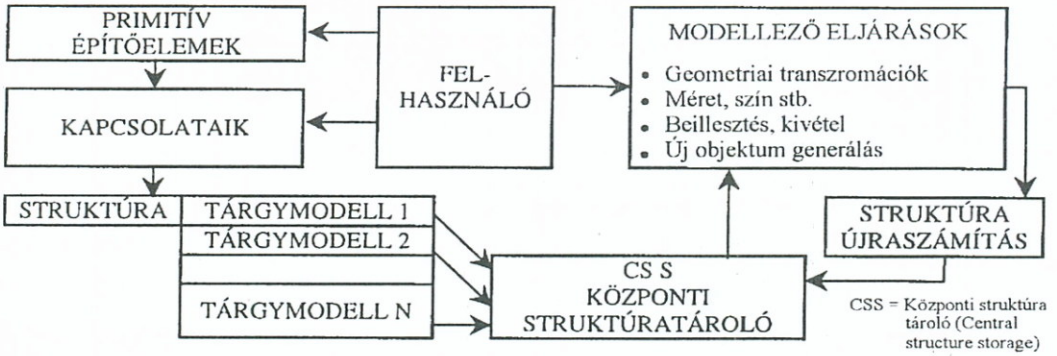
Az előzőeknek megfelelően a vektorgrafikában alkalmazott *geometriai modellek a következő adatokkal jellemezhetők az adatbázisban:*

- *a modell neve, azonosítói, a geometriai alakzatot felépítő geometriai építőelemek fajtái,*
- *az építőelemek kapcsolódásaira vonatkozó adatok,*
- *a geometriai alakzatra vonatkozó méret-, nagyságadatok,*

- *a geometriai alakzatra vonatkozó hely- és helyzetadatok* a modellezési világkoordináta-rendszerben,
- *a geometriai alakzat tulajdonság adatai,*
- *a geometriai alakzat megjelenítésének adatai.*

Azokat az építőelemeket, melyek tovább már nem bonthatók fel geometriai primitíveknek nevezzük. Legtöbb vektorgrafikus rendszerben a geometriai alakzatokat a primitíveknek a modelltérben való „összeépítésével" hozzuk létre.

A modelltér vektorgrafikus objektumai a geometriai primitívekből általában hierarchikusan épülnek fel. A primitívek a vektorgrafikus rendszer objektumkönyvtárából a programcsomagok felhasználói felületein kiválaszthatók, és ezzel a primitívekből való összeépítést a felhasználó határozza meg (lásd a 176. sz. ábra).



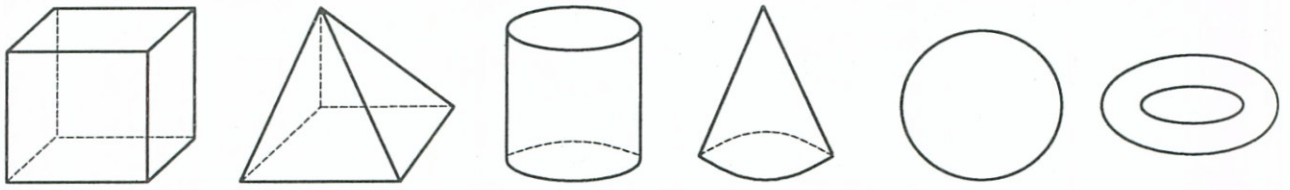
176. sz. ábra
A vektorgrafikus objektumok felépítése primitívekből

A 2D-s rajzolóprogramok az alakzatokat általában
- a vonal (szakasz),
- a téglalap (négyzet),
- a ellipszis (kör),
- a sokszög

primitívek véges számú kombinációjával állítják elő.

A *3D-s vektorgrafikus rendszerekben* ezek általában még a következő térbeli primitívekkel egészülnek ki:
- *hasáb (beleértve a téglatestet és a kockát is),*
- *gúla (csonka gúla),*
- *henger,*
- *kúp (csonkakúp),*
- *gömb,*
- *torusz.*

Ezeket a 3D-s primitíveket mutatja be a 177. sz. ábra.



a) hasáb b) gúla c) henger d) kúp e) gömb f) torusz

177. sz. ábra
Térbeli primitívek

Egyszerűbb modellezőrendszerek a térbeli íveket és felületeket szakasz, illetve háromszögközelítéssel modellezik. Ennél hatékonyabb eszközt használnak a korszerű vektorgrafikus rendszerek, melyek a III. fejezetben megismert eljárásokkal, azaz *paraméteres köbös ívdarab és kétparaméteres köbös felületfolt primitívekkel állítják elő a térbeli íveket és felületeket.*

A vektorgrafikával foglalkozó kutatók már jó pár éve meghatározták azt a követelményrendszert, melyet egy „minden igényt kielégítő" geometriai modellezésnek teljesítenie kell. A teljesség igénye nélkül ezek közül néhány fontosabb:

- a modellező rendszerben biztosított algoritmusoknak megfelelő pontossággal kell megközelíteniük a valódi térbeli alakzatokat (Természetesen a pontosság alkalmazásfüggő. Ugyanakkor vannak olyan modellező eljárások is, melyek teljesen egzakt matematikai objektumokkal dolgoznak),
- a modellező rendszerrel elő kell tudni állítani az „összes" felmerülő térbeli alakzatot,
- a vektorgrafikus objektumokkal engedélyezett műveletekkel (lásd 6.1.4. fejezet) előállított testek csak olyanok lehetnek, melyet a modellező rendszerben is elő lehet állítani.

A fenti kritériumoknak megfelelő, „ideális" geometriai modellezőmódszer kidolgozására irányuló intenzív kutatómunka ellenére sem sikerült olyan univerzális modellezőrendszert találni, mely az elmélet és a gyakorlat által támasztott összes igényeknek teljeskörűen megfelelne.

A vektorgrafikában napjainkban többfajta geometriai modellezőmódszert használunk, melyet a vektorgrafika felhasználási területe és az alkalmazott számítógépkonfiguráció adta lehetőségek alapján választunk ki. A legelterjedtebben használatos geometriai modellező módszerek a következők:

- huzalvázas (drótváz) modellezés,
- palástfelületekkel való modellezés (b-rep),
- tömör testmodellezés (CSG).

Ezek mellett igen gyorsan fejlődik – főként a hardver teljesítmények exponenciális növekedése miatt – a térfelosztással való modellezés, amikor a testet több kisebb tömör tárgyra való felbontással közelítjük. Mindezeket a módszereket a 6.2. fejezetben tárgyaljuk részletesen.

Fontos megemlíteni, hogy a különböző geometriai modellezési módszerek kombinált alkalmazását a konkrét vektorgrafikus rendszerek általában megengedik, és ezeket az adatbáziskezelés szintjén összekapcsolják. Így például, lehetőségünk van egy palástfelületekkel modellezett test huzalvázas megjelenítésére a képernyőn. Ugyanakkor sok esetben a különböző modellezési eljárásokkal előállított objektumok adatait nem lehet egyértelműen konvertálni.

Az „ideális" geometriai modellezési módszer kifejlesztésének nehézségei közül az algoritmusok műveletigényével és a matematikai problémákkal kapcsolatosakat szeretnénk érzékeltetni.

Viszonylag egyszerű testmodellezést valósíthatnánk meg, ha a 3D-s testeket azonos elemi testekkel, például kockákkal közelítenénk (a 2D-s raszteres képek pixeljeihez hasonlóan).

Látható viszont, hogy míg egy n sorból és n oszlopból álló pixelből felépülő raszteres kép előállításának műveletigénye n^2-el arányos (jobb felbontásoknál ez milliós nagyságrend), addig az n sorból és n oszlopból álló n magasságú kocka modellezésének műveletigénye n^3-el arányos (ez viszont már 1000 MIPS nagyságrendű műveletet is jelenthet).

A tetszőleges térbeli alakzatok geometriai modellezése például olyan kérdéseket vet fel (gondoljunk a képernyőn történő megjelenítésre), hogy a térbeli geometriai objektumok mikor képezhetők le kölcsönösen egyértelmű módon 2D-s pontsokaságokra. (Az ilyen tulajdonságú térbeli alakzatokat a szakirodalom manifoldnak nevezi.) Matematikailag könnyen belátható, hogy vannak olyan térbeli alakzatok is, melyeknek az előbbieknek megfelelő leképezése nem lehetséges.

Egy másik problémára mutat rá a b-rep modellezés adatstruktúrája (lásd részletesen 6.2.2. fejezet), melyben a testet határoló felületfoltok és ezek élekben és csomópontokban való csatlakozásának adatait kezeljük. Mikor állítható vissza például ebből az adatstruktúrából egyértelműen az eredeti test?

Mindezek az úgynevezett kombinatorikus topológia alkalmazását követelik meg a vektorgrafika geometriai modelljeiben.

Topológiának a matematikának azt a területét nevezzük, mely a folytonos transzformációkkal szembeni invariáns tulajdonságokkal foglalkozik. Így például a gömb és a kocka topológiailag ekvivalens alakzatok, a gömb és a torusz pedig nem.

6.1.4. Műveletek a modelltér vektorgrafikus objektumaival

A vektorgrafikus szoftverek a *modelltér objektumaival* a következő típusú *műveleteket* teszik lehetővé:

- *új objektum létrehozása,*
- *egy létező objektum transzformálása, másolása, törlése,*
- *meglévő objektumokkal végzett halmazalgebrai műveletek,*
- *struktúra képzés,*
- *jelenetek* (scene) *megjelenítése.*

6.1.4.1. Új objektumok létrehozása

A modelltérben egy új objektumot általában

- primitívpéldányokra való hivatkozással, vagy
- szerkesztéssel, vagy
- pásztázással

definiálhat a felhasználó.

Primitívpéldányokra való hivatkozás

Új objektumot primitívpéldányokra való hivatkozással legtöbb rendszerben úgy hozhatunk létre, hogy az objektumnak megfelelő *elemi test angol nevére hivatkozunk.* Így például:

- *BLOCK* → hasáb definiálása
- *CYLINDER* → henger definiálása
- *CONE* → kúp definiálása
- *TORUS* → torusz definiálása

A primitívpéldányt generáló parancsban a primitív típusa mellett meg kell adni a típusnak megfelelő jellemző geometriai paramétereket (például henger esetén az alapkör sugarát és a henger magasságát). Ha ezt a felhasználó nem közli, akkor az adott objektumot a rendszer alapértelmezés szerinti méretekben generálja le. (Például henger esetében ez általában egységnyi sugarú és magasságú testet jelent.)

Objektum létrehozása szerkesztéssel

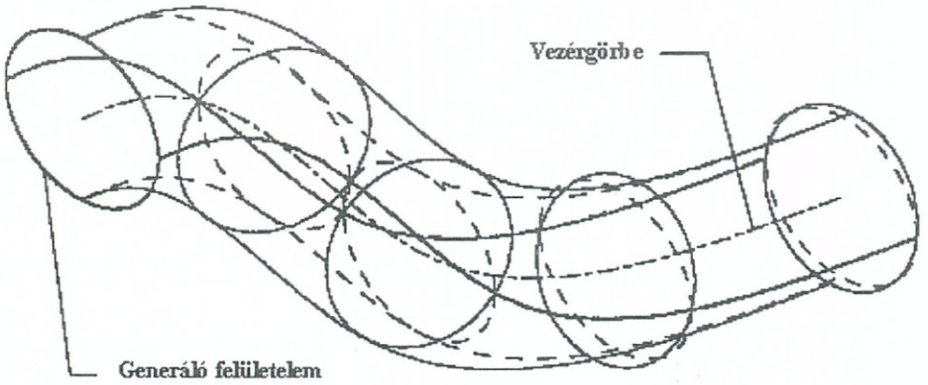
Ebben az esetben a felhasználó adja meg a vektorgrafikus rendszer számára az összes információt, mely alapján a térbeli test összeállítható. Ez a módszer tipikusan jellemző a testek palástfelületekkel való modellezésére (b-rep = részletesen lásd 6.2. fejezetben), amikor a felhasználó egyenként meghatározza az egyes fedőlapok, felületek jellemzőit és ezek csatlakoztatására vonatkozó adatokat.

Objektum definiálása pásztázással

A 3D-s vektorgrafikus objektumok generálásának fontos módszere a pásztázás (ezt gyakran söprésnek is nevezik, angolul SWEEP).

Pásztázásnál egy 2D-s felületelemet mozgatunk egy vezérgörbe mentén, és ennek során a felületelem által „súrolt" térbeli pontok egy testet határoznak meg.

Erre egy példát a 178. sz. ábra mutat, melynél egy körlapnak egy térbeli görbe mentén való mozgatásával, egy 3D-s görbe „csövet" állítunk elő.



178. sz. ábra
Térbeli test generálása pásztázással

Matematikailag a pásztázással generált *T* test pontjai a generáló *F* felületelem és a *V* vezérgörbe ponthalmazainak *F* x *V* Descartes szorzatán értelmezett *f* leképezéssel állítható elő:

$$T = f(F \times V)$$

Legáltalánosabb formájában a pásztázás matematikai kezelése komoly topológiai és geometriai problémákra vezet. (Például a generáló felületelemre fontos kikötéseket kell tennünk: nem lehetnek benne például egy dimenziós „kivágások". Nehéz elméleti vizsgálatokra vezet az a kérdés, hogy milyen tulajdonságú az általános pásztázással előállítható összes test halmaza.) Ugyanakkor viszont a pásztázás egy nagyon szemléletes testgenerálási módszert biztosít a felhasználó számára, és a pásztázással létrehozott testek számítástechnikai kezelése relatíve nem okoz nehézséget (például egy a pásztázást végrehajtó program írása nem túl bonyolult).

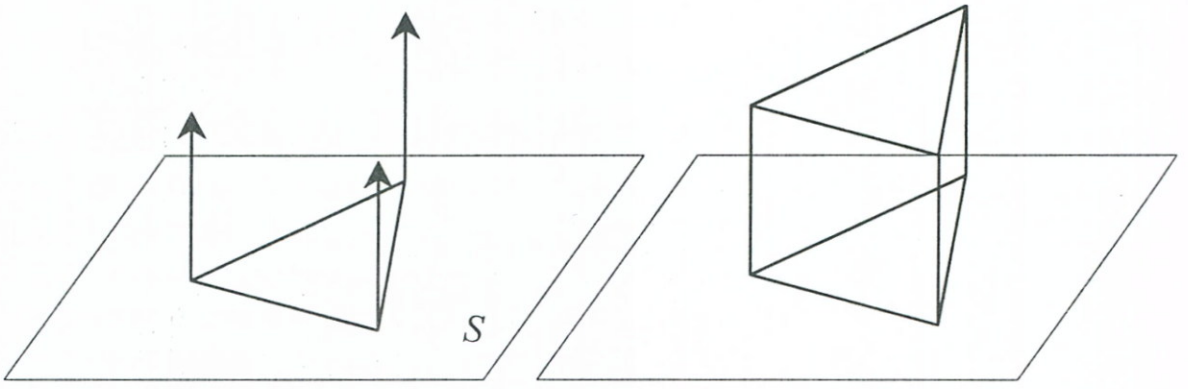
A pásztázás legáltalánosabb formájával létrehozott testek beillesztése a geometriai modellezőeljárások adatstruktúrájába viszont egyszerűen nem algoritmizálható.

A legtöbb vektorgrafikus rendszer kezeli a pásztázás két speciális esetét, ez

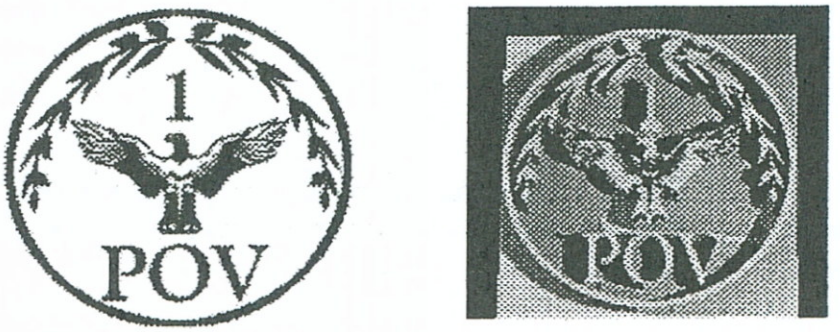
- a kihúzás (EXTRUDE) és
- a forgatás (ROTATE).

A kihúzásnál és a forgatásnál egyaránt síkbeli 2D-s alakzatokból indulunk ki és a vezérgörbe merőleges a generáló alakzatra.

A kihúzásnál a síkra merőlegesen, egy egyenes mentén „kihúzzuk" a 2D-s alakzatokat (lásd a 181., 180. sz. ábrák).



179. sz. ábra
Hasáb generálása 2D-s síkbeli háromszög kihúzásával

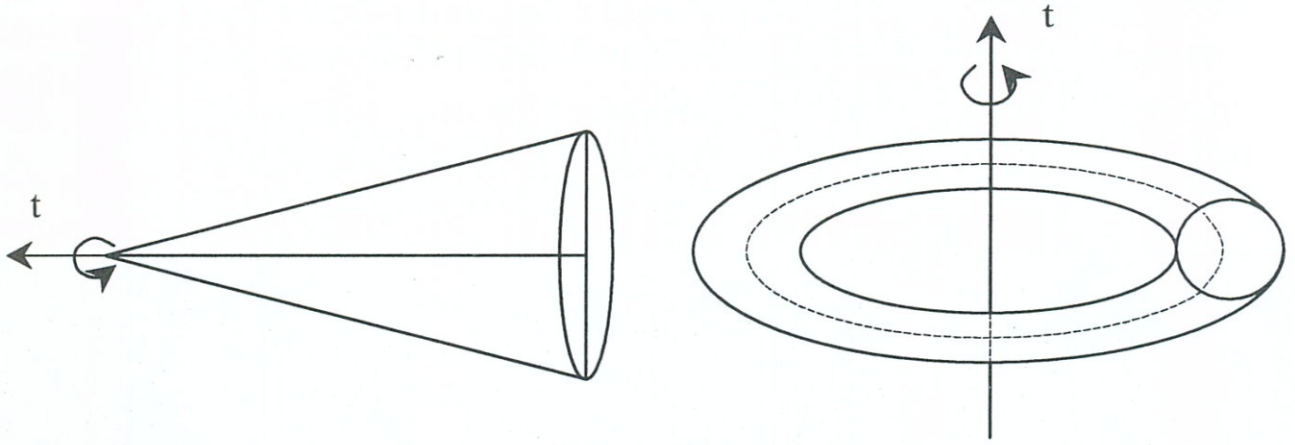


a) b)

180. sz. ábra
Kihúzás a POVRAY programmal
a) POV clipart b) generált 3D-s kihúzott, árnyékolt kép

Meg kell jegyezni, hogy a kihúzás a 2 ½ D vektorgrafikus rendszerek jellemző „test" generálási művelete. Ekkor a létrejövő „test" magassága (azaz a kihúzás hossza) a 2D-s vektorgrafikus alakzathoz tulajdonságként kerül hozzárendelésre.

Forgatásnál a vezérgörbe kör, a generáló alakzat síkja a forgatás bármely pillanatában merőleges a vezérgörbére, és a generáló alakzat pontjai is körön mozognak. Példákat a 181. sz. ábra mutat.



181. sz. ábra
Kúp generálása háromszöglapból (a) és
torusz generálása körlapból (b) forgatással

6.1.4.2. Objektumok másolása, törlése, transzformálása

A másolás a kijelölt vektorgrafikus objektum egy eltérő nevű (azonosítójú) példányát hozza létre a világkoordináta-rendszer egy másik helyén.

A törlés egy korábban definiált struktúrába nem beépített objektumot szüntet meg, ami azonosítójának (nevének) törlését és a megfelelő adatbázisrész felszabadítását jelenti.

Fontos hangsúlyozni, hogy egyszerű törléssel csak struktúrába nem szervezett, illetve Boole-algebrai kompozíciós láncba be nem kapcsolt objektumokat szüntethetünk meg. Ellenkező esetben a struktúrából „ki kell vennünk" az objektumot (lásd 6.1.4.3. és 6.1.4.4. pont), és ehhez általában még további eljárásokra is szükség van.

Objektumok transzformálásához a 2. fejezetben megismert pont-transzformációk közül választhatunk. Általában minden vektorgrafikus rendszer lehetővé teszi az objektumoknak

- *az eltolását*,
- *az elforgatását* (ide értve a tükrözést is) és
- *a léptékváltást* (nagyítás, kicsinyítés, összenyomás, széthúzás).

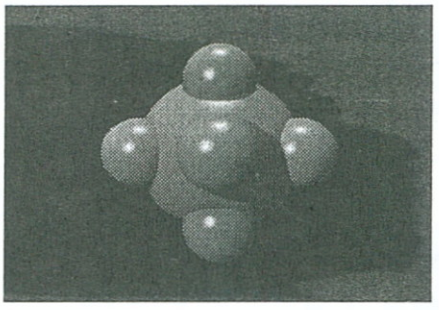
6.1.4.3. Objektumokkal végzett halmazalgebrai műveletek

Az objektumokkal végzett halmazalgebrai műveletekhez a felhasználó a művelet kiinduló testeinek azonosítóit és a művelet eredményeképpen létrejövő test azonosítóit adja meg.

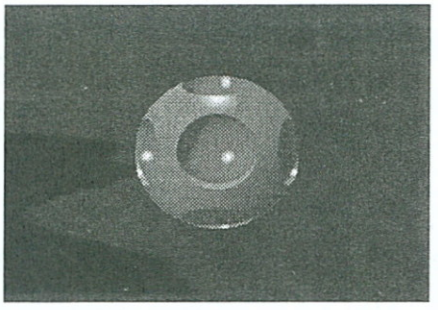
Az általánosan használt és legtöbb vektorgrafikus rendszerben értelmezett Boole-algebrai műveletek a következők:

- egyesítés (UNION),
- kivonás (DIFFERENCE vagy SUBTRACT),
- metszet (INTERSECTION).

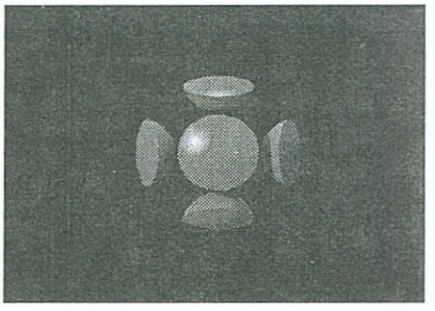
Erre mutatnak példát a 184., 185., 184. sz. ábrák, melyek egy nagyobb és 6 kisebb gömbbel való műveleteket szemléltetik.



182. sz. ábra
Testek egyesítése

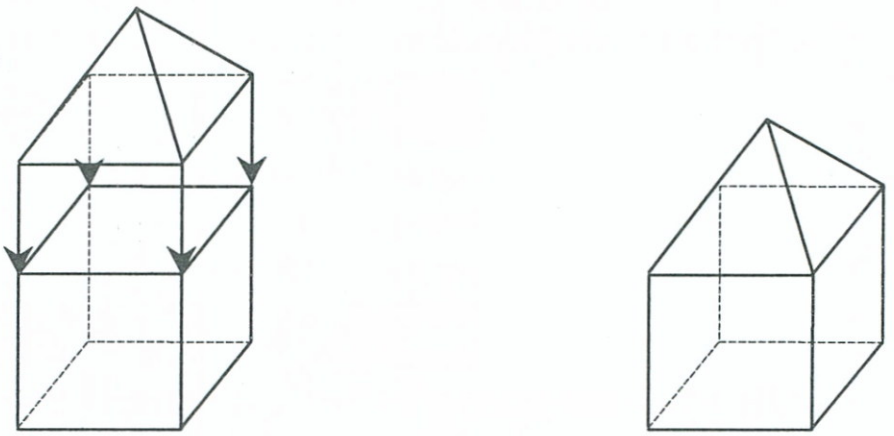


183. sz. ábra
Testek kivonása



184. sz. ábra
Testek metszete

! A napi gyakorlati munkavégzés komfortjához ezek mellett még szükség van az *összeragasztás (GLUE) műveletére is, melynek során a műveletben résztvevő testeket egy közös felületen csatlakoztatjuk* (lásd 185. sz. ábra).



185. sz. ábra
Gúla és kocka testek összeragasztása

6.1.4.4. Az objektumok között strukturális kapcsolatok létesítése és megszüntetése

Mint azt már röviden kifejtettük, egy vektorgrafikus rendszer objektumai között különböző kapcsolatok is felépülnek, illetve felépíthetők. Ezek egy részét a rendszer automatikusan (a felhasználói beavatkozás nélkül) felépíti, jelentős részük kialakítását viszont a felhasználó vezérelheti. Ezek a

lehetőségek és a konkrét parancsok a konkrét programcsomagok függvényében változhatnak, ezért az elkövetkezendőkben csak két példát ismertetünk.

A csoportképzéssel (GROUP) objektumokat összefoghatunk egy egységbe, illetve ezt meg is szüntethetjük (UNGROUP). A csoportok általában több szinten hierarchiában is felépíthetők és a rendszerparancsoknak egy része a csoportokra is értelmezett.

Ezek közül elsőként a tervezőrendszerekben alkalmazták *a rétegeket vagy fóliákat* (LAYER), mely módszert azóta már más szakterületeken használatos grafikus szoftverek is átvettek.

Ennek lényege, hogy *az objektumok a felhasználó által megadott csoportosításban külön fóliákra kerülhetnek*, melyek a rendszerben külön kezelhetők. Ezt egy 2D-s rajzelemek esetében úgy képzelhetjük el, hogy a rajz egyes részeit különálló, teljesen átlátszó pauszra készítjük el. Ezeket különkülön is megtekinthetjük. *Egymásra rakva az átlátszó fóliákat, láthatjuk a teljes rajzot is.*

Az egyes fóliák „lefagyaszthatók", ekkor tartalmuk például a képernyőn nem jelenik meg. Amennyiben viszont később a lefagyasztott fóliára ismét szükség lenne, ezeket „elővehetjük" és így többcélúan is felhasználhatjuk. A fóliák segítségével például elérhetjük, hogy egy ház építészeti terveit épületgépészeti vagy belsőépítészeti terv céljaira is felhasználjuk.

6.1.4.5. Objektumok megjelenítése

A vektorgrafikus objektumok kapcsolataikkal együtt a modelltérben, definiálásukat követően „objektíve" léteznek. Ezeknek az objektumoknak 2D-s raszteres képen való megjelenítéséhez elsőként azt kell eldönteni, hogy a világtérben létező objektumok közül melyeket szeretnénk szerepeltetni a képen. Már említettük, de a fontossága miatt megismételjük:

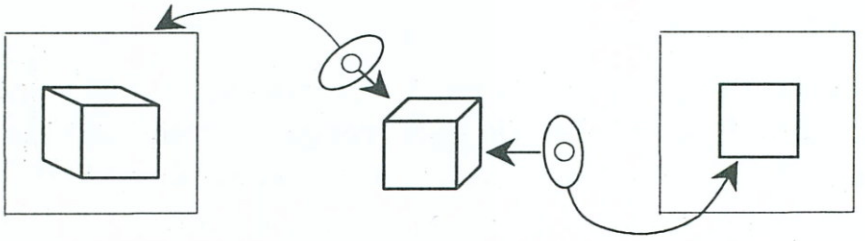
A vektorgrafikus objektumoknak azt a csoportját, melyet egy képen szerepeltetünk, jelenetnek (scene) nevezzük.

A modelltérben definiált összes objektumnak nem szükségképpen kell szerepelnie minden jelenetben, ehhez kiválaszthatjuk az objektumok egy részét, de akár egy objektumból is képezhetünk jelenetet. Ha már kiválasztottunk egy jelenetet, akkor a raszteres kép meghatározásához rögzítenünk kell, hogy ezt a jelenetet a világtér melyik pontjából látjuk. A vektorgrafika nyilvánvalóan lehetővé teszi, hogy a jelenethez tartozó 3D-s alakzatcsoportot a tér bármelyik pontjából szemügyre vegyük és a nézőpont rögzítése erősen befolyásolja a képen látottakat.

! *A vektorgrafikus rendszerekben egy jelenet képének meghatározásához szükséges nézőpontot és irányt a világkoordináta-rendszerben általában a KAMERA paranccsal definiáljuk.*

Szemléletesen ezt úgy képzelhetjük el, hogy a raszteres képen azt fogjuk látni, amit egy fényképezőgéppel tudnánk felvenni, ha a fényképezőgépet a KAMERA parancsban meghatározott helyen működtetnénk a modelltérben.

A kameraállásoktól függő raszteres képekre mutat példát a 186. sz. ábra.



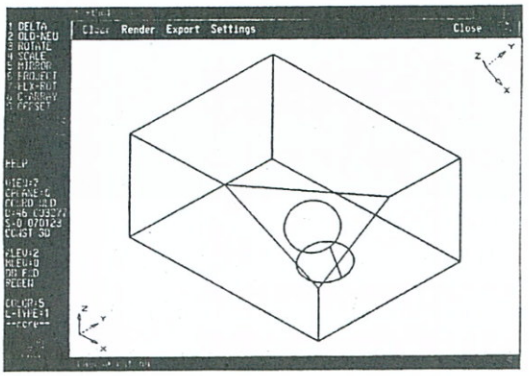
186. sz. ábra
Kocka raszteres képe különböző kameraállásokból

! Különösen a fotorealisztikus ábrázolásnál fontos, hogy *a modelltérben lévő objektumok megvilágítását* meghatározzuk. Ezt megtehetjük például azzal, hogy *a világkoordináta-rendszerben meghatározott helyű, erősségű és színű fényforrásokat definiálunk.*

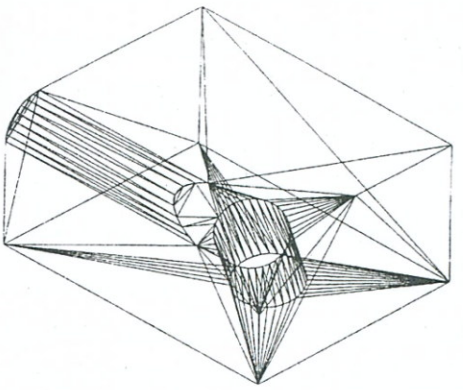
A modelltérben definiált vektorgrafikus objektumok jeleneteinek megfelelő képek előállításához azt is rögzíteni kell, hogy a képen az objektumok szemléltetése milyen formában történjen. A vektorgrafikus szoftverek erre vonatkozó parancsai általában a következők:

! - *WIREFRAME = huzalvázas megjelenítés*
- *NET = poliéder közelítésű vonalhálós megjelenítés*
- *HIDDEN = takartvonalas palástmodell*
- *SHADE = felületárnyalt testmodell*

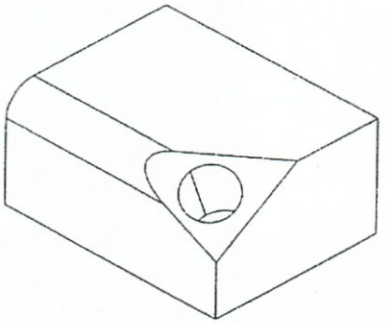
Erre mutatnak példát a következő ábrák:



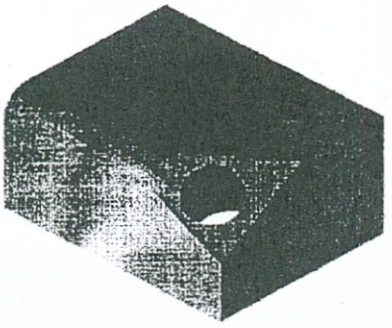
187. sz. ábra
Huzalvázas megjelenítés



188. sz. ábra
Poliéder (háromszög) közelítésű vonalhálós megjelenítés



189. sz. ábra
Takartvonalas palástmodell



190. sz. ábra
Felületárnyalt testmodell

Az ábrákon látható, hogy huzalvázas megjelenítés esetén a tárgyakat úgy rajzoljuk ki a képernyőn, mintha drótból lennének. Ekkor az összes tárgy minden éle látható.

A takartvonalas megjelenítés esetében viszont a rendszer értékeli, hogy az adott kameraállásból egy él látható vagy sem (azaz a testnek a kamera felé eső részén helyezkedik-e el az él, vagy sem, illetve az élt más testek takarják vagy sem).

Árnyékolt megjelenítés esetén a testek lapjai kitöltött területként (színnel vagy felületi mintázattal azaz textúrákkal) kerülnek megjelenítésre. Ekkor például az „egymásfölé" kerülő lapok közül a kamerához közelebbi takarja a távolabbit és az objektumoknak a kamera állásból rejtett részei nem jelennek meg a képernyőn.

Az árnyékolt megjelenítés esetén a vektorgrafikus rendszerek legtöbbször lehetővé teszik a megvilágításfüggő és a nem megvilágításfüggő kirajzolás megkülönböztetését. Ha megvilágításfüggő a megjelenítés, akkor a vektorgrafikus objektumok között megkülönböztetünk megvilágított és világító tárgyakat. Világító tárgyak pixeljeinek színét a képen a fényforrás színe és erőssége határozza meg. Megvilágított objektum esetén a tárgy alapszíne a fényforrás távolsága és a megvilágító fény beesési szöge szerint megváltoztatásra kerül például a Lambert-féle megvilágítási törvények szerint.

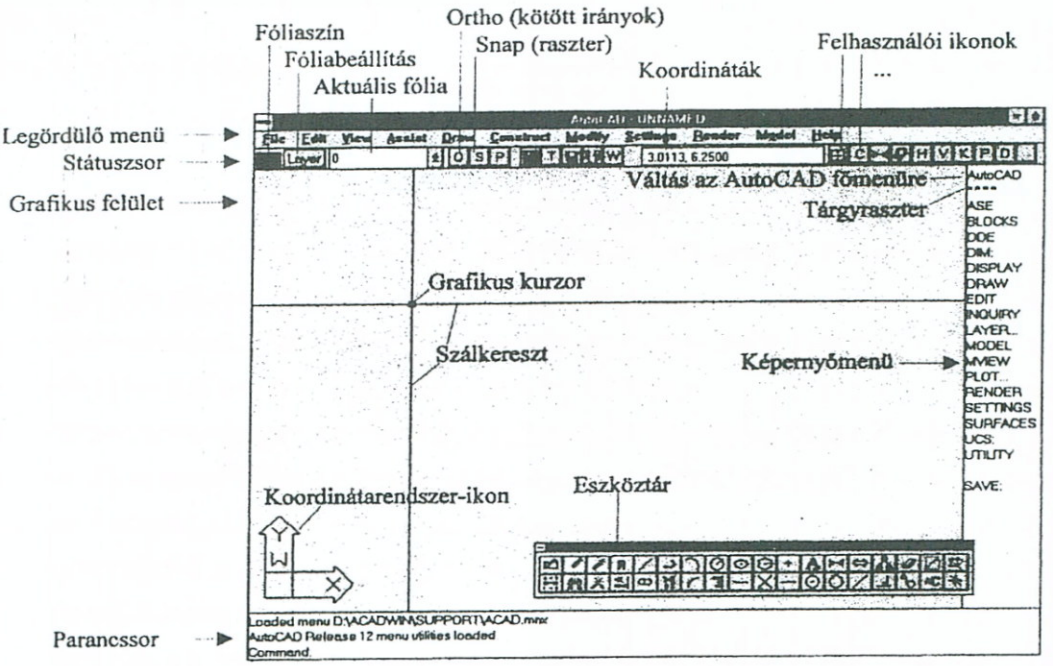
A különböző tárgyak láthatósága és takarása, illetve a fényforrásoktól függő szín és megvilágítási viszonyok meghatározása viszonylag összetett algoritmusokat igényel. A vektorgrafikus jelenetek megjelenítéséhez szükséges legfontosabb láthatósági és megvilágítási fogalmakat és eljárásokat részletesen a 6.3., 6.4., 6.5. fejezetekben fogjuk tárgyalni.

Fontos felhívni a figyelmet arra, hogy az előbbiekben bemutatott objektummegjelenítési formákat nem szabad azonosítani a 6.2. fejezetben részletesen tárgyalt geometriai modellezési módszerekkel (huzalváz, b-rep stb.). Ennek az az oka, hogy a rendszerek többsége képes geometriai modellezési eljárástól függetlenül is a különböző típusú megjelenítési formák (huzalvázas, poliéderközelítéses stb.) képernyőn való előállítására. Így

például, ha palást felület modellezést alkalmazunk, akkor is célszerű egy tervezési munkafolyamat során a részeredményeket huzalvázas formában megjeleníteni, mivel az ehhez szükséges konverzió könnyen biztosítható és a huzalvázas megjelenítés jóval gyorsabb és kevésbé terheli a számítógépet.

6.1.5. Felhasználói kommunikáció a vektorgrafikus szoftverekkel

A vektorgrafikus szoftvereknek a felhasználói felülete általában a jól ismert és általánosan használt WINDOWS-os ablakozó felület. Így megtalálhatjuk a szokásos legördülő és repülőmenüket, az ikonrendszert, az eszköztárat stb. (lásd 191. sz. ábra).



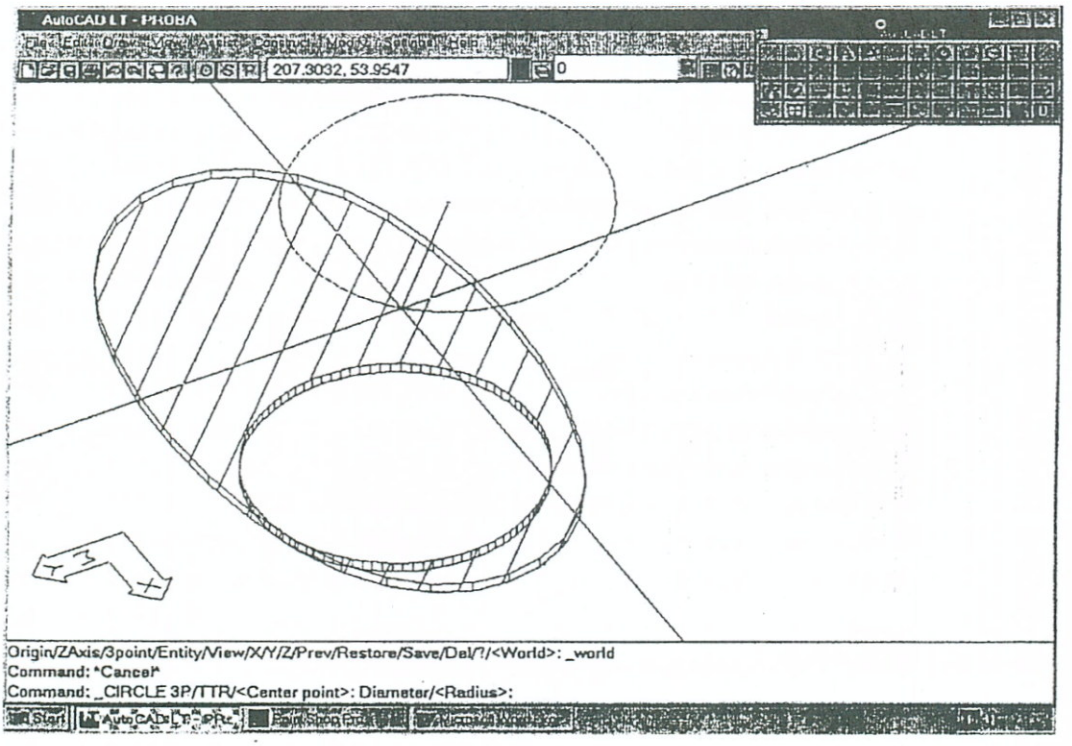
191. sz. ábra
Az AUTOCAD tervezőrendszer jellemző képernyőfelépítése

Az elkövetkezendőkben a vektorgrafikus rendszer és a felhasználó közötti kommunikációnak csak egy, bár nagyon fontos részterületével, a modelltérbeli 3D-s helyzet- és koordinátainformációk cseréjével foglalkozunk.

Az alapkérdés az, hogy hogyan lehet a 3D-s absztrakt világkoordináta-rendszerben lévő modelltér objektumaira vonatkozó információkat nyerni és közölni, mikor ehhez csak egy raszteres felület áll a felhasználó rendelkezésére.

Általában minden vektorgrafikus szoftver lehetővé teszi a modelltérre vonatkozó koordináta-rendszer és koordinátaértékek megjelenítését a képernyőn.

A koordináta-rendszer tengelyeire vonatkozó információt a vektorgrafikus rendszerek például ikon formájában is kirajzolják a képernyőre. Ez lehet az alapértelmezett koordináta-rendszer (lásd 191. sz. ábra) vagy ha a felhasználó saját maga definiált egy felhasználói koordináta-rendszert (UCS = User Coordinate System), akkor ennek az ikonja látható a képernyőn. A felhasználói koordináta-rendszer kijelzésére mutat példát a 192. sz. ábra.



192. sz. ábra
Felhasználói koordináta-rendszer tengelyeit szemléltető ikon
és a koordinátatengelyek kijelzése a képernyőn UCS esetében

Természetesen a harmadik koordinátatengely egy kétdimenziós képen nem jeleníthető meg, ezért az ikont a koordináta-rendszer sodrására vonatkozó konvenciók szerint kell értelmezni.

Ha az aktuálisan érvényes koordináta-rendszer rögzített, akkor minden, a vektorgrafikus rendszerrel közölt, illetve a rendszer által kijelzett koordinátainformáció erre a világkoordináta-rendszerre vonatkozik.

Érdemes ismételten kihangsúlyozni, hogy például a képernyőn megjelenő kurzorkoordináták nem a raszteres képernyő koordináta-rendszerében, hanem a 3D-s lebegőpontos világkoordináta-rendszerben értelmezendők.

! Ha *a felhasználó az alakzatok helyzetére vonatkozó információkat akar közölni a vektorgrafikus rendszerrel, akkor ezt a következő formákban teheti meg:*

- *a konkrét koordinátaértékek begépelésével billentyűzetről,*
- *a grafikus kurzor helyzetének megfelelő koordinátaértékekkel, például az egér vagy más relatív koordinátás eszköz használatával,*
- *abszolútkoordinátás eszköz például digitalizáló tábla (tablet) használatával,*
- *a modelltérben már definiált helyzetű objektumokra, vagy a képernyőn látható, a világkoordináta-rendszerben értelmezett segédeszközökre (vonalzók, segédvonalháló = GRID) való hivatkozással.*

A koordinátaértékek billentyűzetről való bevitele nyilvánvalóan a legpontosabb, bár esetenként munkaigényesebb módszer, mivel ez esetben a pontos koordinátaértékének megadását csak a lebegőpontos számábrázolás lehetőségei korlátozzák.

A grafikus kurzor helyzetével való helymeghatározás nagyon felhasználóbarát megoldás, de több problémát is felvet:

- mivel a képernyőkoordináta-rendszer 2D-s, ezzel a módszerrel csak az x és y koordinátákra vonatkozó értékeket adhatunk meg, nincs lehetőség a z koordináta bevitelére,
- a vektorgrafikus rendszerek világkoordinátákban és nem képpontokban „gondolkodnak", ezért a kurzor helyzetét és elmozdítását a világkoordináta-rendszerben kell értelmezni.

A második probléma kezelésére a vektorgrafikus rendszerek a felhasználó által paraméterezhető megoldást alkalmaznak. Így beállítható, hogy a raszteres képernyőn való kurzorelmozdítást milyen világkoordináta-rendszerbeli értéknek kell megfeleltetni.

A vektorgrafikus rendszerrel való helyzetinformációk közlésének speciális, de talán legfontosabb esete a képernyőn megjelenített modelltérbeli *objektumok egérrel történő kijelölése.*

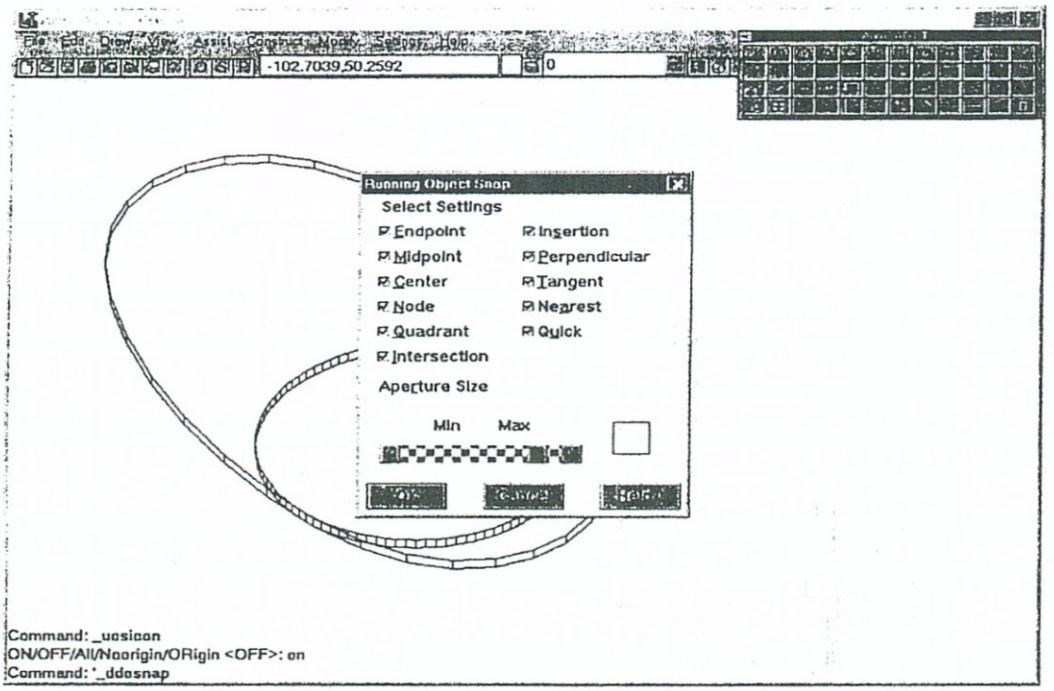
! *Ehhez a vektorgrafikus szoftverek legtöbbje azt a megoldást alkalmazza, hogy az egérrel a képernyőn rá kell mutatni a kijelölendő objektum képének határoló görbéjére, vagy zárt alakzatok belsejére.* A grafikus kurzor raszteres képernyőkoordinátáit ekkor először át kell konvertálni a modelltérbe és meg kell állapítani, hogy a konvertált koordinátaérték melyik grafikus objektumhoz tartozik.

Ennek során akkor léphet fel probléma, ha a grafikus kurzor által meghatározott koordinátaértékhez „közel" több vektorgrafikus objektum „határoló vonala" is megtalálható.

Sikertelen kijelölés esetén a felhasználó két megoldás közül választhat:

- nagyítja a képernyőn kijelölni kívánt vektorgrafikus objektum képét, vagy
- csökkenti a grafikus kurzor helyzetének világkoordinátákban történő értelmezését meghatározó felismerési környezet (Aperture Size) méretét.

Az utóbbira mutat példát a 193. sz. ábra.



193. sz. ábra
A felismerési környezet méretének (Aperture Size) beállítása az objektumok kijelöléséhez

? ELLENŐRZŐ KÉRDÉSEK A 6.1. FEJEZETHEZ

253. Milyen lehetőségei vannak a felhasználónak a 3D-s modelltérben?
254. *Melyek a vektorgrafika és rasztergrafika közötti legfontosabb különbségek?*
255. Egy kocka mozgatásával mutassa be a modelltér, a vektorgrafikus programcsomag és adatbázis kapcsolatát!
256. *Mit tartalmaz a geometriai objektumokról a vektorgrafikus adatbázis?*
257. *Milyen kapcsolatok lehetnek a geometriai objektumok között?*
258. Milyen mellérendelt kapcsolatok lehetnek a geometriai objektumok között?
259. *Mit nevezünk jelenetnek (scene)?*
260. *Milyen tulajdonságokat rendelhetünk a vektorgrafikus geometriai objektumokhoz?*
261. Mit értünk a vektorgrafikus adatbázisban tárolt tulajdonságok átöröklésén?
262. *Milyen adatokat tartalmaz a vektorgrafikus adatbázis?*
263. Mutasson be néhány példát a vizuális információkban megtestesülő szellemi értékek felhalmozódására!
264. *Mi a különbség az adat- és eljárásorientált geometriai modellezés között?*
265. *Milyen adatokat tárolunk az adatbázisban a geometriai modellekről?*
266. *Hogyan hozzuk létre általában a geometriai objektumokat a modelltérben?*
267. Milyen 2D-s primitíveket ismer?
268. *Sorolja fel a 3D-s primitíveket?*
269. *Milyen primitívekkel lehet modellezni a térbeli íveket és felületeket?*
270. Hogyan lehet a különböző geometriai modellezési módszereket együttesen alkalmazni?
271. *Milyen műveleteket végezhetünk a modelltér objektumaival?*
272. Hogyan hozhatunk létre új objektumokat a modelltérben?
273. *Milyen utasításokkal hozhatunk létre új objektumot primitívpéldányokkal?*
274. *Mit jelent a szerkesztéssel történő objektumdefiniálása?*
275. *Mit jelent a pásztázás?*
276. *Mit jelent a kihúzás?*
277. *Hogyan hozhatunk létre térbeli alakzatokat forgatással?*
278. *Mit jelent a másolás?*
279. *Milyen transzformációkat tesznek lehetővé a vektorgrafikus rendszerek?*

280. Milyen utasításokkal hajthatunk végre Boole-algebrai műveleteket vektorgrafikus objektumokkal?
281. *Mi a hatása az összeragasztás műveletének?*
282. Hogyan lehet objektumokat csoportosítani és ezt megszüntetni?
283. *Mi a lényege a fóliák alkalmazásának?*
284. *Mit adunk meg a KAMERA paranccsal?*
285. *Hogyan lehet megadni a vektorgrafikában a megvilágítási viszonyokat?*
286. *Mit jelent a WIREFRAME és a NET parancs?*
287. *Mit jelent a HIDDEN és a SHADE megjelenítés?*
288. Milyen a vektorgrafikus szoftverek felhasználói felülete?
289. Mit jelent az UCS
290. A vektorgrafikus felhasználói felületen megjelenített koordinátaértékek mire vonatkoznak?
291. *Hogyan lehet a vektorgrafikus objektumok helyzetére vonatkozó információkat közölni a vektorgrafikus rendszerrel?*
292. *Hogyan lehet egérrel kijelölni egy vektorgrafikus objektumot?*

? ELLENŐRZŐ KÉRDÉSEK A 6.1. FEJEZETHEZ

253. Milyen lehetőségei vannak a felhasználónak a 3D-s modelltérben?
254. *Melyek a vektorgrafika és rasztergrafika közötti legfontosabb különbségek?*
255. Egy kocka mozgatásával mutassa be a modelltér, a vektorgrafikus programcsomag és adatbázis kapcsolatát!
256. *Mit tartalmaz a geometriai objektumokról a vektorgrafikus adatbázis?*
257. *Milyen kapcsolatok lehetnek a geometriai objektumok között?*
258. Milyen mellérendelt kapcsolatok lehetnek a geometriai objektumok között?
259. *Mit nevezünk jelenetnek (scene)?*
260. *Milyen tulajdonságokat rendelhetünk a vektorgrafikus geometriai objektumokhoz?*
261. Mit értünk a vektorgrafikus adatbázisban tárolt tulajdonságok átöröklésén?
262. *Milyen adatokat tartalmaz a vektorgrafikus adatbázis?*
263. Mutasson be néhány példát a vizuális információkban megtestesülő szellemi értékek felhalmozódására!
264. *Mi a különbség az adat- és eljárásorientált geometriai modellezés között?*
265. *Milyen adatokat tárolunk az adatbázisban a geometriai modellekről?*
266. *Hogyan hozzuk létre általában a geometriai objektumokat a modelltérben?*
267. Milyen 2D-s primitíveket ismer?
268. *Sorolja fel a 3D-s primitíveket?*
269. *Milyen primitívekkel lehet modellezni a térbeli íveket és felületeket?*
270. Hogyan lehet a különböző geometriai modellezési módszereket együttesen alkalmazni?
271. *Milyen műveleteket végezhetünk a modelltér objektumaival?*
272. Hogyan hozhatunk létre új objektumokat a modelltérben?
273. *Milyen utasításokkal hozhatunk létre új objektumot primitívpéldányokkal?*
274. *Mit jelent a szerkesztéssel történő objektumdefiniálása?*
275. *Mit jelent a pásztázás?*
276. *Mit jelent a kihúzás?*
277. *Hogyan hozhatunk létre térbeli alakzatokat forgatással?*
278. *Mit jelent a másolás?*
279. *Milyen transzformációkat tesznek lehetővé a vektorgrafikus rendszerek?*

280. Milyen utasításokkal hajthatunk végre Boole-algebrai műveleteket vektorgrafikus objektumokkal?
281. *Mi a hatása az összeragasztás műveletének?*
282. Hogyan lehet objektumokat csoportosítani és ezt megszüntetni?
283. *Mi a lényege a fóliák alkalmazásának?*
284. *Mit adunk meg a KAMERA paranccsal?*
285. *Hogyan lehet megadni a vektorgrafikában a megvilágítási viszonyokat?*
286. *Mit jelent a WIREFRAME és a NET parancs?*
287. *Mit jelent a HIDDEN és a SHADE megjelenítés?*
288. Milyen a vektorgrafikus szoftverek felhasználói felülete?
289. Mit jelent az UCS
290. A vektorgrafikus felhasználói felületen megjelenített koordinátaértékek mire vonatkoznak?
291. *Hogyan lehet a vektorgrafikus objektumok helyzetére vonatkozó információkat közölni a vektorgrafikus rendszerrel?*
292. *Hogyan lehet egérrel kijelölni egy vektorgrafikus objektumot?*

6.2. VEKTORGRAFIKUS GEOMETRIAI MODELLEZÉS

A vektorgrafikus modellező eljárások közül ebben a fejezetben részletesebben megvizsgáljuk

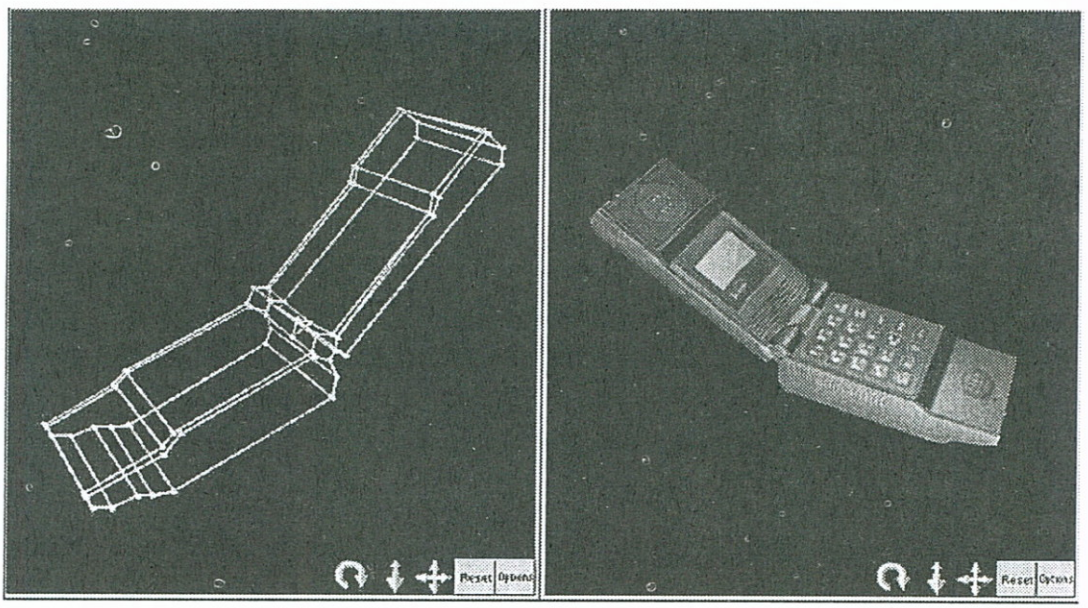
- a huzalvázas modellezést,
- a palástmodellezést (b-rep),
- a konstruktív tömör testmodellezést és
- a térfelosztással való modellezést.

Minden egyes modelltípusra összefoglaljuk a modell lényegi jellemzőit, a modellnek megfelelő adatbázis logikai szerkezetét, és az adott modell tipikus felhasználását a számítógépes grafikában.

6.2.1. Huzalvázmodellezés

A számítógépes grafikában legkorábban alkalmazott 3D-s modell a huzalvázas modell, mely lényegét tekintve a műszaki rajzok szabványain alapuló 2D-s vektorgrafikus rendszerek vonalas ábrázolásának a háromdimenziós általánosítása.

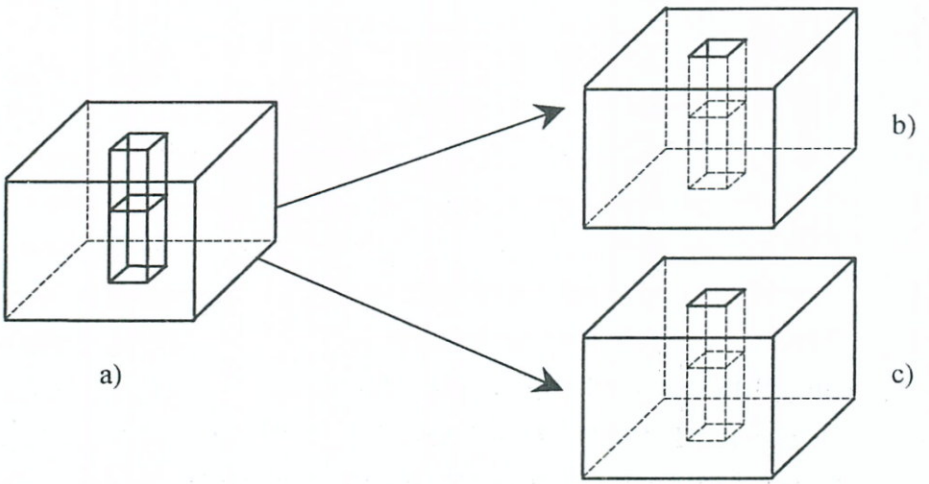
! *A huzalváz modell a 3D-s geometriai alakzatot csúcsaival és éleivel jellemzi, ennek megfelelően a modell csak a csúcsokat és az ezekhez rendelt összekötő éleket tartalmazza* (lásd 194. sz. ábra).



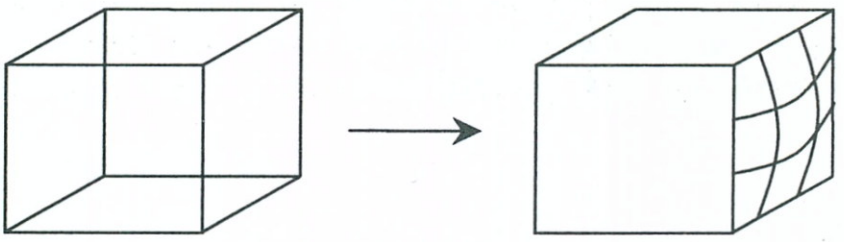
194. sz. ábra
Huzalváz modellje egy mobiltelefonnak

A huzalvázmodellek legnagyobb *előnye, hogy számítógépes megvalósításuk algoritmusigénye a többi geometriai modellezőmódszernél lényegesen kisebb*, így viszonylag kis erőforrású számítógépen is használhatók. (Így például ezeket a modelleket még a legelső mikroszámítógépgenerációk is képesek voltuk kezelni.)

A huzalvázmodellek legnagyobb *problémája, hogy egy huzalvázmodellnek több test is megfelelhet. Nem mindig tehető különbség a tömör és üreges test között a modell alapján, és a testet határoló felületek görbültségét sem tudjuk kezelni* (lásd 197. és 196. sz. ábrák).



195. sz. ábra
A huzalvázmodell nem jellemzi egyértelműen az üreges testet.
Az a) huzalvázmodell értelmezhető a b) és a c) testnek is.

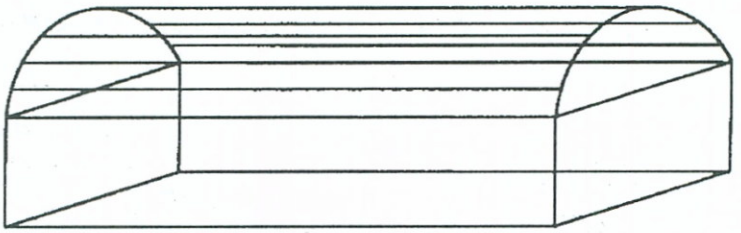


196. sz. ábra
A huzalvázmodellben nem fejezhető ki görbült határolófelület
Az a) huzalvázmodellnek megfelelő test oldallapja lehet sík és görbe felület is

Ezek a problémák miatt a huzalvázmodellezés a nem teljes értékű geometriai modellezések csoportjába tartozik, ami azt jelenti, hogy a modell nem tartalmazza a valós test leírásához szükséges összes geometriai és csatlakoztatási (topológiai) információt.

Ugyanakkor viszont a huzalvázmodell, ha a modellezendő testeket leszűkítjük a konvex poliéderek halmazára, teljes értékű modellnek tekinthető.

A szemléltetési és egyértelműségi problémák kezelésére egyes CAD rendszerekben a modellt továbbfejlesztették és a görbült felületeket felületvonalakkal szemléltetik. Ezt felületvonalas huzalváz modellnek nevezzük (lásd a 197. sz. ábra).



197. sz. ábra
Felületvonalas huzalvázmodell

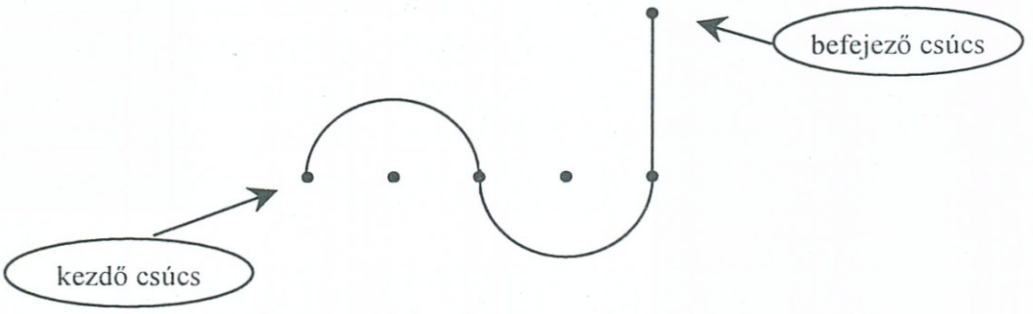
Ez a megoldás sem javította lényegesen a modell alkalmazhatóságát, mivel a vonalas görbült felületábrázolás nyilvánvalóan megjelenítésfüggő, ezért ezt minden kameraállásnál újra kell generálni.

A huzalvázmodell adatstruktúrájának lényege a csúcs, az él és az él–csúcs táblázatok együttese, melyeket a relációs adatbáziskezelés szabványainak megfelelően építenek fel.

Itt a csúcstáblázat tartalmazza a modellezett objektum egyes csúcsaihoz tartozó azonosítót és koordinátaértékeket, az él–csúcstáblázat meghatározza, hogy egy él milyen azonosítójú csúcsokat köt össze és hivatkozik az éltáblázatra, mely az élek generálásához szükséges információkat tartalmazza.

Megjegyezzük, hogy a huzalvázmodell adatstruktúrájának fentiekben történő bemutatása során a lényegkiemelés érdekében az adatbázis felépítését lényegesen leegyszerűsítettük. Így például:

- két csúcsot összekötő él különböző jellegű görbékből is felépülhet, ezért az éltáblázatnak ezekkel a részgörbékkel kapcsolatos összes információt is tartalmaznia kell. Ezek lehetnek például egyenes-szakaszokra, kúpszeletekre és B-spliné-okra vonatkozó paraméterek (lásd 198. sz. ábra);
- a szoftverek általában megengedik, hogy a huzalvázmodell több részmodellből épüljön fel, ezt a gyakorlatban használt adatbázis-struktúra megfelelő kapcsolótáblákkal és azonosítókkal fejezi ki.



198. sz. ábra
A huzalvázmodell két csúcsát összekötő él felépítése részgörbékből
(két félkörív és egy egyenesszakasz)

A huzalvázmodelleket – annak ellenére, hogy nem adják meg egy geometriai objektum teljes értékű leírását – napjainkban is elterjedten alkalmazzák. Felhasználásukra a legjellemzőbb példák a következők:

- műszaki tervezőrendszerekben a huzalvázmodell alkalmazásával jól kezelhetők például az alakzatok bonyolultabb felülethatárgörbéi,
- a huzalvázmodellezés igényli az összes modellezőmódszer közül a legkisebb memória és processzorkapacitást. Ezért még a fejlettebb geometriai modellezőrendszerek felhasználása esetén is célszerű a 3D-s objektumokkal való munka során a részeredményeket, munkapéldányokat huzalvázasan kezelni.

A műszaki tervezés gyakorlatában a rajzoláshoz a tervezőmunkában munkaeszközként legtöbbször a huzalvázmodellezést és megjelenítést alkalmazzák, mivel ennek gyorsasága lehetővé teszi a számítógépre történő várakozásnélküli interaktív tervezést.

6.2.2. Palástmodellezés

A palástmodellezésnél a geometriai objektumokat a vektorgrafikus modelltérben határolófelületeikkel (beleértve e felületek csatlakoztatására vonatkozó adatokat is) jellemezzük.

Ez a műszaki gyakorlatban előforduló testek esetében teljes értékű geometriai modellezést jelent, mivel a testeket a határoló felületeik vagy palástjuk pontjainak összessége a szükséges pontossággal és egyértelműséggel leírja.

Ennek a modellezési módszernek a neve az angol szakirodalomban boundary-representation, vagy röviden b-rep.

A palástmodellezés részét képezi a huzalvázmodellezésnél megismert csúcs, él együttes, de ez kiegészül a testhatároló lapok felületfoltjaival (lásd III. fejezet) és ezek geometriailag pontos csatlakoztatására vonatkozó adatokkal és algoritmusokkal.

A palástmodellező eljárásokat a lapok geometriai jellemzői szerint szokták osztályozni:

- ha a palástot képező lapok síkbeli sokszögekből állnak, akkor poliédermodelleket kapunk,
- ha a palástot képező lapok változó görbületű felületfoltok is lehetnek, akkor valósághű palástmodellezésről beszélünk.

Palástmodelleket létrehozhatunk *lépésenkénti szerkesztéssel. Ekkor a test határoló felületeit egyenként definiáljuk a térbeli felületek csatlakoztatási lehetőségeinek függvényében.*

A lépésenkénti szerkesztés legegyszerűbb elemei lépéseit Euler-operátoroknak nevezik. Ezek például többek között a következők lehetnek:

- képezz csomópontot, élet és palástot,
- kapcsolj ki (törölj) csomópontot és élet.

Ezekkel az operátorokkal a lényegüket tekintve „konvex poliéderekhez hasonló" testek létrehozása egyértelműen leírható.

A „konvex poliéderekhez hasonló" fogalmat matematikailag kissé precízebben kifejtve, azt lehet mondani, hogy minden olyan b-rep test létrehozható Euler-operátorok alkalmazásának véges sorozatával, mely eleget tesz a következő feltételeknek:

- a véges számú felületfolt egyesítése a test határfelületét adja,
- éleit mindig két csúcs zárja le,
- az él két (vagy páros számú) határoló felületfolt találkozásakor keletkezik,
- minden felületfoltot élek véges számú, záródó sorozata határol,
- konvex.

Ekkor igaz a b-rep testekre Euler poliédertétele: lapszám + csúcsszám = élszám + 2.

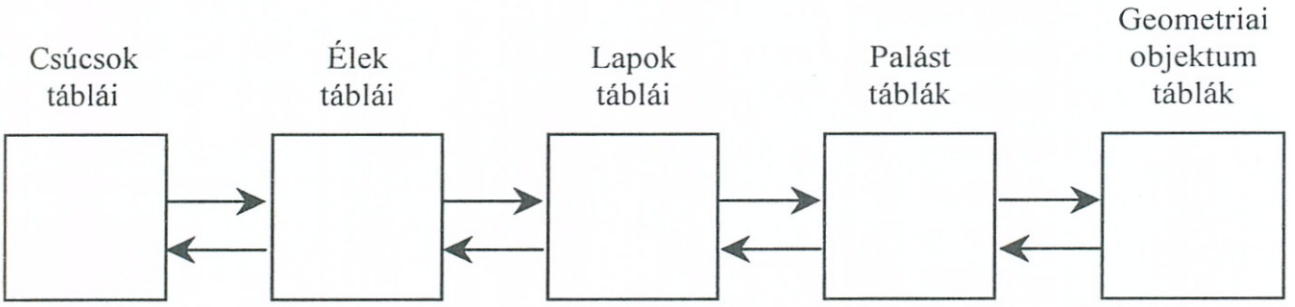
Ezt először konvex poliéderekre (tehát sokszöglapokkal határolt testekre) bizonyították, de ez a gömbbel homeomorf topológiai alakzatokra is általánosítható. Ezért az Euler-operátorokkal felépíthetünk görbült felületfoltokat tartalmaz b-rep testeket is.

A műszaki gyakorlatban azonban különböző üreges elemek, térbeli „lyukak" képzésére is szükségünk van. Ehhez úgynevezett kiegészítő operátorokat alkalmaznak, melyek az egyszeresen vagy többszörösen lyukas (torusz vagy többszörös torusszal egyenértékű) testek létrehozását is biztosítják.

A lépésenkénti szerkesztés mellett a b-rep modellek létrehozásának fontos eszköze a 6.1.4.1. pontban már bemutatott pásztázás.

A műszaki tervezőrendszerek fejlettebb formáinál alkalmazott *„valósághű palástmodellezés" a testet behatároló felületfoltok generálása és illesztése során a B-spline, a Bézier és NURBS felületfoltok alkalmazását is lehetővé teszi.*

A palástmodellek lényegi, egyszerűsített adatstruktúrájának relációs modelljét a 199. sz. ábra mutatja be.



199. sz. ábra
A palástmodellek adatstruktúrája

Mivel – mint ez az ábrán is látható – az egyes entitások N:M kapcsolatban állnak az adatstruktúra ennek megfelelő számú csúcs–él, él–lap stb. kapcsoló táblát is tartalmaz. *Az adatbázis szerkezete egyben a b-rep modellezés lényegét is kifejezi: a felületfoltokat nem egyszerűen egymás mellé rakjuk a modellben, hanem ezek kapcsolata beépül az adatbázisba.*

A palástmodellezést elterjedten a különböző CAD rendszerekben alkalmazzák, mivel a 3D-s műszaki tervezésben szükséges testek teljes körű geometriai leírását biztosítja és lehetővé teszi a felhasználó számára a modellezés során a testek „kényelmes", interaktív szerkesztését.

Elterjedését a poliéder modellek parametrizálhatósága is elősegítette (azaz könnyen kialakíthatók csak egy-két geometriai paraméterben eltérő testcsaládok pl.: fogaskerekek), valamint az is, hogy a palástmodellek huzalváz-modellé való konvertálásához szükséges algoritmusok rendelkezésre állnak. A felületek teljes értékű geometriai leírása különösen előnyös lehet az NC (számjegyvezérlésű) automaták programozásában.

6.2.3. Testmodellezés

Testmodellezés esetén feltételezzük, hogy a modellezendő objektumok olyan merev testek (tehát például a modelltérben való mozgatás során alak-

jukat nem változtatják), melyek a palástjukkal határolt teret teljesen kitöltik (tömör testek).

A testmodellezési eljárások közül a számítógépes grafikában két modellezési módszer vált elterjedté:

- *elemi testekkel és ezek közötti szabályos halmazműveletekkel való modellezés,*
- *a testek elemi sejtekből való felépítése.*

A véges számú tömör elemitest primitívből kiinduló és a modellt a metszet, egyesítés, kivonás és ragasztás halmazműveletek egymás utáni felhasználásával megkonstruáló modellezési módszert konstruktív tömör testmodellezésnek nevezzük. Ennek angol elnevezése: Constructive Solid Geometry, vagy röviden CSG.

A térfogat modellezési (Volumetric Modeling) módszerek közül a legelterjedtebben használatos a testeket elemi sejtekből felépítő modellezési eljárás (Cell Modeling).

6.2.3.1. Konstruktív tömör testmodellezés (CSG)

A CSG két alkotóeleme

- a kiinduló tömör elemi testprimitívek készlete
- a megengedett halmazalgebrai műveletek eszközkészlete.

A standard testprimitívek induló készlete általában a már bemutatott 3D-s primitívekből

- *a hasáb,*
- *a gúla,*
- *a henger,*
- *a kúp és*
- *a gömb*

áll.

Egyes rendszerek megengedik, hogy a felhasználó ezt az induló testkészletet az általa definiált saját primitívekkel is kiegészítse.

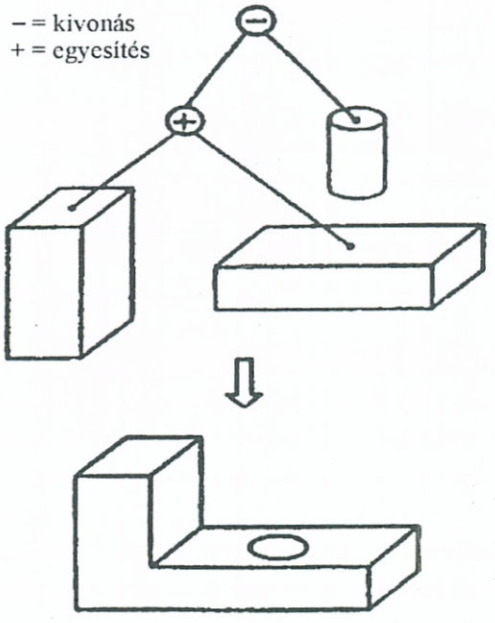
Ezek a primitívek parametrizáltak és helyfüggőek, ami azt jelenti, hogy a felhasználó egy konkrét példányt a geometriai nagyságparaméterek és a világkoordináta-rendszerben való hely megadásával hozhat létre. Például: r=10 sugarú gömb, melynek középpontja (0, 0, 1).

A konstruktív tömör testmodellezést alkalmazó vektorgrafikus szoftverek egy speciális parancsnyelvet használnak, mely a 6.1.4. pontbeli objektummműveletek többségét lehetővé teszi:

- *konkrét primitívpéldány létrehozása,*
- *objektum másolása, törlése, transzformálása,*

- objektumok halmazalgebrai metszete, egyesítése, kivonása, összeragasztása.

A kiinduló primitívkészlet és az előző parancsok ismételt alkalmazásával viszonylag bonyolult testeket is létre lehet hozni a CSG-ben rövid idő, és – például a b-rep-el összehasonlítva – jóval kevesebb adat közlésével (lásd 200. sz. ábra).



200. sz. ábra
Testmodell létrehozásának lépései CSG-ben

A CSG-nek megfelelő matematikai struktúra a következő:

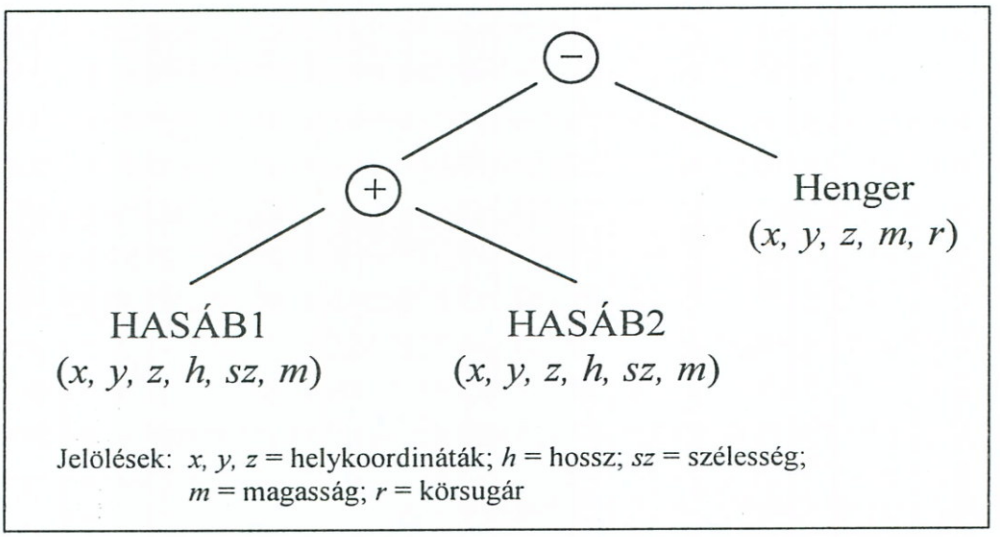
Legyenek adottak a P_1, P_2, ... P_n véges kiterjedésű, zárt, reguláris ponthalmazok a 3D-s euklidészi térben és az ezek között értelmezett

$M = \cup, \cap, \setminus, \otimes$ műveletek (egyesítés, metszet, kivonás és kontanáció)

A P_i-kből M végesszámú alkalmazásával képzett T testek alkotják a CSG-vel generálható objektumok halmazát. Meg kell jegyezni, hogy egy ilyen test előállítása nem szükségképpen egyértelmű, azaz ugyanazt a testet több különböző műveletsorozat is előállíthatja.

Legtöbb modellező szoftver véges kiterjedésű P_i primitívekből indul ki. Egyes rendszerek – általában a véges primitívek használata mellett – félterek, mint primitívek alkalmazását is megengedi. (Például 6 féltérrel definiálható egy hasáb.) A félterek használata akkor előnyös, ha sűrűn kell az adott felhasználásban testek síkmetszetekkel képzett áthatását kiszámolni. A féltér primitívek alkalmazása ugyanakkor érvényességi problémát is okoz, ugyanis félterek metszete nem mindig eredményez zárt tömör testet.

A konstruktív tömör testmodellezéssel létrehozott modellek adatstruktúrájára a bináris fa gráf jellemző, melynek ágcsomópontjai a halmazalgebrai műveletek, levelei pedig a műveletben részt vevő testek (lásd 201. sz. ábra).



201. sz. ábra
CSG-vel létrehozott modell bináris fastruktúrája

Ezt a fastruktúrát a szoftverek jelentős része képernyőn is megjeleníti és így a felhasználó számára is hozzáférhetővé teszi.

A CSG továbbfejlesztéseként jött létre a testpalást modellezés vagy Constructive Shell Representation: CSR, mely a CSG és a b-rep palástmodellezés előnyös tulajdonságait próbálja meg ötvözni:

- a modellezésben részt vevő testeket fastruktúrával írja le,
- a levelek b-rep modellek is lehetnek.

A CSG a 3D-s modelltér objektumait az ember valós térérzékeléséhez hasonló módon kezeli és szemlélteti, egyrészt ez magyarázza ennek a modellezési módszernek a széleskörű elterjedését.

A testmodellező rendszereket interaktív és kötegelt üzemmódban egyaránt alkalmazzák. Az előző esetben a testszerkesztést a képernyőn folyamatosan követhetjük és a parancsnyelvvel lépésenként vezéreljük.

Kötegelt üzemmódban (batch processing) a modellépítést a felhasználó teljes körűen definiálja a parancsnyelvvel és azt egy fáljban adja át a vektorgrafikus rendszernek, mely ennek alapján általában a 3D-s objektum képi változatát állítja elő.

! *A konstruktív tömör testmodellezésnek a gyakorlatban történő jól használhatóságára* hívják fel a figyelmet azok a vizsgálatok, melyek eredményei szerint *a műszaki tervezés során szükséges testek döntő része előállítható néhány egyszerű geometriai test* (például hasáb és henger) *megfelelő kombinációjából.*

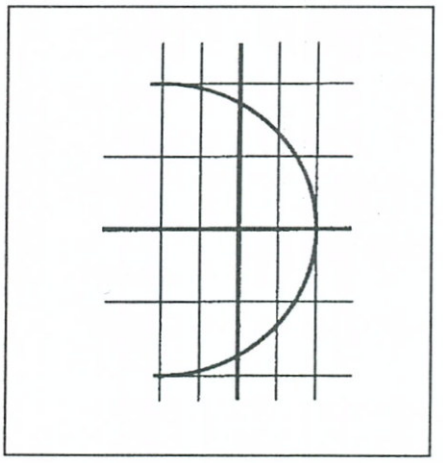
6.2.3.2. Térfogatmodellezés elemi sejtekkel

A térfogatmodellezésnél egy tömör tárgyat több egymáshoz csatlakozó, de egymást nem metsző kisebb tömör tárgyra, azaz sejtekre bontunk fel. Az elterjedtebb modellezési módszerek a sejtek két típusát kezelik: !

- *a sejtek azonos típusú alakzatok* (például hasábok), *de méretük egy paramétertől függően változhat,*
- *a sejtek azonos típusú és méretű alakzatok, ekkor ezeket* (a képponthoz, azaz a pixelhez hasonlóan) *voxelnek* (volument element) *nevezzük.*

Az előzőnek tipikus képviselője a hasábnyolcadolásos lebontást (oct-tree decomposition) alkalmazó eljárás, míg utóbbinak a térkitöltéses felsorolás (Spatial-occupancy enumeration).

A hasábnyolcadolásos eljárásnál a modellezni kívánt objektumot egy térbeli hasábba foglaljuk bele, majd a hasábot oldallapjaival párhuzamos síkokkal megfelezzük. Ez alapján a hasábból 8 db részhasábot állítunk elő (innen kapta a módszer a nevét). Ezt követően a felbontást a részhasábok nyolc részre vágásával növeljük. Mindezt az eljárást addig folytatjuk, míg a modellezni kívánt test kellő pontosságú közelítését el nem érjük. Az eljárás miden egyes lépésénél rögzítjük, hogy a modelltest egy részhasábot teljesen vagy részben kitölt, illetve ezzel nincs közös része. Az eljárás lényegét 3D helyett 2D-ben szemléltetve mutatja a 202. sz. ábra. (Itt nyilván 8 helyett 4 részalakzatot kapunk.)

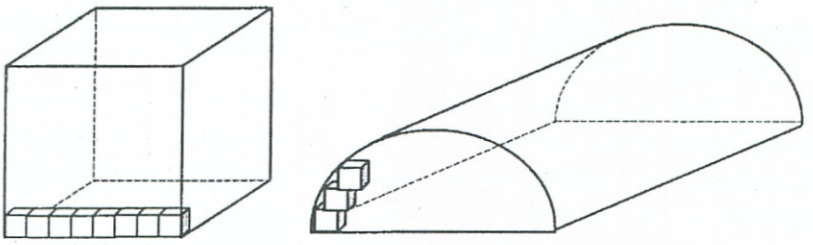


202. sz. ábra
Hasábnyolcadolásos térfogatmodellezés szemléltetése síkban

Az azonos formátumú és méretű voxelekkel való kitöltése a modellezendő testnek, ha a voxeleket elegendő kis méretűre választjuk, a test relatíve pontos leírását eredményezi. A leggyakoribb voxeltípus a kocka. A modelle- !

zendő objektumokat a voxelekkel úgy írjuk le, hogy minden egyes, a testhez teljes egészében, vagy csak részlegesen hozzátartozó voxel adatait hozzárendeljük a testhez.

Ezzel a módszerrel a 2D-s raszteres képek pixeljeihez hasonlóan a voxelekkel a testek egy megfelelő pontosságú modellezését tudjuk elérni. Ugyanakkor e nagyszámú voxel (ezek száma a test 3D-s méreteinek köbével arányos) feldolgozása a *számítógéptől nagy tárolókapacitást és processzorteljesítményt igényel.* Emiatt a voxelekkel történő térfogatmodellezést ma még elterjedten nem használják, ugyanakkor több szakértő szerint a számítógépek teljesítményének növekedésével a jövőben a voxeles modellezés meghatározó szerepet tölthet be a számítógépes grafika geometriai modellezési módszerei között.



203. sz. ábra
Voxelekkel történő térfogatmodellezés

ELLENŐRZŐ KÉRDÉSEK A 6.2. FEJEZETHEZ ?

293. *Mit értünk huzalváz modell alatt?*
294. *Mi az előnye a huzalvázmodell alkalmazásának?*
295. *Milyen korlátai vannak a huzalvázmodell alkalmazásának?*
296. Milyen testekre ad a huzalvázas modell teljes értékű leírást?
297. Miben különbözik a huzalvázmodell a felületvonalas huzalvázmodelltől?
298. *Mit tárolunk az adatbázisban huzalvázmodell esetén?*
299. Miért alkalmazzák a műszaki tervezés gyakorlatában a huzalvázmodellezést?
300. *Hogyan jellemezzük a geometriai objektumokat palástmodellezésnél?*
301. *Mi a b-rep?*
302. Mi különbség van a poliédermodellek és valósághű palástmodellezés között?
303. *Mit értünk lépésenkénti szerkesztés alatt?*
304. Milyen testeket írhatunk le egyértelműen Euler operátorokkal?
305. Hogyan modellezhetjük a térbeli üreges testeket?
306. *Mivel generáljuk a testeket határoló felületeket valósághű palástmodellezésnél?*
307. *Milyen táblákból áll a palástmodell adatbázisa?*
308. Mire használható a palástmodellezés a műszaki gyakorlatban?
309. *Milyen két típusa van a tömör testmodellezésnek?*
310. *Mi a CSG modellezés lényege?*
311. Melyik a legelterjedtebben használatos térfogat modellezési módszer?
312. *Milyen standard testprimitívek vannak a CSG-ben?*
313. Hogyan lehet definiálni egy testprimitívet a CSG-ben?
314. *Milyen műveletek alkalmazhatók a CSG-ben?*
315. *Milyen a CSG modellek adatstruktúrája?*
316. Hogyan használhatjuk a tömör testmodellezést interaktív, illetve kötegelt üzemmódban?
317. *Miért használható a tömör testmodellezés a műszaki tervezésben?*
318. *Mit nevezünk voxelnek?*
319. *Hogyan modellezhetjük a testeket voxelekkel?*

6.3. A MODELLTÉR JELENETEINEK ÁBRÁZOLÁSA RASZTERES KÉPEN

A felhasználó a vektorgrafikus szoftverekkel való munkavégzés során az előző fejezetben megismert modellezési módszerekkel definiál a modelltérben vektorgrafikus objektumokat, ezekkel műveleteket végez, közöttük kapcsolatokat határoz meg. Mindezek meghatározott formátumban a vektorgrafikus adatbázisban tárolásra kerülnek és ilyen értelemben a modelltér objektumai a 3D-s világkoordináta-rendszerben „objektíve" a képi megjelenítésüktől függetlenül léteznek.

A vektorgrafikus rendszerekkel végzett munka meghatározott fázisaiban szükségünk van a modelltér elemeinek képi megjelenítésére is. Ez az eset áll elő például, ha

- a vektorgrafikus objektumokkal végzett műveletek eredményét szeretnénk megtekinteni, vagy ha
- a vektorgrafikus rendszerrel végzett munka végeredménye önmagában képi jellegű, például egy filmes animáció.

A modelltérnek egy adott nézőpontból látható és a képi megjelenítés szempontjából összetartozó objektumait jelenetnek nevezzük.

A jeleneteket képen – a megjelenítő eszköztől és a megjelenítés céljától függően – raszteresen vagy vektorosan ábrázolhatjuk. Tipikus raszteres képmegjelenítő eszköz a monitor és a nyomtató, a vektoros rajzok készítésére pedig a rajzgépet vagy plottert használhatjuk (lásd VIII. fejezet). A továbbiakban a vektorgrafikus objektumok raszteres képen történő megjelenítésével fogunk foglalkozni, de lényegét tekintve a tárgyalt eljárások a vektoros megjelenítésre is vonatkozhatnak azzal a megjegyzéssel, hogy ebben az esetben a képgeneráló pipeline (lásd 6.3.4. fejezet) utolsó lépése a vektor–raszter konverzió nyilvánvalóan elhagyásra kerül.

A vektorgrafikus objektumainkat raszteres képen ábrázolhatjuk

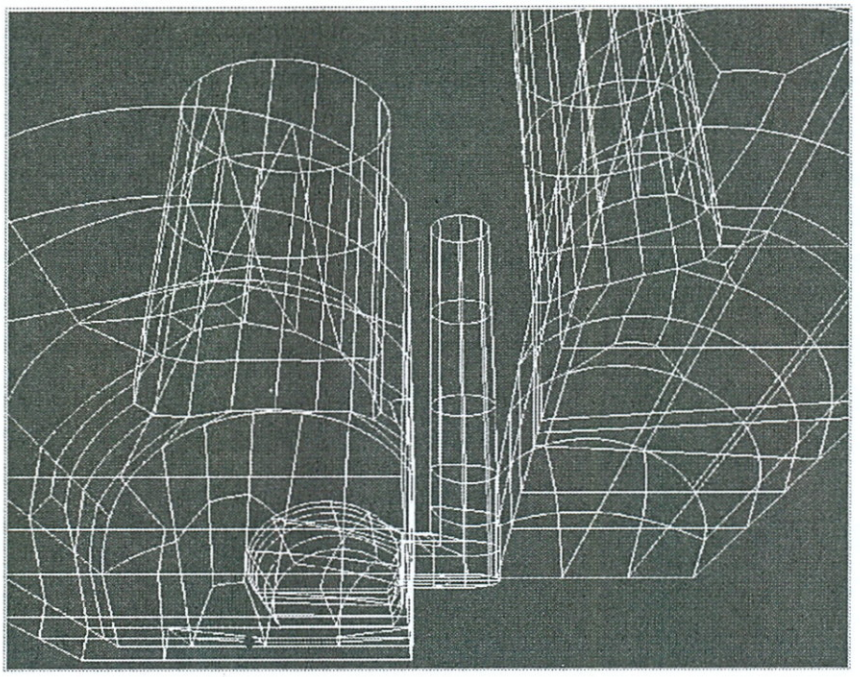
- huzalvázas módon,
- árnyalt módon és
- fotorealisztikus módon.

Ez a sorrend egyben a vektorgrafikus megjelenítési technikák fejlődését is tükrözi.

6.3.1. Huzalvázas ábrázolás (wireframe)

A vektorgrafikus objektumok *huzalvázas vagy drótvázas megjelenítése* a legegyszerűbb, ugyanakkor a legkevésbé valósághű. Ez esetben *a testeket éleikkel ábrázoljuk, azaz a testeket a képen úgy jelenítjük meg, mintha drót-*

ból készültek volna. Az ábrán nincsenek takart vonalak, minden él teljes egészében megjelenítésre kerül (lásd 204. sz. ábra).



204. sz. ábra
A modelltér jelenetének ábrázolása huzalvázas formában

A huzalvázas megjelenítés műveleti igénye nagyságrendileg kisebb a különböző árnyalt és fotorealisztikus képeket előállító algoritmusokénál. Ezért főként ezt a megjelenítési formát alkalmazzuk például CAD rendszerekben, amikor a munka során az egyes részeredményeket képernyőn megnézve is ellenőrizzük. Ennek oka, hogy a huzalvázas kép kirajzolása képernyőre még viszonylag kisebb teljesítményű PC esetében is gyakorlatilag azonnal megtörténhet, míg ugyanennek az objektumnak valósághű képen történő megjelenítése percekig is eltarthat.

Ismételten felhívjuk a figyelmet arra, hogy a huzalvázas megjelenítést nem szabad azonosítani a vektorgrafikus objektumok huzalváz modellezésével. Ha például b-rep vagy CSG modellekkel dolgozunk, akkor is szokásos – a képernyőre való kirajzolás sebessége miatt – a részmunkafázisok eredményét huzalvázas formában megjeleníteni.

6.3.2. Árnyalt megjelenítés

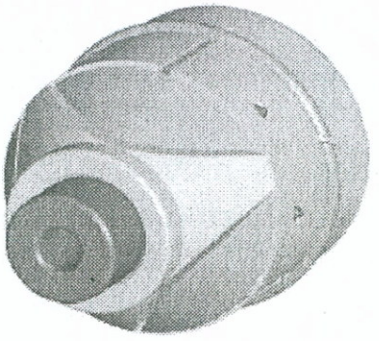
A jelenetek sokkal realisztikusabb képi megjelenítését kapjuk a testek felületének árnyalt ábrázolásával. Ekkor a képernyőre nem az objektumok átlát-

szó vázát, hanem határoló felületeik kitöltött képét rajzoljuk ki. *Az árnyalással azt próbáljuk kifejezni, hogy a természetben látható megvilágított felületek a fényforrások és a felület térbeli helyzetéből és anyagi minőségéből, valamint a néző elhelyezkedésétől függően különböző fényességűnek látszanak. Ekkor már a nézőpont és az objektumok modelltérbeli elhelyezkedése alapján a kirajzolásnál a takarási viszonyokat is figyelembe vesszük. Ez azt jelenti, hogy a képen egy test vagy felület által eltakart élek és testrészek nem fognak megjelenni.*

Az árnyalt képet előállító algoritmusoknak az előbbiek miatt elég komplex feladatot kell megoldaniuk, és minél élethűebb egy árnyalt kép, az előállításához szükséges megjelenítési idő annál nagyobb.

Azt az eljárást, melynek segítségével a modelltér objektumait tartalmazó jelenetekből árnyalt képet állítunk elő, renderelésnek (rendering) nevezzük.

Egy gépalkatrész rendereléssel készített árnyalt képet szemlélteti a 205. sz. ábra.



205. sz. ábra
Gépalkatrész (kupplung) árnyalt megjelenítésű képe

Az árnyalt megjelenítést a legkorábbi vektorgrafikus szoftverek kezdetben viszonylag egyszerű algoritmusokkal oldották meg. Ilyen volt például a flat shading, melynél az objektumok felületeit alkotó sokszögeket azonos színnel töltötték ki, melyet a fényforrás helyzetétől függően a sokszöglap megvilágításerősségének megfelelően módosítottak.

A renderelő algoritmusok a vektorgrafikus szoftverekkel és a grafikus hardver teljesítményével együtt fejlődtek, és a számítógépes grafikában a valós világot egyre jobban megközelítő képeket generáló eljárások kidolgozása az egyik legfontosabb kutatási területté vált.

6.3.3. Fotorealisztikus megjelenítés

Fotorealisztikus megjelenítés alatt a számítógépes grafikában azt értjük, hogy a modelltérbeli jelenetről olyan minőségű képet állítunk elő egy vektorgrafikus rendszerrel, mely teljesen valószerű és a valós világról készített fényképtől nem különböztethető meg.

A fotorealisztikus megjelenítés különösen a számítógépes grafikával készített filmek, valamint a szimulációs kiképzés esetében alapkövetelmény, de ma már a különböző vektorgrafikus tervező és rajzoló szoftverek is lehetővé teszik a fotorealisztikus renderelést. (Például egy építész által megtervezett lakóház foto minőségű képének előállítása esetén.)

A fotorealisztikus megjelenítés definíciójához két megjegyzést kell fűzni:

- nem mindig szükséges a számítógépes grafikában a nagyon erőforrásigényes fotorealisztikus megjelenítést alkalmazni. Például, ha oktatási célból meghatározott geometriai testeket akarunk szemléltetni, akkor erre a célra az egyszerűbb árnyalt megjelenítést biztosító algoritmusok is teljesen kielégítőek;
- a felhasználások egy részénél a fotorealisztikus ábrázolás szándékosan nem valószerű: például elképzelt sci-fi világokról készített filmeknél, vagy a realisztikus képábrázolás esztétikai okokból történő torzítása esetében.

A fotorealisztikus megjelenítés több követelményt is támaszt a képalkotással szemben. Ezek közül foglaljuk össze az elkövetkezendőkben a legfontosabbakat.

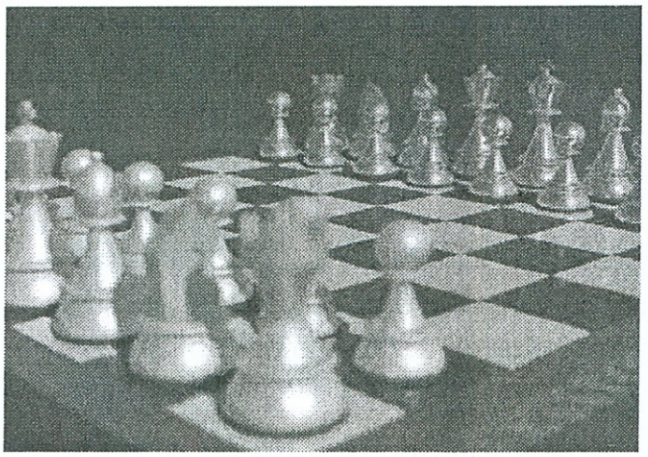
6.3.3.1. Térhatású képek előállítása (Depth Cueing)

A fotorealisztikus megjelenítés egyik fontos követelménye, hogy a 3D-s modelltér jelenete a 2D-s raszteres képen is térhatású legyen. A térhatás elérésének legfontosabb eszközei a következők:

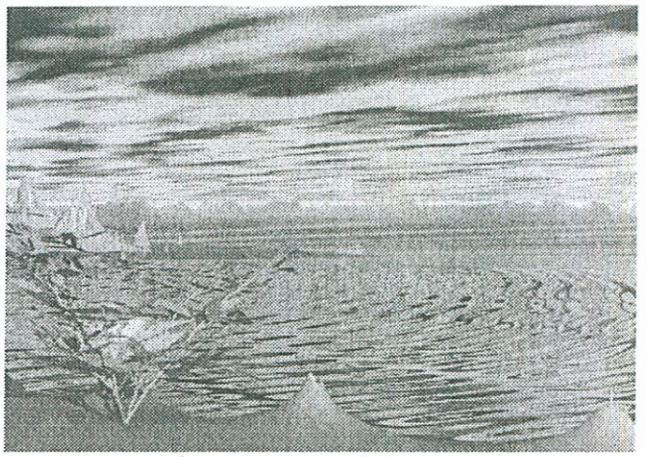
- *A nézőponttól távolodva a képen a párhuzamos egyeneseknek fokozatosan összetartozóknak kell lenniük, és a távolabbi testek méreteit arányosan csökkenteni kell.* Ezeket a hatásokat a megjelenítés során megfelelő perspektíva transzformációkkal érhetjük el.
- A 2D-s képen szerepeltetett jelenet objektumai egy adott nézőpontból szemlélve takarhatják egymást. Fontos tehát, hogy *a képen reálisan ábrázoljuk a tárgyak látható és nem látható éleit, felületeit.*

- A nézőtől távoli objektumok már nem látszanak olyan tisztán, mint a közeliek. Ezért a *képen a „messzeségbe tűnő" tárgyaknak elmosódottabbaknak és kevésbé részletesen kidolgozottaknak kell lenniük.* Ezt a hatást elérhetjük a színintenzitások megfelelő változtatásával, illetve a textúráknak a nézőponttól való távolság függvényében történő alkalmazásával (például mip-mapping, lásd 6.3.8.3. pont).

Ehhez hasonló hatást vált ki az ún. kamera modellek alkalmazása. Ezek esetében a jelenetben szereplő összes objektumot nem látjuk élesen, hanem csak azokat, melyekre a „kamera lencséinek fókusztávolsága" be van állítva.



206. sz. ábra
Térhatás elérése perspektív ábrázolással és a takart felületek kiküszöbölésével



207. sz. ábra
A nézőtől távoli táj érzékeltetése

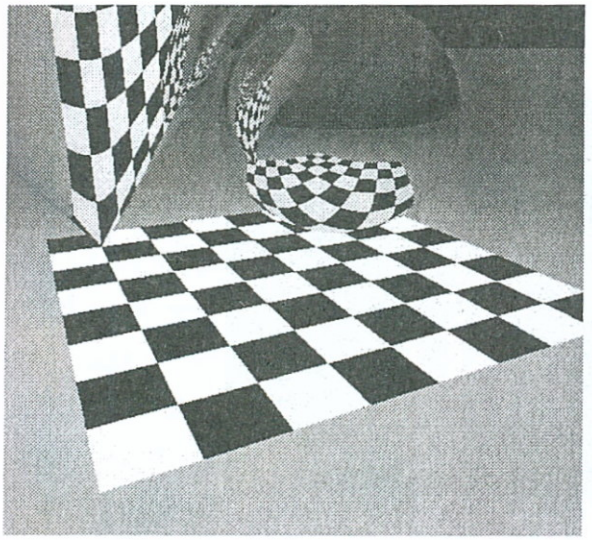
6.3.3.2. Felületek megvilágítása, tükröződés, árnyékok

! *A realisztikus képek készítése során a látható felületeket saját tulajdonságaiknak (szín, fényvisszaverőképesség, az anyaguknak megfelelő felületi részletek) és a modelltérben elhelyezett fényforrásoknak megfelelően kell árnyalnunk.*

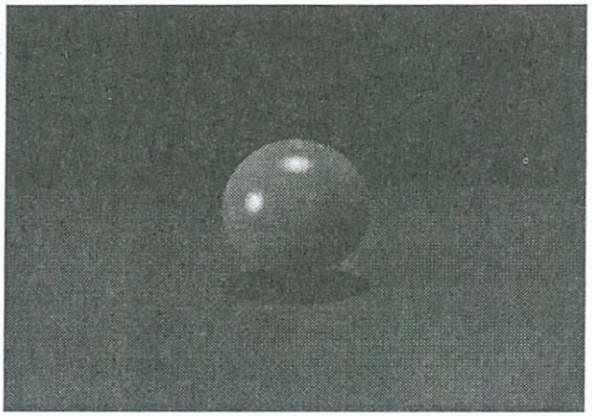
Ennek során tehát meg kell különböztetnünk a *matt* és a *tükröző* felületeket (lásd 208. sz. ábra).

Az árnyékolt felületek sokkal szebb képét kaphatjuk azáltal, ha a *különböző színű és intenzitású felületek csatlakozásánál az éles átmeneteket* megfelelő átlagolással *elmossuk.*

A nem matt felületű testeken a *fényforrások fényfolt formájában* jelennek meg, ezt az effektet fotorealisztikus képek előállításánál szintén kezelni kell (209. sz. ábra).



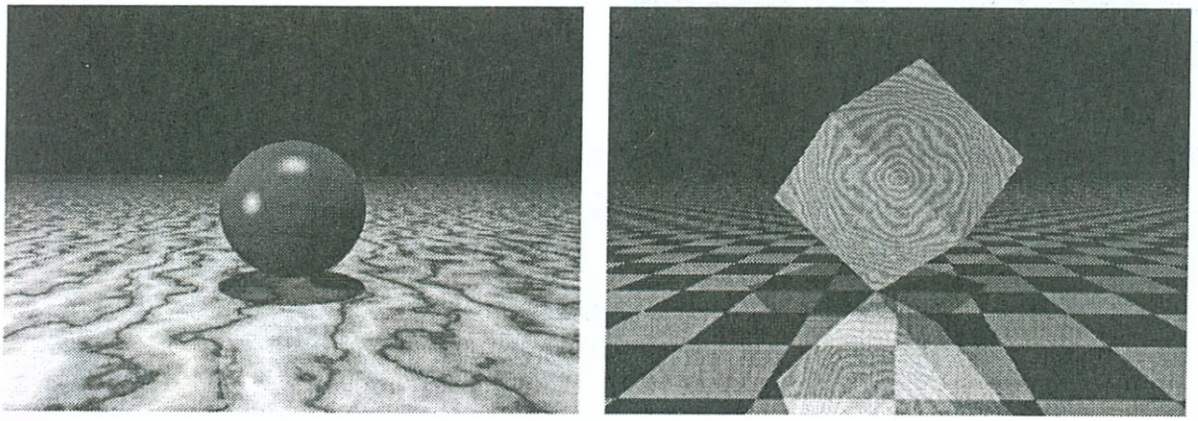
208. sz. ábra
Tükröző tárgyak ábrázolása



209. sz. ábra
Fényforrás megjelenítése fényfoltként tükröző felületen

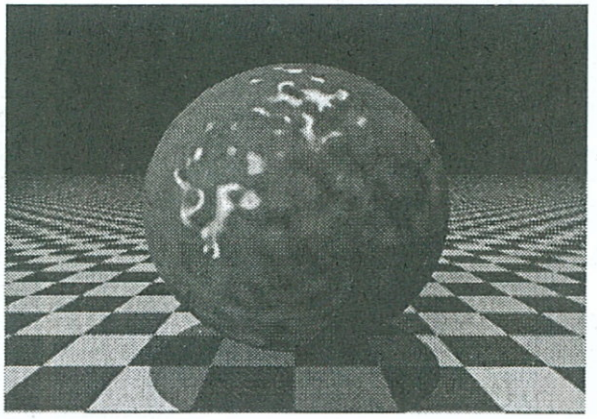
Például TV-kamerával felvett jeleneteknél fordulhat elő, hogy egy fényforrás megcsillan és a nézőt vakítja. Ezt a hatást modellező effektus az ún. lencse csillogás (lens flare).

A testek anyagi minőségének megfelelő képét textúrákkal állítjuk elő. Márvány (Marble) és fa (Wood) felület modellezésére mutat példát a 210. sz. ábra.



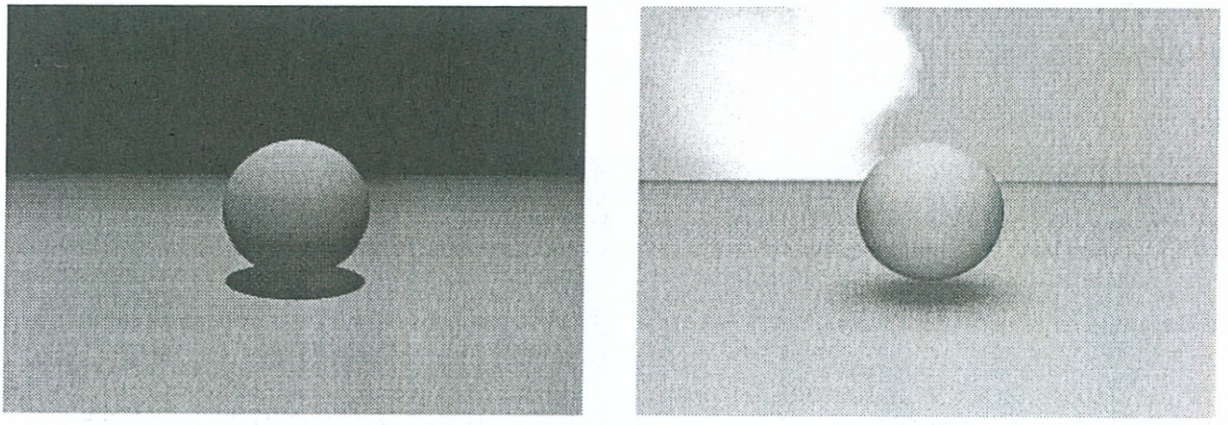
210. sz. ábra
Márvány, fa felületű testek modellezése

Érdes, göröngyös térhatású felületeket tudunk elérni a bump-mapping alkalmazásával, amikor a felületre merőlegesen véletlenszerűen módosítjuk a tárgy felszínét: egyes részeit kiemeljük vagy lesüllyesztjük (lásd 211. sz. ábra).



211. sz. ábra
„Göröngyös" felület létrehozása bump-mappinggel

A fotorealisztikus hatás eléréséhez elengedhetetlen, hogy a testek egymásra vetett árnyékát (shadows) is megjelenítsük a képen. Itt külön kell választanunk a pontszerű és a kiterjedt fényforrások által létrehozott árnyékokat, mivel előző esetben éles, az utóbbi esetben viszont elmosódott az árnyékhatár (lásd a 212. sz. ábra).



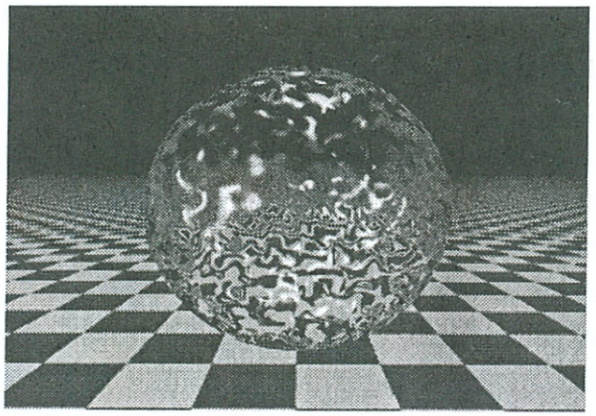
212. sz. ábra
Pontszerű és kiterjedt fényforrás árnyéka

6.3.3.3. Átlátszóság, köd és füst modellezése

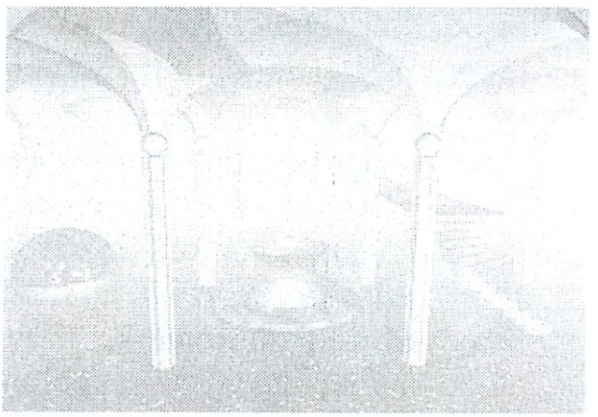
Az átlátszóság modellezésénél figyelembe kell vennünk a fénytörést, a diffúz áttetszőséget és a fény intenzitásának csökkenését az átlátszó testen történő áthaladás során (lásd 213. sz. ábra).

Az átlátszóság modellezésében jelentős segítséget jelenthet a textúrák használata. Erre legjobb példa az áttetsző víz ábrázolása, amikor a víz felszínének textúráját részlegesen „átlátszóvá tesszük" és ez folyamatosan átmegy a vízfenéknek megfelelő textúrába. Ezt az eljárást alpha-blending-nek nevezzük. Örvénylés, hullámzás ábrázolásához két textúra már kevés, ezt az effektet csak multitextúrázással tudjuk kezelni.

Ködöt fotorealisztikus képeken úgy tudunk ábrázolni, hogy a ködnek *megfelelő színt (általában fehér vagy fehéresszürke) a nézőponttól távolodva egyre nagyobb arányban keverjük a képhez* (lásd 214. sz. ábra).



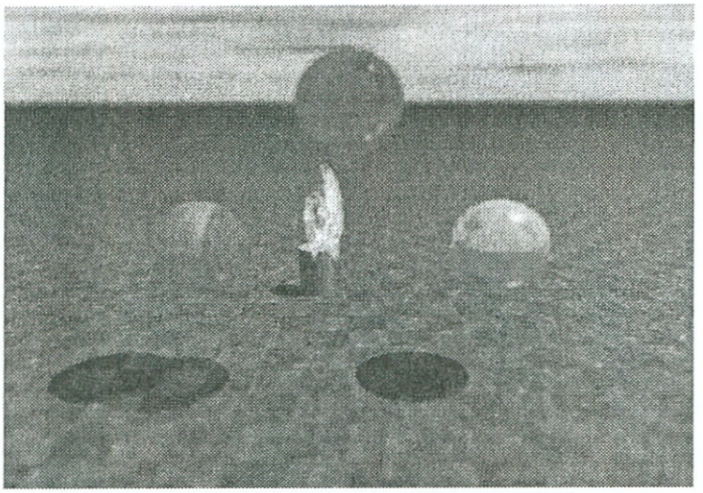
213. sz. ábra
Áttetsző gömb modellezése fénytöréssel



214. sz. ábra
Köd modellezése

Füstöt általában speciális textúrákkal modellezhetünk. A füstnek megfelelően mozgatjuk az ábrázoló textúraelemeket, melyek részben átlátszóak, hatásuk elmosódott.

Hasonló módszerrel tudunk előállítani például lángot is (lásd 215. sz. ábra).



215. sz. ábra
Láng ábrázolása textúrával

6.3.4. A 2D-s raszteres kép előállításának lépései

A modelltér jeleneteiből a raszteres kép előállítása több lépésben történhet. Amennyiben a modelltér elemeit már definiáltuk, akkor a képelőállításhoz még:

- *meg kell adni a modelltér objektumait megvilágító fényforrásokat a világkoordináta-rendszerben;*
- *rögzítenünk kell a nézőpontot vagy a kameraállást a világkoordinátarendszerben*, ahonnan a jelenetet szemléljük;
- el kell döntenünk, hogy a modelltér milyen objektumait kívánjuk szerepeltetni a generálandó képen. Ehhez *egy ablakot (windows) kell definiálni a világkoordináta-rendszerben, amelyen keresztül a nézőpontból a jelenetet látjuk.* Azok az objektumok, melyek ezen az „ablakon kívül" esnek, nem vesznek részt a képgenerálásban, azokat kivágjuk. A megjelenítésnek ezt a lépését Windowing and Clipping-nek nevezik a szakirodalomban;
- *a jelenetben szereplő objektumait a modelltérnek a világkoordináta-rendszerből affin és perspektív transzformációval egy normalizált ábrázolási térbe kell leképeznünk;*
- *az ábrázolási térben meg kell határozni az objektumok takarási viszonyait*, azaz a nézőpontból látható éleket és felületeket. *Ez a látható kép meghatározó algoritmusokkal történik;*
- *a látható felületelemek képpontjaihoz ezt követően hozzárendeljük a fényviszonyoknak és a textúráknak megfelelő színeket;*

- *a raszteres képernyőn kiválasztott ablaknak megfelelő pixelekre „vetítjük" a felületelemek képpontjainak színértékeit a monitor fizikai eszközkoordináta-rendszerében.*

!

Ezt az egész folyamatot a szakirodalomban a monitorképernyőre való kirajzolással együtt képgenerálási pipeline-nek is nevezik.

Az elkövetkezendőkben a fenti képgeneráló folyamat

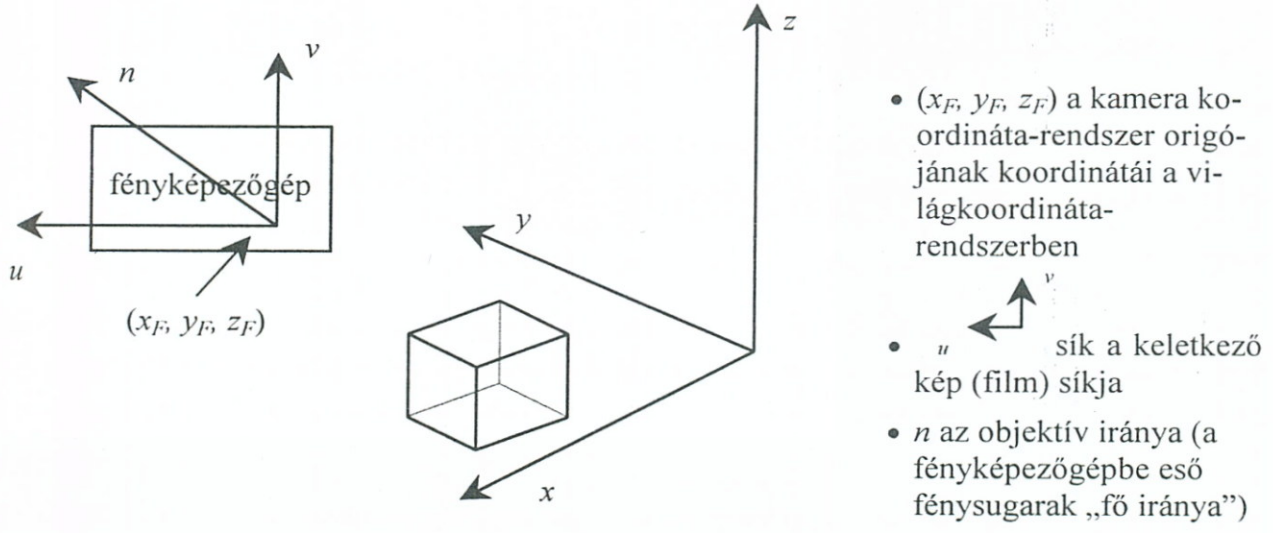
- kivágás és az ábrázolási térbe történő transzformálás,
- látható kép meghatározás és a
- megvilágítás és árnyékolás

legfontosabb modelljeit és algoritmusait fogjuk bemutatni.

6.3.5. Kivágás és a normalizált ábrázolási térbe történő leképezés (Windowing and Clipping)

A kivágás és az ábrázolási térbe történő leképezés lényegét legegyszerűbben egy, a modelltérben elhelyezett fényképezőgéppel szemléltethetjük. *A jelenet nézőpontjának definiálásához nyilván meg kell adni annak a pontnak a koordinátáit a világkoordináta-rendszerben, ahova a fényképezőgépet elhelyezzük. Ez azonban a pontos képalkotáshoz még kevés, szükség van a fényképezőgépben lévő film síkjának és az objektív térbeli irányának meghatározására is* (lásd 216. sz. ábra).

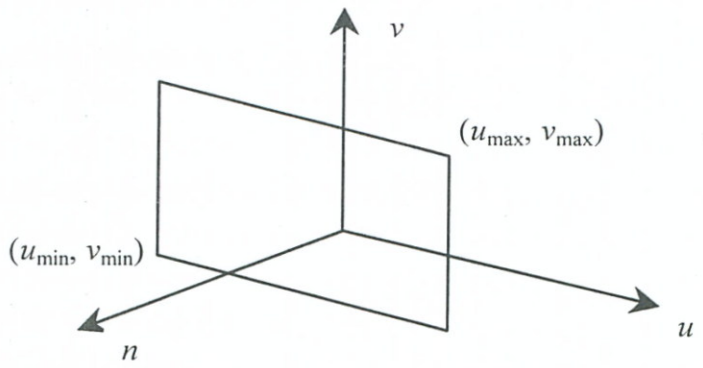
!



216. sz. ábra
Kamera definiálása a világkoordináta-rendszerben

Ennek megfelelően a fényképezőgépet a térben a „felvenni kívánt" tárgytól különböző távolságra, különböző beállításokban helyezhetjük el. (Természetesen a számítógépes rendszerünkben a fényképezőgép tulajdonképpen egy számítógépes program, a vele felvett tárgyak pedig vektorgrafikus adatállományok.)

A fényképezőgépben lévő filmen található egy filmkocka, és csak azt a lehatárolt térrészt tudjuk lefotózni, amit a filmkockának a méretei megengedtek. Ezzel analóg módon a képgeneráláshoz az (*u, v*) síkban fel kell vennünk egy ablakot, melyen keresztül a 3D-s modellteret látjuk. Mindazon objektumok, melyek képe az (*u,* v) síkra vetítve az ablakon kívülre esik, a jelenet képén nem fog szerepelni. Ezt a kivágó ablakot a nézőpont vagy kamera *u, v, n* koordináta-rendszerében mint egy (*u, v*) síkon felvett téglalapot határozzuk meg (lásd 217. sz. ábra).

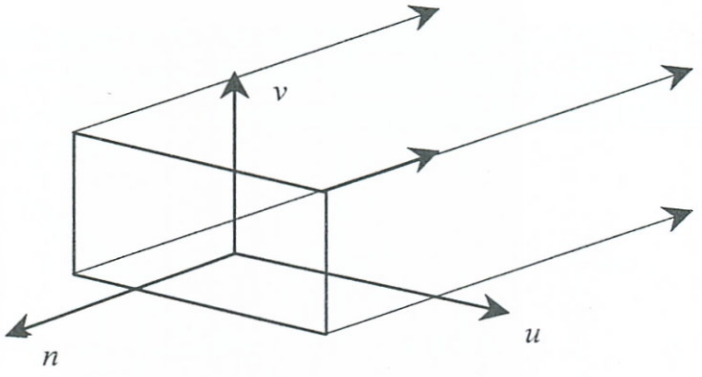


217. sz. ábra
Kivágó téglalap definiálása az (*u, v*) ábrázolósíkon

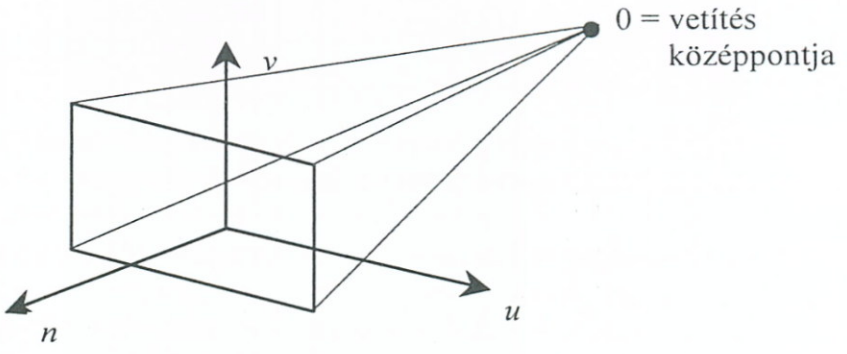
A jelenet képét tehát az (u, v) ábrázolósíkban definiált ablakra történő vetítéssel állítjuk elő. Ez lehet középpontos vetítés – ekkor perspektivikus térhatást érhetünk el a keletkező képen – vagy lehet párhuzamos vetítés.

Perspektíva képzésnél meg kell adnunk egy térbeli pontot az (*u, v, n*) koordináta-rendszerben, mely a centrális vetítés középpontja, a párhuzamos vetítést pedig egy irány, a vetítés iránya határozza meg. Ezek a kivágótéglalappal együtt egy látótestet (view volumen) határoznak meg, mely

- párhuzamos vetítésnél egy végtelen hasáb,
- középpontos vetítésnél pedig egy gúla (lásd 220. és 219. sz. ábrák).



218. sz. ábra
View Volumen párhuzamos vetítésnél



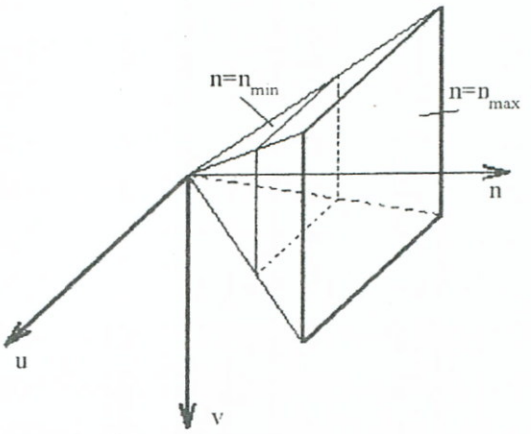
219. sz. ábra
View Volumen centrális vetítésnél

A látótestet gyakorlati megfontolások miatt (azaz azért, hogy a különböző algoritmusoknak a túl közeli és távoli koordinátákkal ne kelljen foglalkozniuk) egy első és hátsó síkkal korlátozni szokták, és így a különböző megjelenítési eljárásokban csak ezt a korlátozott térrészt vesszük figyelembe.

Miután meghatároztuk a megjelenítő algoritmusok által figyelembe veendő látótért, arra a kérdésre kell keresnünk a választ, hogy az *x, y, z* világkoordináta-rendszerben definiált objektumok koordinátái hogyan írhatók fel az *u, v, n* ábrázolási koordináta-rendszerben. (Azaz azt, hogy az *u, v,* n koordináta-rendszerben hol helyezkednek el a modelltér elemei.) Ehhez egy affin koordinátatranszformációt kell végrehajtanunk, mely az (*x, y, z*) koordináta-rendszert forgatással és eltolással átviszi az *u, v, n* koordináta-rendszerbe.

Ezzel azonban még mindig nem fejezhetjük be a kivágást és az ábrázolási térbe történő transzformálást, mivel látótesteink mérete és formája még eltérő. Ha ezen megfelelő egységesítéssel nem tudnánk változtatni, akkor a megjelenítési algoritmusainknak a legkülönbözőbb koordináta-rendszereket kellene figyelembe venniük. Ezért egy további affin koordináta transzformációval a felhasználásspecifikus látóteret egy normalizált látótérbe transzformáljuk.

Ez párhuzamos vetítésnél az $u=0$, $u=0$, $v=0$, $v=0$, $n=0$, $n=0$ síkok által meghatározott egységkocka középpontos vetítésnél pedig a $u=n$, $u=-n$, $v=n$, $v=-n$, $n=n_{min}$, $n=n_{max}$ síkok által határolt csonka gúla (lásd 220. sz. ábra).



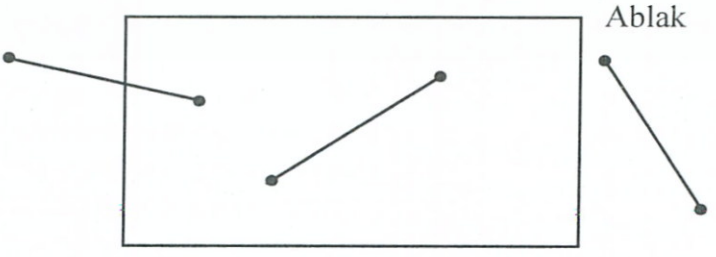
220. sz. ábra
Középpontos vetítés normalizált látótere

A megjelenítő algoritmusok egységes programozhatóságában most már csak az okoz problémát, hogy a normalizált látótér egyik esetben egységkocka, a másik esetben viszont csonka gúla. Ezen azzal tudunk segíteni, hogy a kivágást homogén koordinátákban hajtjuk végre, melyet w-clipping-nek neveznek. Mivel a homogén koordináták bevezetése lehetővé teszi a párhuzamos és a centrális vetítés egységes matematikai kezelését (a „végtelen távoli" pontból történő centrális vetítés egybeesik a párhuzamos vetítéssel) ezáltal a w-clipping-el az algoritmusokban egységesen kezelhető normalizált projektív látóteret kapunk. (Ezt a szakirodalomban NPC térnek szokták nevezni.)

! A figyelembe veendő *látótér meghatározása után, a kivágás utolsó lépéseként meg kell határoznunk, hogy a modelltér objektumai közül melyek esnek bele a látótérbe. Ehhez speciális kivágóalgoritmusokat használunk, melyek közül legismertebb az ún. Cohen-Sutherland w-clipping.*

Ezt 2D-ben egyenesszakaszokkal fogjuk szemléltetni. A következő esetek lehetségesek:

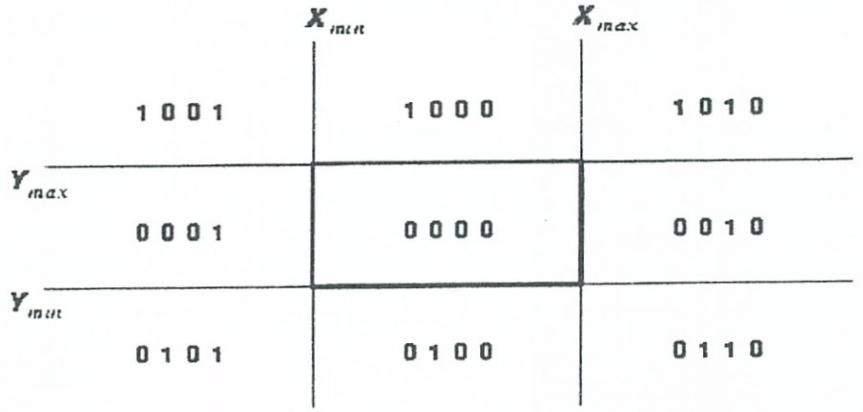
- egy szakasz teljes egészében a látótérbe esik,
- egy szakasz teljes egészében a látóteren kívül helyezkedik el,
- egy szakasz metszi a látótér határát.



221. sz. ábra
Egyenesszakaszok elhelyezkedési lehetőségei az ablakban

A kivágáshoz egy olyan algoritmusra lenne szükség, mely viszonylag egyszerűen lehetővé teszi a szakaszoknak ebbe a három osztályba való besorolását.

A Cohen-Sutherland-Clipping esetében a síkot az ablakkal együtt 9 részre osztjuk fel a 222. sz. ábra szerint.



222. sz. ábra
A Cohen-Sutherland-algoritmus szerinti felosztása a síknak és az outkódok

Minden egyes síktartományhoz egy 4 bites ún. outkódot rendelünk a következők szerint:

$x < x_{min}$ Bit 1 = 1
$x > x_{max}$ Bit 2 = 1
$y < y_{min}$ Bit 3 = 1
$y > y_{max}$ Bit 4 = 1

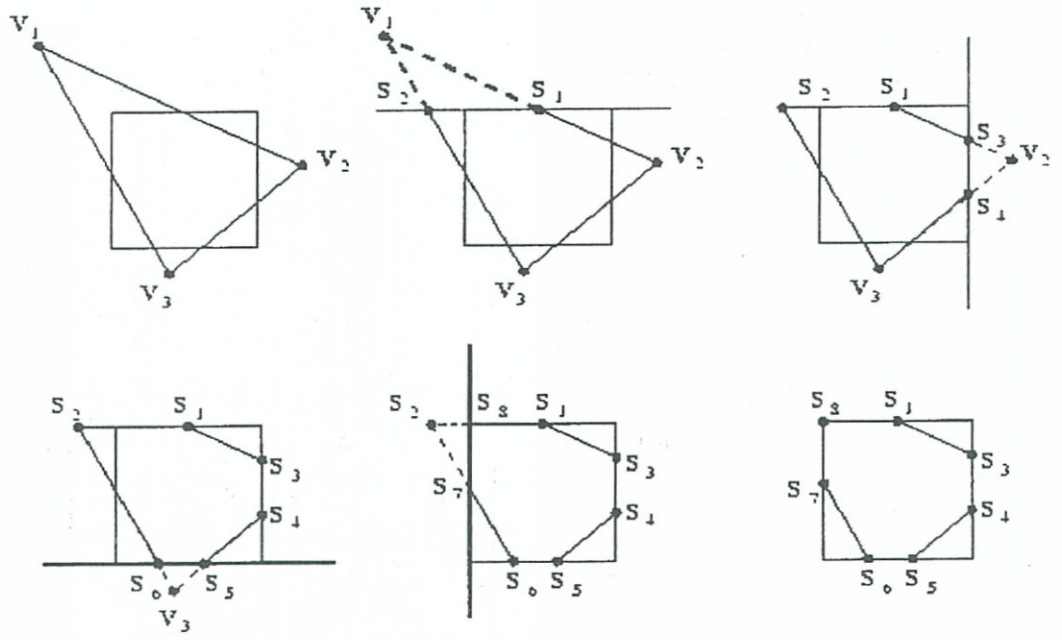
Az outkódok szerint a szakaszokat a következő módon osztályozhatjuk:

- a szakasz az ablakon belül helyezkedik el, ha végpontjainak outkódja = 0,
- a szakasz teljesen az ablakon kívül helyezkedik el, ha végpontjai outkódjának logikai „ÉS" művelettel képzett eredménye nem 0,

Ha a szakasz végpontjai outkódjainak a logikai „ÉS" művelettel kapott eredménye 0, akkor a szakasz metszi az ablakot. Ekkor kiszámítjuk a szakasz és az ablakhatárok metszéspontjait, és „eliminálju k" a szakasz ablakon kívüli részeit.

A megjelenítésnél a legfontosabb a háromszögeknek a látótérbe történő vágása. Ezt a Sutherland–Hodgman-algoritmussal valósíthatjuk meg. Az eljárás alapgondolatát a 223. sz. ábra szemlélteti.

Ennek a két algoritmusnak a 3D-s általánosításával lehet megvalósítani a w-clippinggel képzett normalizált látótér szerinti vágását a modelltér 3D-s objektumainak.



223. sz. ábra
A Sutherland–Hodgman kivágás elve

6.3.6. Látható kép meghatározó algoritmusok (Hidden-line, Hidden-Surface-Algorithmen)

A modelltér objektumaihoz tartozó vonalak és felületek láthatóságának meghatározása a grafikus adatfeldolgozás létrejötte óta az egyik központi probléma. Ennek lényege, hogy adott 3 dimenziós objektumok és nézőpont esetén el kell döntenünk mi látható az alakzatokból a vetítés központjából vagy irányából nézve. Ennek eldöntését segítő algoritmusokat a szakirodalomban takart vonal, illetve takart felület (hidden-line, hidden-surface) meghatározó eljárásoknak nevezik.

E területen jelentős előrelépést jelentett a hardver gyors fejlődése, ami lehetővé tette az egyszerűbb látható képmeghatározó algoritmusok áramkörökben történő realizálását. (Így például az úgynevezett z-buffer algoritmust a ma kapható legtöbb kártya már ismeri.) A legkorszerűbb grafikus hardver ezen túlmenően már a különböző képjavítási eljárásokat (például az aliasing probléma) és a speciális effekteket (például átlátszóság) is korrekten képes kezelni.

Az elkövetkezendőkben a *látható képmeghatározó algoritmusok* közül néhány tipikus és elterjedt eljárást mutatunk be. Ezek alapvetően *két kategóriába sorolhatók:*

- *A modelltér eljárások vagy tárgypontosság algoritmusok az objektumokat az adott nézőpontra vonatkoztatva közvetlenül hasonlítják össze.* Ezek pontosságának csak a gépi számábrázolás szab határt.
- *A képpontosság eljárások a megjelenítendő kép pixeljei szerint határozzák meg a jelenetek látható részeit.* Ezek pontossága nyilvánvalóan függ a megjelenítés eszközének felbontásától is.

A képpontosság algoritmusok azt határozzák meg, hogy a kép adott pixelében a jelenet *n* objektuma közül melyik látszik. Így egyetlen pixel színértékének meghatározásához az összes objektumot meg kell vizsgálnunk, és eldönteni, hogy a pixelen áthaladó vetítősugáron melyik objektum döféspontja van legközelebb a nézőponthoz. Ha a képet alkotó pixelek száma *p*, akkor az elvégzendő számítások mennyisége arányos $n \cdot p$-vel (itt *p* értéke milliós nagyságrendű). A modelltér tárgypontosság algoritmusai az objektumokat közvetlenül egymással hasonlítják össze, kiszűrve a takart objektumokat, objektumrészeket. Itt az objektumok összehasonlítására fordított számításigény $n \cdot n$-nel arányos. Azt gondolhatnánk, hogy a második csoport $n<p$ esetén ésszerű választás, de a megoldandó részlépések ebben az esetben általában sokkal bonyolultabbak és időigényesebbek.

A tárgypontosság algoritmusokat kezdetben a nagy- és középgépes vektorgrafikus rendszerek számára fejlesztették ki, míg a képpontosság algoritmusokat eleinte csak rasztergrafikus megjelenítőkön alkalmazták, kihasználva a megjelenítő kis pixelszámát. Későbbi algoritmusok aztán ötvözték a képpontosság és tárgypontosság számításokat, így a pontosságot és a sebességet is növelhették.

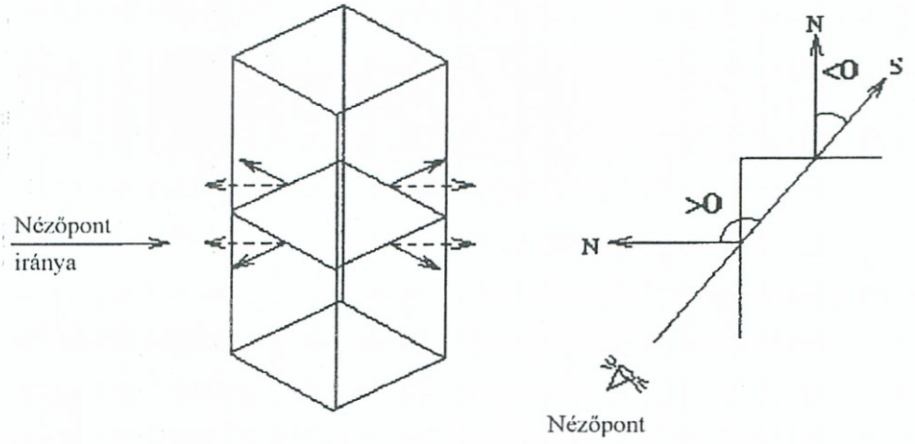
A modelltér elemeinek láthatóságát meghatározó algoritmusoknak nyilvánvalóan a 3D-s látótérben kell működniük, mivel a megjelenítő eszköz 2D-s koordináta-rendszerére való vetítés megsemmisíti a takarási viszonyok vizsgálatához szükséges mélységi információkat.

Az egy-egy pontban való láthatóság meghatározásához a w-clippinggel létrehozott homogén koordinátás normalizált látótérben arra a kérdésre kell megtalálnunk a választ, hogy egy pont eltakar-e egy másik pontot. Nyilvánvalóan a pontok csak akkor takarhatják egymást, ha azonos vetítősugáron vannak rajta.

Több olyan eljárást is kifejlesztettek, mely tetszőleges modelltérbeli jelenetek láthatóságának meghatározására ugyan teljes körűen nem alkalmas, de alkalmazásuk a számításigény egy részének kiküszöbölésével jól előkészíti a bonyolultabb algoritmusokat. A legismertebbek ezek közül az ún. hátsó felületek eltávolítása és a min/max teszt.

6.3.6.1. Hátsó felületek eltávolítása (Back face culling)

A zárt, sokszögekből (poligonokból) felépülő objektumokat teljesen körbezárják a felületüket alkotó poligonok. *Ha a poligonok normálvektorait úgy állítjuk be, hogy az objektumból kifelé mutassanak, akkor azok a poligonok, melyek normálvektorai nem a néző felé mutatnak, biztosan takarva lesznek az objektum közelebbi felületei által. Ezeket a takarásba kerülő, úgymond hátrafelé néző poligonokat back-face poligonoknak nevezzük, és az objektumokat leíró adatbázisból való eltávolításukat eredményező eljárást pedig back-face culling-nak.* Analóg módon az előre néző poligonokat front-face-nek nevezzük. A back-face poligonok könnyen azonosíthatók a normálvektoruk és a vetítés középpontjából a poligonhoz mutató vektor skalárszorzatának vizsgálatával: a pozitív érték hátrafelé néző poligonra utal (lásd 224. sz. ábra).



224. sz. ábra
Poliéderek hátsó lapjának eltávolítása

Ha az ábrázolandó jelenet egyetlen, konvex poliéderből áll, akkor a látható felületek meghatározása egyetlen back-face culling műveletre egyszerűsödik. Más esetekben további vizsgálatok szükségesek, hiszen a front-face poligonok is takarhatják egymást.

Egy poligonális objektumot metsző vetítősugár pontosan annyi back-face poligonon megy át, mint ahány front-face poligonon. *A back-face poligonok eltávolítása tehát éppen megfelezi a képpontosság algoritmusok által egy-egy pixel esetén megvizsgálandó poligonok számát.* Átlagos esetben a jeleneteket alkotó poligonok nagyjából fele hátrafelé néz, *így a back-face culling a tárgypontosság algoritmusok által vizsgálandó poligonok számát is nagyjából megfelezi.*

6.3.6.2. Min/max teszt

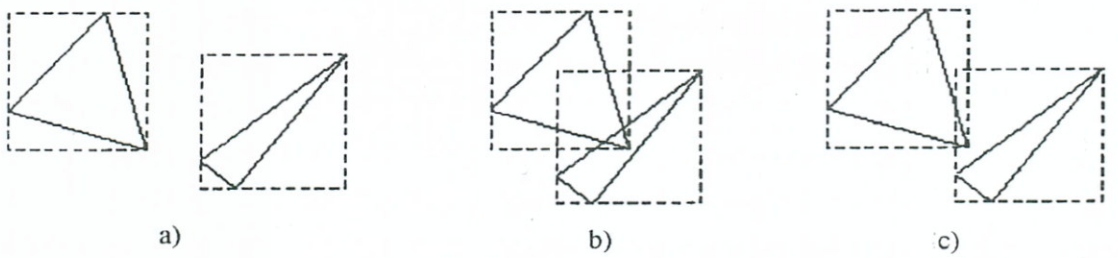
A térbeli objektumok takarási viszonyainak meghatározását esetenként segítheti, ha ezeket egyszerűbb formájú testekbe zárjuk, melyek térbeli elhelyezkedését hatékonyabb algoritmusokkal tudjuk tesztelni. Ezeket befoglaló testeknek (bounding volume) nevezik. A leggyakrabban alkalmazott befoglaló test a téglatest, de esetenként a céloknak jobban megfelelhet a henger vagy a gömb.

A látható felületek meghatározásának műveletigényét ez azért csökkentheti, mert azokkal a vetítősugarakkal, melyek a befoglaló testet nem metszik, az eredeti testünknek sem lehet közös pontja. (Így ezeknek a vetítősugaraknak a vizsgálatát a megjelenítő algoritmusból kiküszöbölhetjük.)

Hasonló az *alapgondolata a kivágóablak x, y síkjában az objektumok vetületeit tesztelő min/max eljárásnak. Ennél az objektumok vetületeit téglalapokkal vesszük körül, és ha ezeknek a téglalapoknak nincs közös része, akkor biztos lehetünk abban, hogy a vetületeknek megfelelő felületek sem fedik egymást.* Így

- ha az A poligonvetület legnagyobb x értéke kisebb, mint a B poligonvetület legkisebb x értéke, vagy
- ha az A poligonvetület legnagyobb y értéke kisebb, mint a B poligonvetület legkisebb y értéke.

akkor az A és B poligonok takarását nem kell vizsgálnunk (lásd 225. sz. ábra).



225. sz. ábra
Min/max teszt poligonvetületekkel
a) A két térbeli háromszög nem takarhatja egymást; b) A két térbeli háromszög takarási viszonyban van; c) A két térbeli háromszög nem takarja egymást, de ez a min/max teszttel nem dönthető el

Az elkövetkezendőkben a fontosabb képpontosság és tárgypontosság algoritmusokat fogjuk bemutatni.

6.3.6.3. A z-buffer vagy mélységtároló algoritmus

Ezt az algoritmust a Catmull már 1974-ben kidolgozta, de az informatika akkori fejlettségi szintjén a számítógépes megvalósítása még nagyon költséges lett volna.

A *z-buffer képpontosság algoritmus* előnye, hogy a szoftveres megvalósítás mellett a 3D-s gyorsítókártyákban is könnyen realizálható.

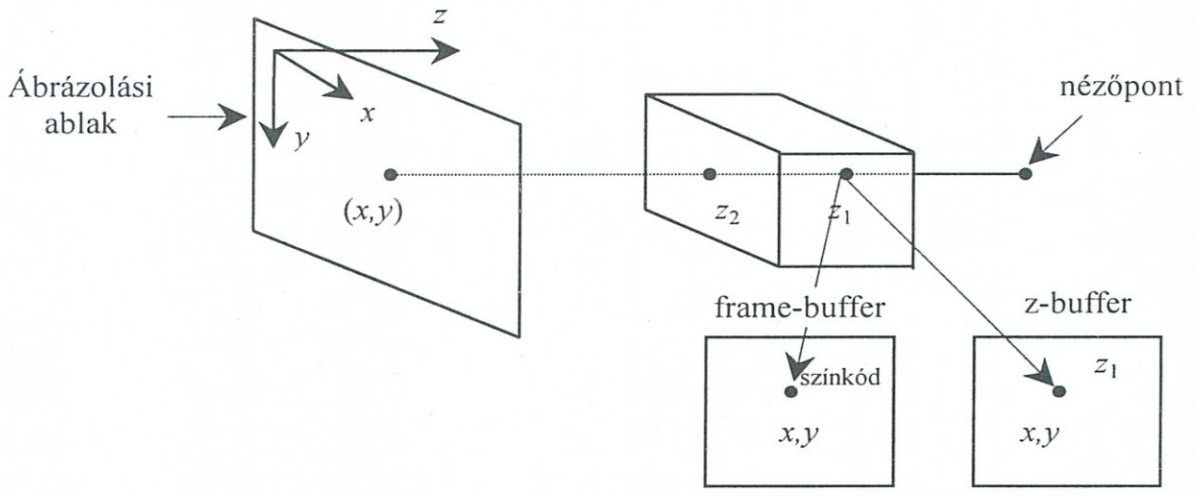
Az algoritmus két tárolóterületet használ:

- *a frame-buffert, mely a képernyő pixeljeihez rendelt színértékeket tárolja, induló feltöltése a képernyő háttérszíne,*
- *a z-buffert, mely az egyes pixelekhez rendelt z értékeket tárolja a normalizált látótérből, kezdeti értéke a hátsó kivágósík z koordinátája* (a maximálisan ábrázolható z-érték).

Ezt követően a *jelenet minden egyes normalizált látótérbeli objektumára a következő algoritmus kerül végrehajtásra*

- *egy (x, y) koordinátájú pixelhez tartozó vetítősugárhoz kiszámoljuk az összes metszett objektumhoz tartozó z értékeket,*
- *ha z értéke kisebb mint egy korábban a z-bufferben letárolt érték (azaz a pont közelebb van a nézőponthoz), akkor ezzel felülírjuk a korábban letárolt értéket a z-bufferben, és egyúttal a neki megfelelő színértékkel felülírjuk a frame-buffer (x, y) koordinátájú tárolóhelyét* (lásd 228 sz. ábra).

Az algoritmus lényege tehát az, hogy egy raszterponthoz tartozó vetítősugáron kiválasztjuk az objektumoknak a nézőponthoz legközelebb fekvő döféspontját.



226. sz. ábra
A z-buffer algoritmus elve

Az alapalgoritmus szempontjából közömbös, hogy az objektumokat, illetve a raszterpontokat milyen sorrendben teszteljük. Ha az összes (x, y) képpontra, illetve objektumra végrehajtjuk az eljárást, akkor ennek végén a frame-bufferben a kívánt képet kapjuk.

A z-buffer algoritmussal kapcsolatban már most két fontos megjegyzést kell tenni:

A raszterpontok száma a korszerűbb monitorok esetén milliós nagyságrendű. Ennek megfelelő számú z-érték tesztelést kell végezni az algoritmus végrehajtása során és ezzel arányos az eljárás memóriaigénye is.

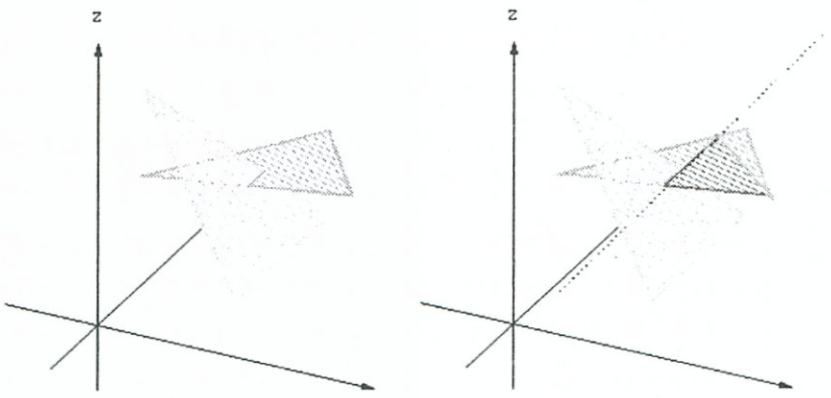
Vegyük észre, hogy *a z-buffer algoritmus a modelltér elemeinek formájától független, így háromszög, poliéder vagy görbült felületek láthatóságának megállapítására egyaránt alkalmas*. Alkalmazhatóságának egyetlen feltétele, hogy az objektumok felületi pontjaiban a nézőponttól való z távolság és az árnyalási információk (szín, megvilágítás, textúrák) meghatározhatók legyenek.

Az algoritmus jelentős erőforrásigénye, valamint tiszta formájában való alkalmazásánál egyes további eljárások (anti-aliasing, az átlátszóság stb.) elvi megvalósíthatatlansága (ugyanis a z-bufferben csak egy z értéket tárolunk) miatt a z-buffert általában más eljárásokkal kombinálva szokták alkalmazni.

6.3.6.4. Mélységi rendező algoritmusok (Depth-Sort)

Ennek a tárgypontosság algoritmusnak az az alapgondolata, hogy a megjelenítendő objektumokat a nézőponttól való távolság függvényében sorba kell állítani. Az algoritmus a helyes takarási viszonyokat úgy alakítja ki, hogy a nézőhöz közelebb eső objektum képe felülírja a távolabbi objektum képét. Ennek legegyszerűbb változata az úgynevezett festő algoritmus (painter algorithm), mely a képfestés technikájához hasonlóan a távolabbi objektumokra ráfesti a közelebbieket.

Ezek az algoritmusok általában az objektumok poligonfelületekkel, legtöbbször háromszögekkel való közelítését tételezik fel. Ha egy háromszög „z" irányban egyértelműen takarja a másikat, akkor a megjelenítésnél ennek képével felül kell írni a távolabbi háromszöget. Ha két háromszög mindegyike takarja a másikat, akkor ezeket úgy kell felbontani részháromszögekre, hogy a takarási viszonyok egyértelműek legyenek (lásd 227. sz. ábra).



227. sz. ábra
Két egymást kölcsönösen takaró háromszög felosztása
a mélység rendező algoritmusban

! *A mélység rendező algoritmusokat általában valamilyen képpontosság eljárással együtt alkalmazzák. Ekkor az objektumok sorba rendezése és szükségszerinti feldarabolását követően a pixeles megjelenítés például egy z-bufferrel kombinált scan-line algoritmussal történhet (lásd 6.3.6.3. és 6.3.6.6. pontok).*

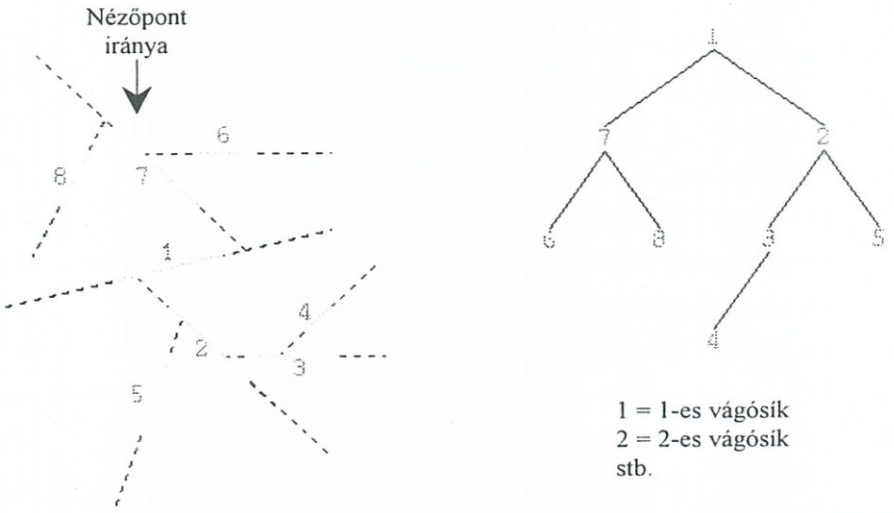
6.3.6.5. Bináris térfelosztó fák (BSP = Binary Space-Partitioning)

! *A háromdimenziós objektumok tetszőleges nézőponthoz tartozó megjelenítésére használt algoritmusok közül az egyik leghatékonyabb a bináris térfelosztó fákat (BSP-fák) alkalmazó eljárás.*

Bár az *algoritmusnak az objektumok térbeli helyzetét feltáró, előfeldolgozó része meglehetősen időigényes, később, új nézőpont definiálásakor gyorsan képes előállítani az új képet, hiszen ekkor már csak a frame bufferbe kell konvertálnia a takaráshelyes objektumokat.*

☝ A BSP fa algoritmusok azon alapulnak, hogy az ábrázolandó jeleneteket úgy is tekinthetjük, mint objektumcsoportok gyűjteményét. Ha található olyan sík, amely elválaszt egymástól két objektumcsoportot, akkor az a csoport, amely oldalán a nézőpont is van, eltakarhatja a másik csoportot, de a másik csoport által sohasem kerülhet fedésbe. Minden egyes objektumcsoport rekurzívan tovább osztható, ha megfelelő szeparáló síkok találhatók. A jelenet felosztását reprezentálhatjuk olyan bináris fával, melynek gyökere az elsőként választott szeparáló síknál van. A fa belső csomópontjai a szeparáló síkokat jelentik, a fa levelei pedig a felosztás során keletkezett térrészeket. Minden térrészhez hozzárendelhetjük az objektumcsoportok olyan sorrend-

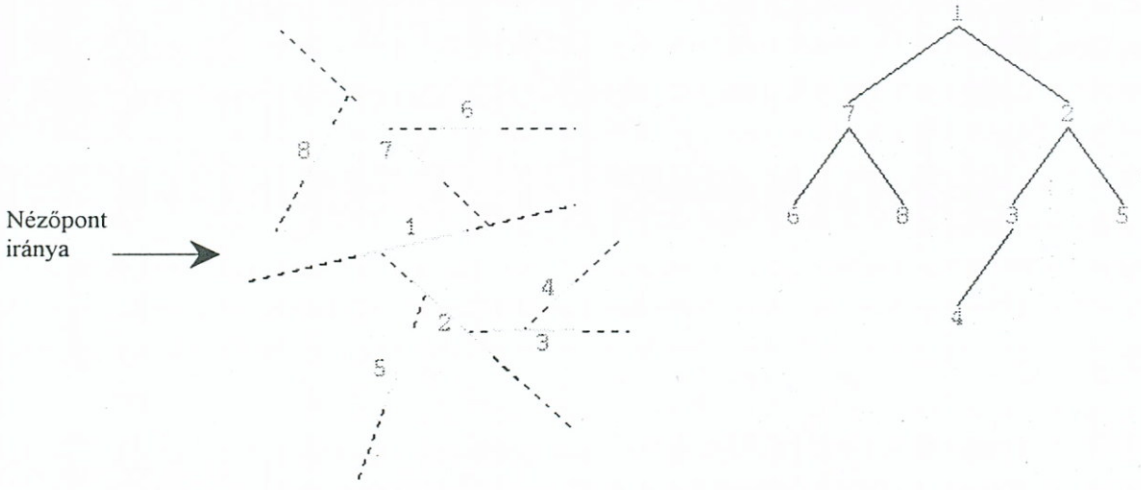
jét, amely az adott térrészben elhelyezkedő nézőponthoz tartozó helyes takarási viszonyokat rögzíti a megjelenítés során (lásd 228. sz. ábra).



228. sz. ábra
A BSP algoritmus

Bár az algoritmus helyes működése nem függ attól, hogy hogyan választjuk meg a szeparáló síkjainkat, a síkok által végrehajtandó vágások száma azonban jelentősen befolyásolja az algoritmus időigényét.

A BSP fa algoritmusok, a mélységi-rendező algoritmusokhoz hasonlóan az objektumok vágását és rendezését tárgypontosság módszerekkel végzik, képpontosság technikákhoz pedig csak a megjelenítés során nyúlnak. Az itt elvégzett objektum felosztásokat, ellentétben a mélységi-rendező algoritmusokkal, csak akkor kell újra elvégezni, ha a jelenet megváltozik (lásd 229. sz. ábra). Másként fogalmazva, a nézőpont megváltoztatása esetén nem kell a BSP fát módosítani.



229. sz. ábra
A nézőpont megváltoztatása nem módosítja a BSP fát

6.3.6.6. Scan-line algoritmusok

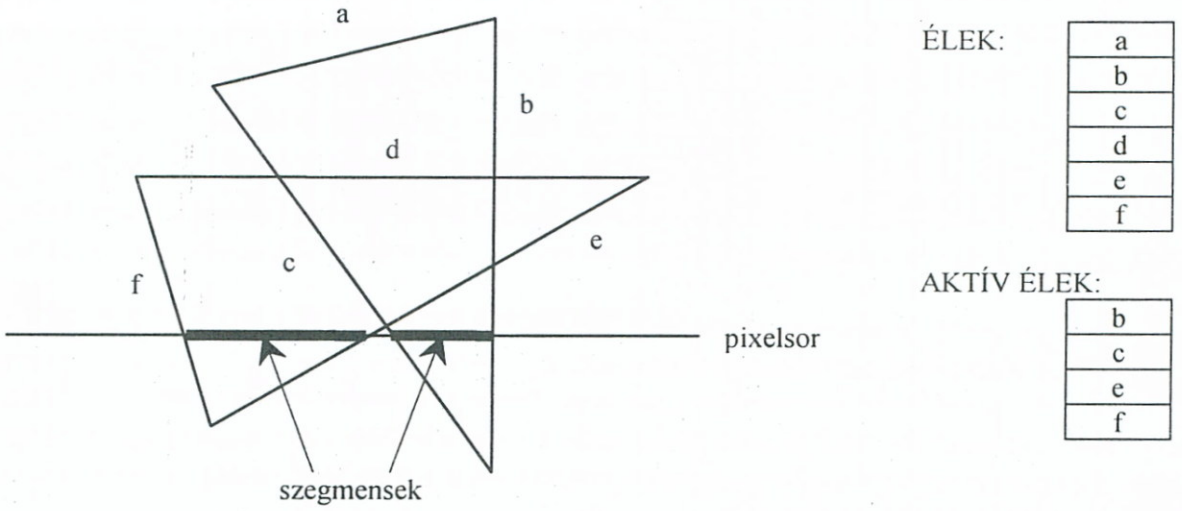
A scan-line algoritmusok képpontosság műveleteket használva, pixelsorokként készítik a képet, és a poligonok (háromszögek) frame bufferbe történő soronkénti konvertálásának módszerén alapulnak.

Az algoritmus, a jelenetet felépítő objektumokról gyűjtött információkat táblázatokban tárolja:

- a képsíkra vetített, nem vízszintes éleket az él táblázat,
- a poligonok fontosabb paramétereit a poligon táblázat,
- az éppen vizsgált pixelsorhoz tartozó éleket az aktív él táblázat

tartalmazza.

Egy-egy pixelsor vizsgálatakor az él táblázat azok az elemei, melyek metszik a sort, áttöltődnek az aktív él táblázatba (lásd 230. sz. ábra).



230. sz. ábra
A scan-line algoritmus él és aktív él táblázata

A pixelsor és a háromszögek közös részét az ábra szerint szegmenseknek nevezzük.

Ha a pixelsor egy szegmense csak egy háromszöghöz tartozik (mint például az ábrán), akkor ezt egyszerűen meg kell csak jeleníteni. Ha egy pixel több szegmenshez tartozik, akkor a megfelelő háromszögek mélységi vizsgálatával kell eldönteni, hogy melyik háromszög felületi pontját kell kirajzolni. (Ezt legcélszerűbben háromszögcsúcspont koordináták interpolációjával oldhatjuk meg.)

Ebből látható a scan-line algoritmusoknak az a nagy előnye, hogy az objektumok közötti geometriai összefüggéseket figyelembe véve a látható

pixelek meghatározásához szükséges tesztelésekből a feleslegeseket képes kiszűrni. Így például, ha két egymást követő pixelsorhoz tartozó aktív élek és ezek sorrendje megegyezik, akkor nem történt változás az objektumok takarási viszonyaiban, így nem szükséges a háromszögek újabb mélységi vizsgálata.

6.3.6.7. Területfelosztó algoritmusok

A *területfelosztó algoritmusok* alapgondolata, hogy néhány esetben (például a képen csak egy objektumot kell ábrázolni) nagyon egyszerűen megállapítható a képernyőn megjelenítendő kép, *ezért a bonyolultabb takarási vizsgálatokat a kép területének kisebb részekre való rekurzív felosztásával vezessük vissza az egyszerűen kezelhető esetekre.*

A képfelosztás történhet a raszteres képernyőkoordináta-rendszerben illetve a vektoros látótér koordináta-rendszerben.

6.3.6.7.1. Warnock algoritmusa

A területfelosztó algoritmusok közül legismertebb Warnock algoritmusa, mely a raszteres kép négy egyenlő részre történő felosztásán alapul. Egy adott, a felosztás során kapott raszteres képrészhez az algoritmus a következő eseteket vizsgálja:

- A képrésznek egyetlen poligonnal sincs közös része. Ekkor a háttérszínt kell megjeleníteni.
- A képrészben csak egyetlen poligonnak egy darabja található. Ekkor a háttérszín mellett ezt a metsző poligonrészletet kell megjeleníteni.
- Az adott képrészt egy poligon teljesen lefedi. Ez esetben ennek a poligonnak a színét kell megjeleníteni.
- Több poligon is metszi a képrészt, de ezek között egy egyértelmű sorrend állapítható meg a nézőponthoz való közelség szerint. Ekkor például „egymásra festéssel" rajzolhatjuk ki ezt a képrészletet.

Vannak természetesen olyan esetek is, amelyek nem sorolhatók be az előző négy kategóriába (például egymást kölcsönösen metsző háromszögek). Ekkor ezt a képrészt az algoritmus ismét felbontja 4 részre és ezek közül mindegyiket vizsgál az előző 4-es osztályozás szerint. A nem eldönthető képrészek felosztása egészen addig történik, míg ezek a pixel méretét el nem érik. Ha ez megtörtént akkor erre a pixelre külön meghatározásra kerül a nézőhöz legközelebb eső pont.

Az algoritmus könnyen párosítható anti-aliasing-ot eredményező eljárással, ha a felosztást subpixel méretekben tovább folytatjuk és így meghatározzuk az egy pixelt részben fedő poligonokat, és ezek színértékeit átlagoljuk.

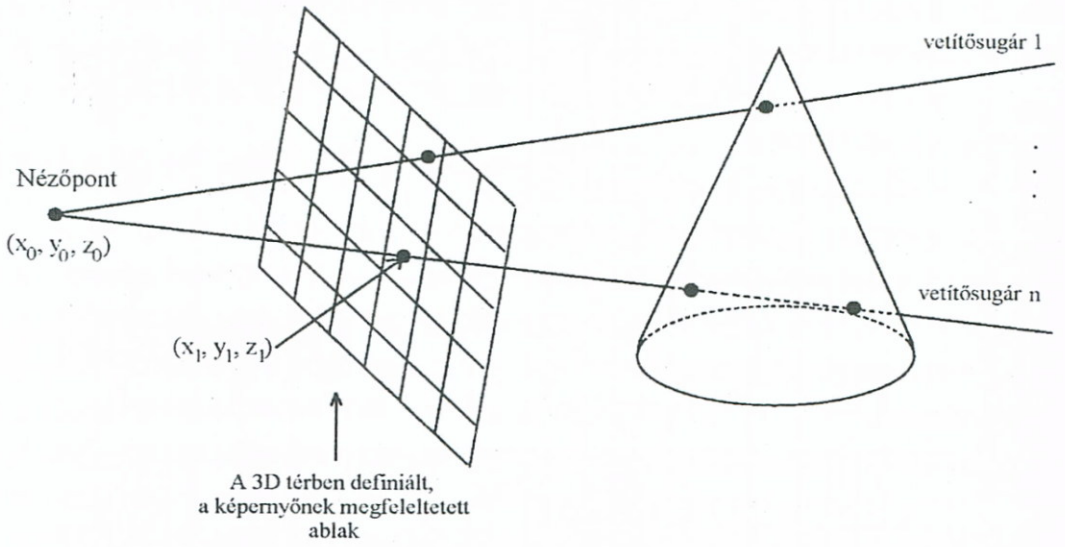
6.3.6.7.2. A Weiler-Atherton-algoritmus

Az eljárás első lépésben rendezi a jelenet poligonjait a nézőponttól való „z" távolságuk szerint, majd a legközelebbi poligonnal elmetszi az összes többit. Ezek után azokat, melyek takarása egyértelmű az algoritmus eliminálja. Azokban az esetekben, amikor kölcsönös takarás van, tovább felosztja a poligonokat.

6.3.6.8. Sugárkövetéses algoritmusok (raytracing)

! *A sugárkövetéses* (raytracing) *algoritmusok a felületek láthatóságát a nézőpontból kiinduló, képzeletbeli fénysugarak követésével határozzák meg.* Tipikus képpontosság algoritmusok. *Működésükhöz a nézőpontot és egy tetszőleges vetítési síkon felvett ablakot kell definiálni. Az ablak téglalap alakú területét felosztó szabályos rács a képernyő pixeljeinek felel meg.*

A raytracing algoritmussal a nézőpontból vetítősugarat indítunk az ablak minden egyes pixelén keresztül a jelenet objektumai felé. Az adott pixel színét a sugár által a nézőponthoz legközelebb metszett objektum színe határozza meg (lásd 231. sz. ábra).



231. sz. ábra
A raytracing elvi működése

Ebben a részben a raytracing algoritmust, mint egy, a látható felületek meghatározására szolgáló eljárást vizsgáljuk meg. Az árnyékok képzésére, a tükröződésekre és a fénytörésekre, azaz a raytracing-et híressé tevő jellemzőkre a 6.4.1. fejezetben térünk ki részletesen.

Minden raytracing algoritmussal szemben támasztott legfontosabb követelmény, hogy képes legyen fénysugarak és objektumok metszéspontjait megtalálni. A feladat megoldásához a fénysugarat a nézőpont és egy pixel által definiált vektor segítségével parametrikus formában írjuk fel. Ha a nézőpont az (x_0, y_0, z_0), a vizsgált képpont pedig a (x_1, y_1, z_1) koordinátahármas által meghatározott helyen található, akkor a rájuk illeszkedő sugár bármely (x, y, z) pontja megadható az egyenes paraméteres egyenletével (lásd II. fejezet):

$$x = x_0 + t(x_1 - x_0), \quad y = y_0 + t(y_1 - y_0), \quad z = z_0 + t(z_1 - z_0),$$

A t paraméter 0 és 1 közé eső értékei a nézőpont és a vizsgált pixel közé eső fénysugár-pontokat definiálják, negatív értékek esetén a nézőpont mögött vagyunk, míg 1-nél nagyobb t értékekre a képsík túlsó oldalára kerülünk. A fénysugár és az objektumok metszéspontjainak meghatározásához minden egyes, a jelenetet alkotó objektumtípust matematikailag le kell írnunk.

Poligonok esetében célszerű először azt meghatározni, hogy a sugár metszi-e a poligon síkját, és csak ezután azt, hogy ez a metszéspont (ha létezik) benne van-e a poligonban vagy sem. A síkot leíró egyenletet

$$Ax + By + Cz + D = 0$$

és az egyenes előbbi egyenleteit, mint egyenletrendszert megoldva dönthető el, hogy a sugár metszi-e a síkot.

A z-buffer algoritmushoz hasonlóan a raytracing-nek is megvan az az előnyös tulajdonsága, hogy metszéseket csak objektumok és fénysugarak között kell elvégeznünk, objektumok egymás közti helyzetének közvetlen meghatározására nincs szükség.

A raytracing algoritmus legegyszerűbb (egyben legdrágább) változata a pixelekhez tartozó sugarakkal elmetszi a jelenetben szereplő összes objektumot. Egy száz objektumból álló jelentről készítendő 1024x1024-es felbontású kép tehát több mint 100 millió metszési műveletet igényel! Nem meglepő tehát, hogy az átlagos jeleneteket ábrázoló képek előállítási idejének döntő része is a metszéspontok meghatározására fordítódhat. Ezért a raytracing algoritmusok időigényének csökkentését célzó kutatások fő iránya a metszések hatékonyságának növelése, ill. a felesleges metszési műveletek teljes elkerülése. Erre vonatkozó kérdésekre a 6.4.1.2. pontban térünk vissza.

6.3.7. Megvilágítási és árnyalási modellek és algoritmusok

! *A modelltér egy objektumának felületén látható színárnyalatokat a következő tényezők befolyásolják:*

- *a fényforrás típusa (pontszerű, kiterjedt stb.),*
- *a fényforrás sugárzásának erőssége és színe,*
- *a felület fényvisszaverő képessége,*
- *a felület és a fényforrás távolsága,*
- *a felület normálisa és a fénysugár által bezárt szög.*

Ennek megfelelően először a vektorgrafikus rendszerekben szokásosan definiálható fényforrások típusait fogjuk bemutatni.

6.3.7.1. Fényforrások modellezése

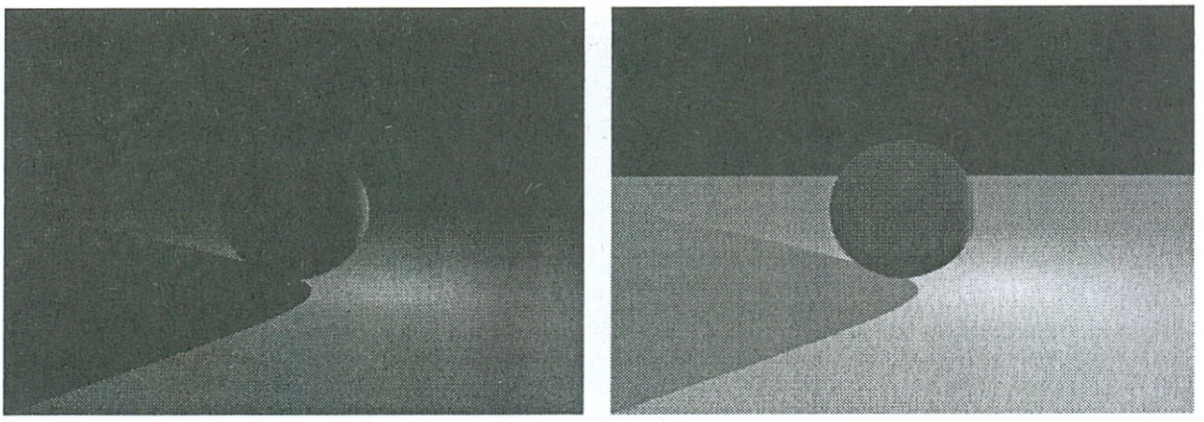
! *A vektorgrafikus rendszerekben általában a következő típusú fényforrásokat definiálhatjuk:*

- *szórt háttérvilágítás*, melynek nincs iránya,
- *távoli fényforrások*, melyek irányított párhuzamos fénysugarakat bocsájtanak ki,
- *pontszerű fényforrások*, melyek egy térbeli pontból minden irányban kibocsájtanak fénysugarakat,
- *reflektor fényforrások*, melyek a tér egy adott helyétől egy meghatározott irányban, párhuzamos vagy csonka kúp alakú fénynyalábot sugároznak,
- *kiterjedt fényforrások*, melyek meghatározott méretű és alakú sugárzó testek.

A *fényforrások sugározhatnak monokromatikus fehér színnel* (ez a leggyakoribb), *de lehet saját színük is, melyet a RGB színtérben adhatunk meg.*

6.3.7.1.1. Szórt háttérvilágítás (ambient light)

Ez a fogalom a fizikában nem értelmezhető. A számítógépes grafikában azért vezették be, hogy a felhasználó az ábrázolt jelenet összes objektumának a megvilágítását szabályozhassa. Hatása jó közelítéssel a nappali fényviszonyoknak felel meg egy erősen felhős égbolt esetén, amikor a testek egyenletesen, minden irányból kapnak fényt. *Ebben az esetben tehát a modelltér összes objektuma a tér minden irányából azonos erősségű fényforrással lesz megvilágítva.* A háttérvilágítást általában akkor állítjuk be megemelt értékre, ha a kép enélkül rosszul lenne látható (lásd 232. sz. ábra).



232. sz. ábra
Az ambient light hatása
a) háttérvilágítás nélkül; b) háttérvilágítással

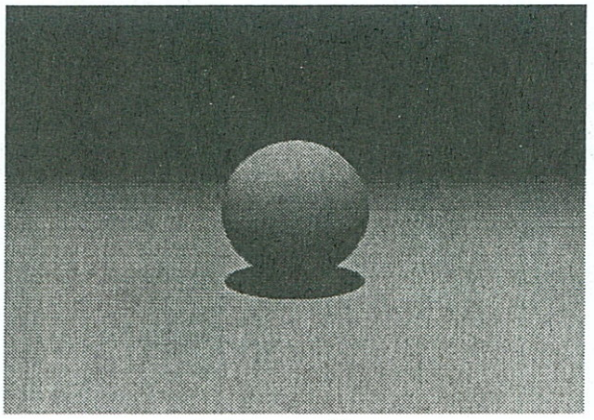
A háttérvilágítás önmagában árnyékot nem eredményez, de segítségével az árnyékos részek láthatóságát is befolyásolhatjuk.

6.3.7.1.2. Távoli fényforrások

Ezek párhuzamos fénysugarakat bocsátanak ki olymódon, hogy a fényforrástól távolodva sem csökken a megvilágítás intenzitása. Ezzel modellezhetjük például a napot, ami a nagy távolság miatt gyakorlatilag párhuzamos és változatlan erősségű fénysugarakat bocsát ki. Hatására árnyék képződik.

6.3.7.1.3. Pontszerű fényforrások (Point Light)

A fényforrásoknak ez a fajtája hatását tekintve az izzólámpákéhoz hasonló. *A pontszerű fényforrás a modelltér minden irányában azonos intenzitással sugároz. A közeli tárgyak erősebben meg lesznek világítva, mint a távolabbiak a Lambert-féle távolságtörvénynek megfelelően. A felületek megvilágításerősségének meghatározásakor ilyen fényforrásnál a Lambert-féle koszinusztörvényt is figyelembe veszik. Ez a fényforrástípus éles árnyékot eredményez (lásd 233. sz. ábra).*

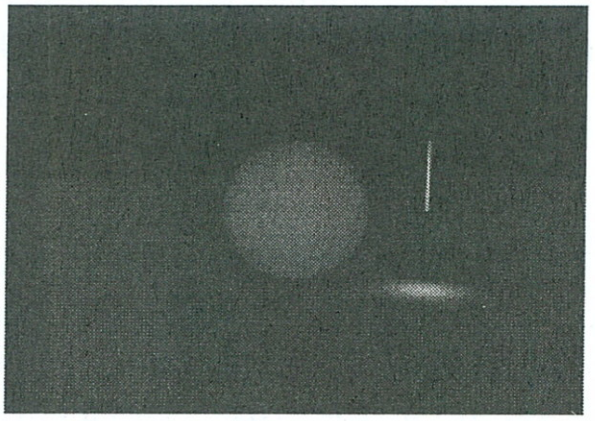


233. sz. ábra
Pontszerű fényforrás által megvilágított test

A fényforrás definiálásakor meg kell adnunk a térbeli koordinátáit és intenzitását.

6.3.7.1.4. Reflektorfény (Spotlight)

Ez a fényforrás a valódi reflektorokhoz hasonlóan egy irányban csonka kúp alakú sugárnyalábot bocsát ki. Megadásakor meghatározhatjuk a reflektor térbeli helyét, sugárzáserősségét, a sugárzás kúpszögét és az árnyéknál a megvilágított és árnyékban lévő térrész közötti átmenet gyorsaságát (lásd 234. sz. ábra).



234. sz. ábra
Reflektor fényforrás hatása

6.3.7.1.5. Kiterjedt fényforrások (area light)

Ezek olyan fényforrások, melyek sugárzását geometriai alakjuk is befolyásolja. Ezeket például sugárzó síklapok esetében pontszerű fényforrások együttesével közelíthetjük. Ezt úgy képzelhetjük el, hogy például egy tégla-

lap alakú sugárzó test *n* sorban és m oszlopban elhelyezkedő *n* x *m* db pontszerű fényforrásból áll, mint például egy labdarúgópályát megvilágító reflektorok összessége.

6.3.7.2. Megvilágítási modellek

A megvilágítási modellekkel írjuk le a modelltér objektumainak és a fényforrásoknak a kapcsolatát.

A 6.3.6. fejezetben tárgyalt algoritmusok a modelltér objektumainak direkt láthatóságát határozták meg. Ekkor egy felületi pont megjelenítésének csak az az egyetlen feltétele, hogy a nézőpont és a felületi pont közé ne essen semmilyen felületi elem, azaz a pont ne legyen takarásban. Ha egy felületi pont vagy egy közelítő háromszög ennek a feltételnek megfelelt, akkor a megfelelő pixelt a felület színére állítottuk be. A valóságban az objektumok és a fény kapcsolata ennél összetettebb, ennek vektorgrafikus kezelését segítik a megvilágítási modellek.

Az egyes modelleket a gyakorlati tapasztalatok, illetve optikai törvények alapján alkalmazzák, mindegyik egy-egy speciális fényeffektus figyelembevételét teszi lehetővé. Általában egy realisztikus fényviszonyokat tükröző kép megjelenítéséhez több modellt is felhasználunk, viszont minél többet veszünk figyelembe, a képelőállításnak annál nagyobb az erőforrásigénye (azaz annál lassúbb lesz a renderelés).

Az egyes megvilágítási modellekben a felületi pontok színét, világosságát külön-külön határozzuk meg. *Egy felületi pontnak megfelelő pixel végleges színét úgy kapjuk meg, hogy a test eredeti alapszínéhez az egyes megvilágítási modellek alapján számított módosító világosság és színértékeket hozzáadjuk.*

Az ismertebb megvilágítási modellek a következők:

- *általános háttérmegvilágítás (ambient light), amikor az objektumok saját színükkel egyenletesen sugároznak,*
- *diffúz fényvisszaverődés (diffuse reflection), mely matt felületekre jellemző,*
- *csillogó (fénylő) fényvisszaverődés (specular reflection), amely fénylő felületekre jellemző,*
- *tükröző fényvisszaverődés (total reflection), mely tükörként viselkedő felületekre jellemző,*
- *átlátszóság (transparency), mely olyan testekre jellemző, melyeken a fénysugár teljes egészében vagy részben áthalad,*
- *árnyék (shadows).*

A realisztikus képek előállításához az egyes megvilágítási modelleket a testek anyagi minőségétől függően különböző mértékben használhatjuk fel. Erre mutat néhány gyakorlati példát a 235. sz. ábra.

Anyagfajta	Ambient	Diffuse	Specular	Reflection	Transparency
Kréta	erős	erős	–	–	–
Csiszolt fém	–	–	erős	erős	–
Üveg	–	–	erős	átlagos	erős
Billiárdgolyó	átlagos	gyenge	erős	átlagos	–

235. sz. ábra
A felület minősége és a megvilágítási modellek összefüggése

A megvilágítási modelleket aszerint is osztályozhatjuk, hogy a megvilágítás szempontjából a modelltér elemei egymásra hatást gyakorolnak vagy sem.

A lokális megvilágítási modellekben az objektumok színét, világosságát más objektumok nem befolyásolják, ezek csak az objektumtól, a fényforrásoktól és a nézőponttól függnek. Ilyenek a háttérmegvilágítás, valamint a diffúz és csillogó fényvisszaverődés.

A globális megvilágítási modellekben az objektumok színe nemcsak a fényforrásoktól, hanem a más objektumokon áthaladó, illetve más objektumok által visszavert fénytől is függ.

A modelltér objektumainak kölcsönhatását figyelembe kell venni a tükröződő fényvisszaverésnél, az árnyékképzésnél és az átlátszóság kezelésénél.

Az elkövetkezendőkben a lokális megvilágítási modelleket tárgyaljuk, a globális modellekre a fotorealisztikus ábrázolás algoritmusainál fogunk visszatérni (lásd 6.4. fejezet).

6.3.7.2.1. Háttérfény vagy szórt fény (ambient light)

Ebben a megvilágítási modellben nincs fényforrás, az objektumok „saját" fényüket bocsátják ki. Ez megfelel annak, hogy a jelenetet egy irányfüggetlen, szórt fény világítja meg. A megvilágított felületi pont intenzitását leíró megvilágítási egyenlet:

$$I = k_a \cdot I_a$$

ahol I_a a megvilágított felületelem saját intenzitása, k_a a háttér megvilágítás-erőssége, mely a jelenet összes objektumára azonos.

A háttér k_a megvilágításerősségének értékét a felhasználó állíthatja be. Ha $k = 0$, akkor az árnyékban lévő objektumok feketék, k növekvő értékével pedig az objektumok árnyéktere egyre jobban meg lesz világítva. (A $k = 0$ értéknek megfelelnek a légkör nélküli esetben, például a világűrben észlelhető árnyékok.)

6.3.7.2.2. Diffúz visszaverődés (diffuse reflection)

A diffúz visszaverődés a matt felületek jellemzője, ezek a Lambert-féle megvilágítási törvényekkel modellezhetők. Ezért a megvilágítási egyenlet pontszerű, I_p intenzitású fényforrás esetén

$$I_p = f_{att} \cdot I_p \cdot k_d \cdot \cos(\underline{n}, \underline{l})$$

ahol

f_{att} = a fényforrás távolságtól függő gyengülését kifejező tényező. Ez a Lambert-féle távolságtörvény szerint $1/r^2$, ha r a pontszerű fényforrás és a felület távolsága. Az f_{att} tényezőt a szakirodalomban „light-source atenuation"-nak nevezik. Közeli fényforrás esetén (a hirtelen intenzitásváltozás miatt) a négyzetes gyengülés hatását általában korrigálják

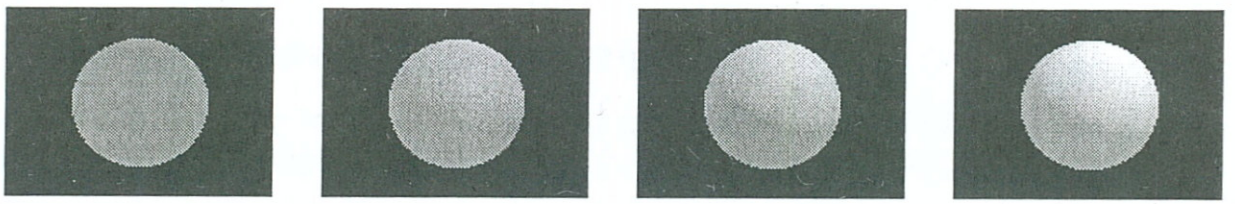
k_d = a felület anyagi minősége szerint beállítandó 0 és 1 közé eső diffúz visszaverődési együttható

$\cos(\underline{n}, \underline{l})$ = a felület felületi normálisa és a fénysugár iránya által bezárt szög

A gyakorlatban a diffúz visszaverődést és a szórt háttérfényt együttesen szokták alkalmazni, ekkor az ennek megfelelő megvilágítási egyenlet a

$$I = k_a \cdot I_a + f_{att} \cdot I_p \cdot k_d \cdot \cos(\underline{n}, \underline{l})$$

formát ölti. Erre az esetre mutat példát a 236. sz. ábra.



236. sz. ábra
Háttérfény (40%) növekvő diffúz visszaverődés együttes hatása
a) $k_d = 0\%$; b) $k_d = 20\%$; c) $k_d = 40\%$; d) $k_d = 60\%$

Az eddig bemutatott megvilágítási egyenleteink feltételezték, hogy a fényforrásnak monokromatikus fehér színe van. *A megvilágítási egyenletek színes fényforrásokra könnyen általánosíthatók, például RGB színtérben ekkor a fenti egyenleteket a három alapszínnek megfelelően kell felírnunk.*

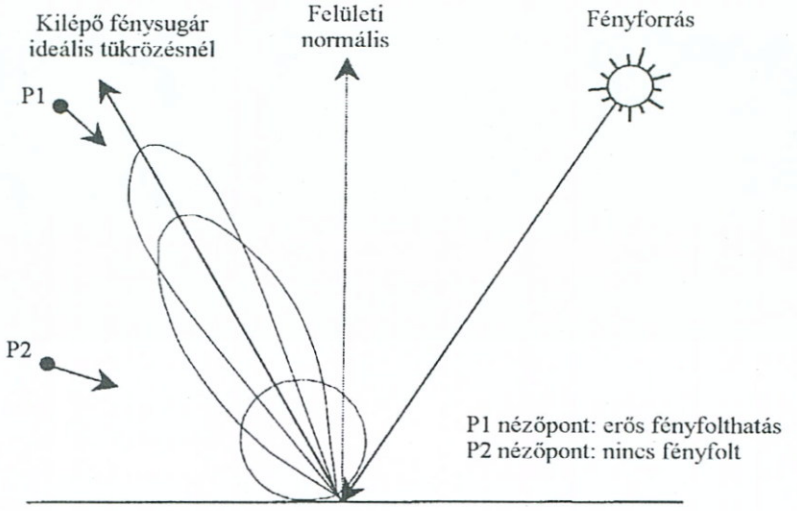
Ennek speciális esetei:

- fehér objektum és színes fényforrás esetén az objektum megvilágított színe megkapja a fényforrás színét,
- színes objektum és fehér fényforrás esetén az objektum eredeti színe szorzódik az intenzitásértékkel, a jellemző az eredeti szín lesz,
- színes objektum és színes fényforrás esetén általában valamilyen kevert színt kapunk.

6.3.7.2.3. Fényvisszaverődés fényes és csillogó felületekről (specular reflection)

A valóságban számtalan olyan tárggyal találkozhatunk, melyek felületére nem csak a már megismert diffúz fényvisszaverődés a jellemző, hanem a felületen fényes foltokat is látunk, melyek helye nézőpontunkkal együtt változik. Ez általában a sima felületekre jellemző, melyek bizonyos irányokban visszatükrözik felületükön a fényforrásokat. Ekkor tehát a matt felületekre jellemző diffúz és a tökéletesen (ideálisan) tükröző felületekre jellemző visszaverődés közti átmeneti esetet kell modelleznünk. Ezt a szakirodalom specular reflection-nak, illetve irányított diffúz fényvisszaverődésnek nevezi.

Megfigyelhetjük, hogy ha a felületet az ideális tükrözésnek megfelelően kilépő fénysugárral közel azonos irányából nézzük, akkor látunk fényfoltot, majd a látószög változásával ez gyorsan csökken (lásd 237. sz. ábra). Emellett a fényfolt nagysága és intenzitása erősen függ e felület simaságától.



237. sz. ábra
Irányított diffúz fényvisszaverődés és a nézőpont helyzete

E tapasztalati megfigyelések alapján javasolta *Phong a róla elnevezett megvilágítási modellt tökéletlenül tükröző testek számára. A Phong-féle fényfoltmodellezésnél* a megvilágítási egyenlet a következő formátumú:

$$I = f_{att} \cdot I_p \cdot k_s \cdot (\cos(\underline{v},\underline{r}))^n$$

ahol

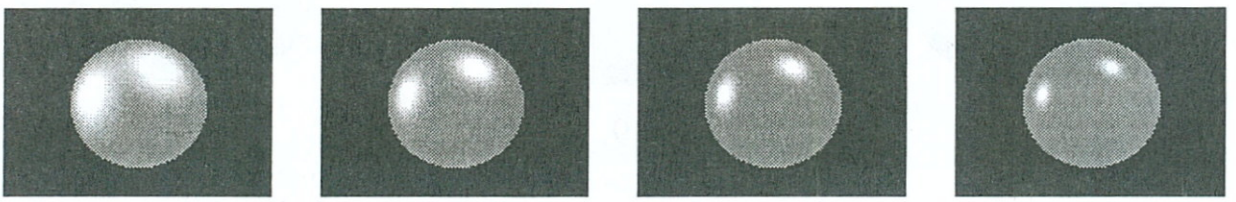
k_s = a felület fényvisszaverő képességét jellemző specular visszaverődési együttható

$\cos(\underline{v},\underline{r})$ = az ideális tükrözésnél kilépő fénysugár és a nézőpont iránya által bezárt szög koszinusza

n = az anyag simaságára jellemző specular visszaverődési kitevő

A $\cos(\underline{v},\underline{r})$ n-ik hatványa jól kifejezi azt a tapasztalati tényt, hogy a fényfoltintenzitás a kilépő fénysugár és a nézőpont bezárt szögének növekedésével rohamosan csökken.

A 238. sz. ábra mutatja, hogy „n" növekedésével a fényfolt mérete gyorsan csökken és n végtelen nagy értékénél kapnánk meg a fényforrás pontszerű képét.



238. sz. ábra
A fényfoltok méretének csökkenése a specular visszaverődési kitevő növekedésével

6.3.7.3. Lokális megvilágítási algoritmusok

Az előzőekben tárgyalt lokális megvilágítási modellek alapján a jelenetben található minden egyes objektum felületének pontjaira meghatározható a pontnak megfelelő pixel szín és intenzitásértéke. Nyilvánvaló viszont, ha a szín és intenzitásértéket minden egyes pontra meghatároznánk, akkor a renderelés reális időn belül számítógéppel nem lenne elvégezhető.

Ha a test határolófelülete poligonokból áll, akkor (feltéve, hogy fényfolthatással nem kell számolnunk) *elegendő a poligon egy pontjához tartozó szín és intenzitásértéket meghatározni és ezt követően a poligont ezzel a színnel feltölteni.* Így ebben az esetben a megvilágítás kiszámításának műveleti igénye nem a felületi pontok számával, hanem a testet borító poligonok darabszámával lesz arányos.

Erre a gondolatmenetre alapozva, *a nem poligon határolta testeket a megvilágítás kiszámításához sokszögekkel, leggyakrabban háromszögekkel szokták megközelíteni.*

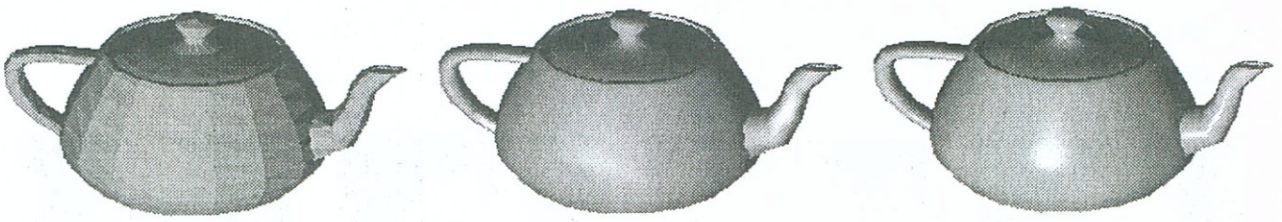
A sokszögekre bontott testekre dolgozták ki a lokális megvilágítási algoritmusokat, melyekkel – ha a sokszögekre való felbontás elég finom – görbült felületek képét is valósághűen meg tudjuk jeleníteni.

Ezeket az eljárásokat árnyalási algoritmusoknak (shading) is szokták nevezni a szakirodalomban.

Ezek közül a legfontosabbak:

- *a „Flat shading",*
- *a „Gouraud shading",*
- *a „Phong shading".*

A három algoritmussal kapott kép minőségének eltéréseit mutatja be a 239. sz. ábra.



239. sz. ábra
Az árnyalási algoritmusok képminősége
a) Flat shading; b) Gouraud shading; c) Phong shading

Látható az ábra alapján, hogy a képminőség fokozatosan javul, de ezzel együtt természetesen az algoritmusok számításigénye is növekszik.

Az elkövetkezendőkben röviden bemutatjuk ezeket az algoritmusokat.

6.3.7.3.1. Flat shading

Az objektumokat beborító poligonok árnyalásának az a legegyszerűbb módja, hogy minden egyes poligont azonos színnel jelenítünk meg, ez a constant vagy flat shading.

Az eljárás a szórt háttérvilágítás és a diffúz visszaverődés együttes megvilágítási modelljét használja, ezért egy felületi sokszög színének meghatározásához elegendő a poligon normálvektorának kiszámítása (lásd 6.3.7.2.2. pont megvilágítási egyenlete) *és a fényforrás jellemzőinek figyelembevétele.*

Ez alapján számítjuk ki poligononként azt az egy értéket, amely meghatározza a sokszög színét.

A flat shading-gel megjelenített jelenetek csak viszonylag ritkán eredményeznek realisztikus képet. Ezt csak a következő két esetben várhatjuk:

- a fényforrás „végtelen" távol van,
- a test valóban egy sokszöglapokkal határolt objektum és nem egy görbült felület sokszögközelítése.

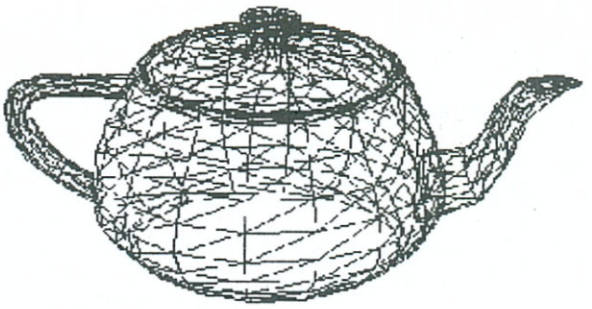
Ugyanakkor a megvilágítás kiszámítása ezzel az eljárással viszonylag gyorsan megtörténhet.

Mint ezt már a 239. sz. ábrán láttuk, görbült felületek sokszögközelítéssel és flat shading algoritmussal történő megjelenítése főként a sokszögek csatlakozó éleinek láthatóvá válása miatt jelent problémát. Ezt meg fokozza egy, még a múlt században Mach által felfedezett effektus, mely szerint az árnyalás intenzitáskülönbségeit az emberi agy felnagyítja.

A Mach-hatás kiküszöbölésére dolgozták ki az interpolált árnyalással működő algoritmusokat, melyek a közelítő poligonokat (háromszögeket) nem azonos színnel árnyalják, hanem a sokszög pontjainak intenzitásértékét a csúcspontok intenzitásértékeiből lineáris interpolációval származtatják, ezáltal az éles intenzitáskülönbségeket elsimítják.

6.3.7.3.2. Gouraud shading

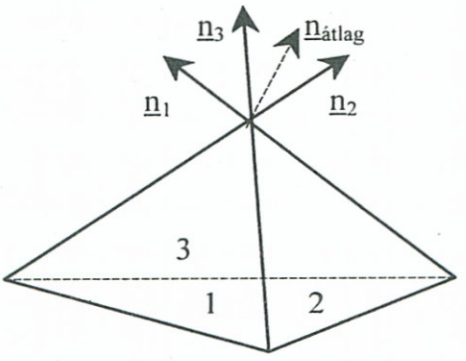
Az interpolált árnyalás (interpolated shading) technikáját először háromszög közelítésű testekre dolgozták ki (lásd 240. sz. ábra), melyet Gouraud általánosított tetszőleges poligonra.



240. sz. ábra
A teáskancsó háromszög közelítése

A Gouraud shading, vagy más néven intenzitás interpoláló árnyalás lényege háromszög közelítés esetén a következő:

- *Minden csúcspontban, ahol háromszögek találkoznak, kiszámítjuk a találkozó lapokhoz tartozó normálisok átlagát, így kapunk csúcspontonként egy ún. „pszeudo-normális"-t* (lásd 241. sz. ábra).



241. sz. ábra
A pszeudo-normáls képzése

- *A szórt háttérvilágítás és a diffúz visszaverődés megvilágítási modellje alapján kiszámítjuk a modelltérben a csúcsok intenzitásértékét a pszeudo-normálisból.*
- *Vetítjük a háromszögeket a képsíkra és itt a csúcsok intenzitás értékeit interpoláljuk az élekre és lapokra.*

A matt felületekre a Gouraud árnyalás elfogadhatóan realisztikus képet eredményez. Emellett az eljárásnak további előnye, hogy jól illeszkedik a scanline-algoritmus, z-bufferrel kiegészített változatához. Ez az oka többek között annak, hogy a korszerű grafikus kártyák chipjeibe a Gouraud shading algoritmus hardver megvalósítását ma már beépítik.

6.3.7.3.3. Phong shading

A normálvektor interpoláló árnyalás néven is ismert Phong-árnyalás a felületi normálvektorokat interpolja, nem pedig intenzitásértékeket. A normálvektorok interpolálása (hasonlóan a Gouraud árnyalás interpolálásához) a poligonok élein, majd a poligonok belsejében lévő pontokra történik meg. *Specular visszaverődési modell esetén jelentős különbség van a Gouraud- és a Phong-árnyalás között, az utóbbi sokkal élethűbben adja vissza a felületeken megjelenő fényfoltokat.*

Tegyük fel, hogy a $\cos^n\alpha$ megvilágítási tényező kitevője nagy, a poligon egyik csúcsában kicsi az α érték, míg az összes szomszédos csúcsban nagy (azaz az egyik csúcsra fényfolt esik, a többire pedig nem). Gouraud-algoritmusa az egyetlen csúcspontra jutó fényfoltot is szétosztja a poligon egész területén. Ezzel szemben Phong módszere képes előállítani a csúcspontra eső, éles fényfoltot. A poligon belsejébe eső fényfoltokat a Gouraud-árnyalás teljesen elveszti, hiszen csak a csúcspontok intenzitásértékei közé eső értékeket tud interpolációval előállítani. A Phong shading természetesen jobban képes kezelni ezeket az eseteket is.

Bár a *Gouraud- és a Phong-modell közötti különbség leglátványosabb megnyilvánulása a fényfoltok megjelenítésében tapasztalható, általában is kedvezőbb eredményeket kapunk ez utóbbi algoritmus használatakor*, mivel ez minden ábrázolandó ponthoz viszonylag jól közelítő normálvektort állít elő. Ennek eredményeként csökken a Mach-féle effektus, *viszont a normálvektorok kiszámítása miatt az algoritmus időigénye számottevően növekszik.*

Fontos kérdés, hogy a Phong shading algoritmusát a hardverben is realizálni tudjuk. Ezt az eljárás direkt implementációjával már megvalósították. Emellett elterjedt az a megoldás is, hogy a Gouraud-árnyalás alkalmazásánál azokban az esetekben, amikor a normálvektor gyorsan változik, a felület háromszögekre való felbontását finomítják. Ez lényegét tekintve a Phong-modell közelítését jelenti.

6.3.8. Felületi részletek modellezése, textúrák

Az eddigiek során tárgyalt árnyalási modellek a sík vagy görbült felületeket sima, egységes felületekként ábrázolták. A valós környezetünkben látható és érzékelhető felületek általában nem ilyenek. Ebben a fejezetben olyan módszereket és eljárásokat tekintünk át, melyek megpróbálják a felületi részletekkel valósághoz közelebb álló módon ábrázolni a modelltér objektumait.

6.3.8.1. Felületi-részlet poligonok

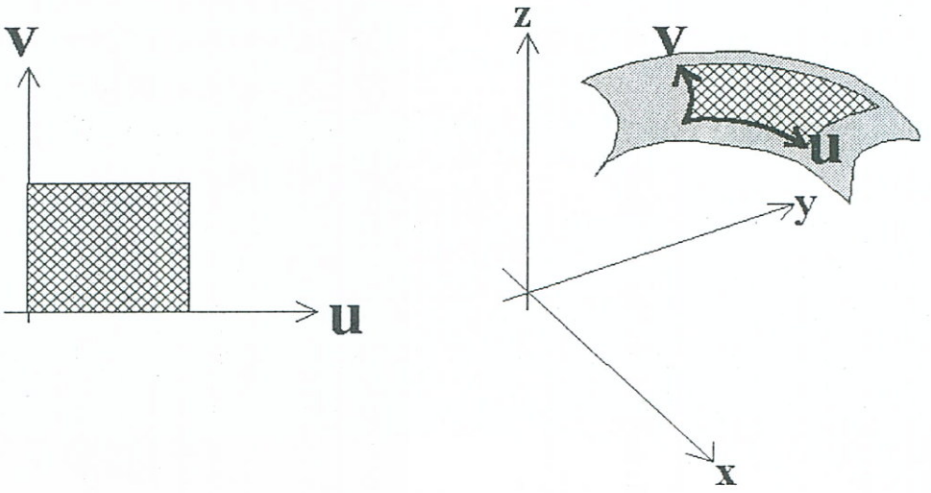
Az objektumainkra jellemző fontosabb részleteket (például egy épület falán levő ajtókat, ablakokat, feliratokat stb.) *legegyszerűbben újabb poligonok, ún. felületi-részlet poligonok felhasználásával tudjuk megjeleníteni.* Ezeket a poligonokat egyszerűen felvisszük a bázisfelület megfelelő oldalára, azzal párhuzamosan elhelyezve. *Így az árnyalás során ezek a poligonok egyszerűen helyettesítik a bázisfelület megfelelő részét az általuk fedett területeken.*

6.3.8.2. Textúra

Az ábrázolt objektumok felületi részletezettségének finomodása és bonyolultabbá válása, az explicit, poligonokból álló részletmodellezés nehezebbé válását, és hatékonyságának rohamos csökkenését eredményezte. Kétdimenziós képek felületekre történő leképzésével a Catmull által kidolgozott

! *texture mapping* (textúrázás) *néven ismert technika alternatív lehetőséget biztosít objektumaink megjelenítésének javítására. A művelet során felhasznált képet texture map-nek, vagy textúrának, egyes elemeit pedig texel-nek hívjuk.* A textúrát saját (u, v) koordináta rendszerében definiáljuk, a megjelenítendő pixelek intenzitásértékeit helyettesíthetik vagy módosíthatják a megfelelő texel-ek értékei. (Egy pixelre több texel is juthat.)

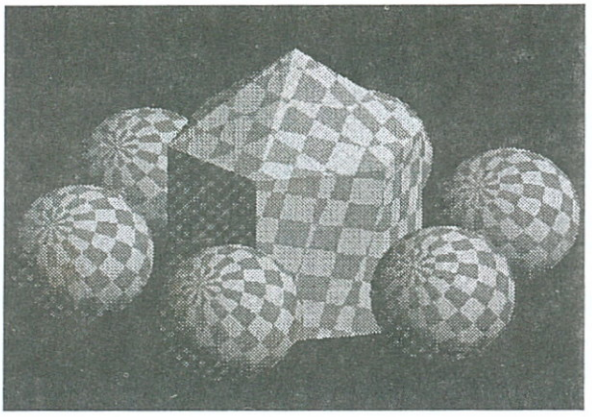
A textúrák felületekre „borítását" általában kétlépcsős transzformáció végzi. Ez a kiválasztott felületelemet áttranszformálja a textúra saját koordináta-rendszerébe, majd az így kapott texeleket visszavetíti a felületre (lásd 242. sz. ábra).



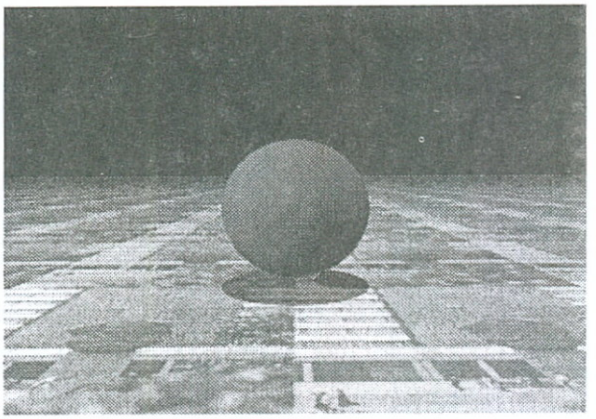
242. sz. ábra
Textúra vetítése felületre

A számítógépen a mintázatok adatait textúra-tárolóterületek tartalmazzák. A textúrák lehetnek:

!
- *egy egyszerű mintázatnak megfelelő bitmap-ok,*
- *ciklikusan ismétlődő mintázatok* (lásd 243. sz. ábra),
- *változó tartalmú mintázatok, mely összességében lehet egy 2D-s kép is* (lásd 244. sz. ábra),
- *olyan változó tartalmú képi információkat tartalmazó textúrák, melyeket speciális effektusok előállítására alkalmazunk* (például Enviroment mapping – lásd 6.3.8.4. pont).



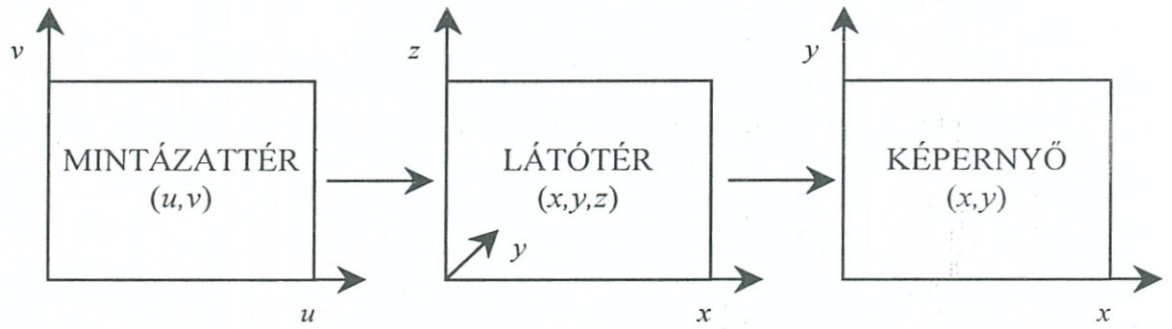
243. sz. ábra
Ciklikusan ismétlődő mintázat ráfeszítése objektumok felületére



244. sz. ábra
Raszteres kép ráfeszítése felületre

Textúrák alkalmazása esetén a felületnek megfelelő képernyőrész minden egyes pixelje esetén meg kell vizsgálni, hogy a hozzátartozó színértéket a mintamező (texture map) megfelelő része mennyiben módosítja. Ehhez két leképezéssel egyértelmű kapcsolatot kell létesítenünk a textúra (u,v) koordináta-rendszere és a képernyőkoordináta-rendszer között (lásd 245. sz. ábra).

!

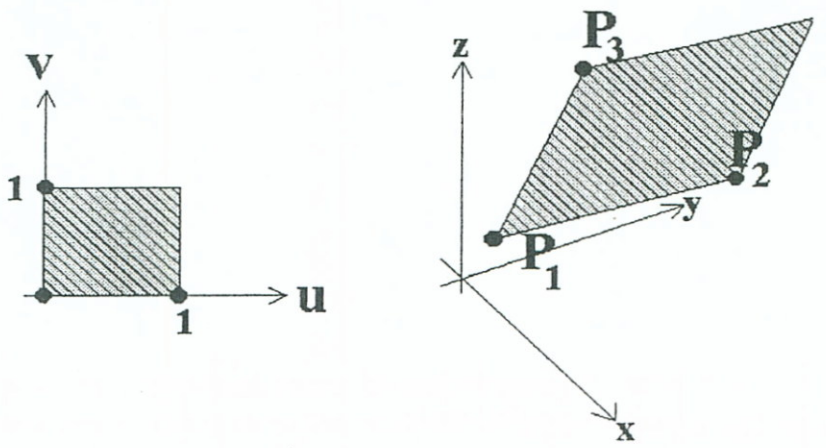


245. sz. ábra
A textúrák hozzárendelése az objektumokhoz és a képernyő pixeljeihez

E transzformációk inverze biztosítja a pixelek színértékének meghatározásához szükséges információt (azaz súlyozó intenzitásértéket) a mintázattérből.

Az (u,v) koordináta-rendszerrel jellemzett textúra map leképezését a 3D-s tér pontjaira görbült felületek esetén a következő, a leképezés lényegét érzékeltető példán mutatjuk be:

A 246. sz. ábra jelölései szerint legyen hozzárendelve az (u,v) koordináta-rendszer (0,1), (0,0), (1,0) pontjaihoz a felületen a P_1, P_2, P_3 pont. Megvizsgáljuk, hogy milyen módon lehet kapcsolatot létesíteni az (u,v) tér satírozott négyzetének és az (x,y,z) tér satírozott parallelogramjának pontjai között.



246. sz. ábra
A textúra map pontjainak hozzárendelése egy felület pontjaihoz

Mint azt a II. fejezetben megmutattuk a P_1, P_2, P_3 pontoknak megfelelő sík egy pontjához mutató $\underline{x}$ vektorra

$$\underline{x} = \underline{p}_1 + t_1 (\underline{p}_2 - \underline{p}_1) + t_2 (\underline{p}_3 - \underline{p}_1)$$

Válasszuk meg most t_1-nek a textúra map u koordinátáját, t_2-nek pedig a v koordinátáját $0 \leq u,v \leq 1$. Ekkor

$$\underline{x}(u,v) = \underline{p}_1 + u (\underline{p}_2 - \underline{p}_1) + v (\underline{p}_3 - \underline{p}_1)$$

megadja az összefüggést sík felület esetén a textúra map és a P_1, P_2, P_3 pontokkal meghatározott parallelograma pontjai között.

! *Algoritmikus textúrák esetén a felületre feszített mintázatot programmal állítjuk elő.* Erre jó példa egy fraktálok segítségével generált fűszálakból álló rét.

6.3.8.3. Szűrő és képjavító eljárások

A textúráknak a felületekre való feszítése során a különböző torzulások is felléphetnek. (Ezt tapasztalhatjuk a gyakorlatban is, például tapétázásnál.) Kialakulhat egy nem kívánatos alias effektus is a textúrák diszkrét mérete miatt. (Felhívjuk a figyelmet arra, hogy itt a textúra mérete a lényeges és nem a pixel mérete.)

Ezek kiküszöbölésére szolgálnak a különböző szűrő technikák (textúra mapping filtering) és képjavító eljárások. Ezek közül mutatjuk be a legfontosabbakat.

Anti-aliasing

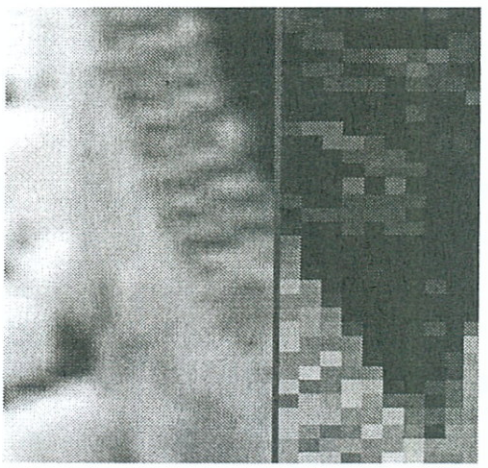
A textúra elemekből képzett ferde vonalak lépcsőzetes kialakulásának megakadályozását célozza az *Edge-Antialiasing. Ennek lényege, hogy a vonal széleinek és a vonal melletti textúrák színét átlagoljuk és ezt az átlagértéket jelenítjük meg a képen.* Ezzel a „simító" eljárással a kapott átmeneti színek elmossák a képen a „lépcsőket". Ezt a technikát esetenként a teljes képre is alkalmazzák (Full Scene Anti-aliasing), de ez a képet el is homályosíthatja.

Anisotrop Filtering

Speciális szűrő eljárás, mely azt a célt szolgálja, hogy például egy felirat ferde textúrákon is felismerhető maradjon. (Ezeket a bilineáris szűrés egyébként nagyon eltorzíthatja.)

Bilinear texture filtering

A textúra egy képpontjának színét, például a függőleges és vízszintes irányban mellette lévő 4 pixel színértékének átlagából számítjuk ki. Hatására a textúra elemek közötti éles átmenetek elmosódnak (lásd 247. sz. ábra).

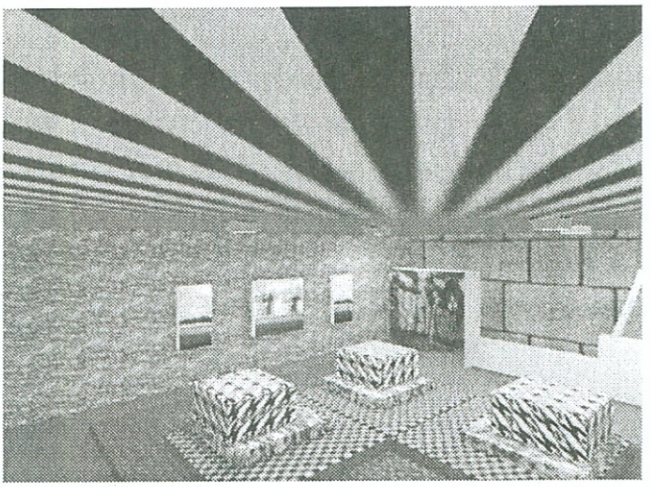


a) b)

247. sz. ábra
Kép korrigálása bilinear texture filtering-el
a) Korrekció után b) Korrekció előtt

Mip-mapping

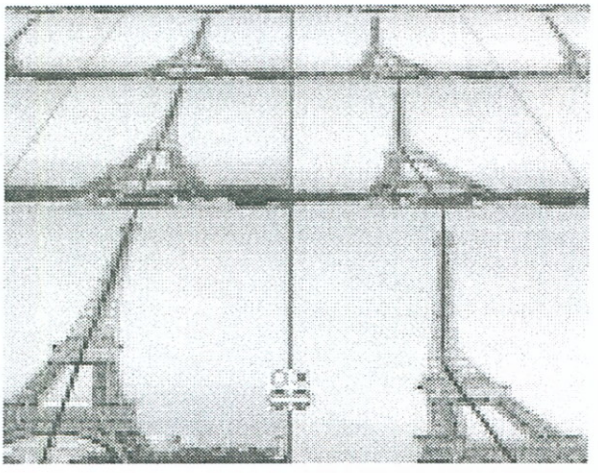
Ennél az eljárásnál egy textúrát több különböző felbontásban is letárolunk. Ha közeledünk a nézőponttal egy textúrázott objektumhoz, akkor a pixelesedés kizárása érdekében egyre finomabb felbontású változatát feszítjük fel a textúrának az objektum felszínére. Minél közelebb van egy objektum, annál részletesebb a textúra, a távolabbi tárgyak pedig elmosódottabbak.



248. sz. ábra
A mip-mapping hatása

Perspective correction

! *A térhatás elérése érdekében a textúrákat a távolság függvényében kisebbítve, ferdítve visszük fel a felületre* (lásd 249. sz. ábra).



249. sz. ábra
Textúra perspektíva korrekcióval

Trilineár textúre filtering

☝ Ez az eljárás tulajdonképpen a bilineáris szűrés és a Mip-mapping kombinációja. A távolság függvényében egy objektumhoz több különböző felbontású textúra tartozik, melyet a bilineáris szűrésnél megismert átlagolást követően végrehajtott interpolációhoz használunk fel. Ezáltal a tárgyakhoz való közeledésnél a textúra nem ugrásszerűen, hanem folyamatosan változik.

6.3.8.4. Speciális effektusok generálása textúrákkal

A textúrák jól felhasználhatók a fotorealisztikus megjelenítéshez szükséges speciális effektusok (átlátszó vízfelület, göröngyös felszínű tárgyak stb.) előállítására is. Az elkövetkezendőkben ezeket tekintjük át.

Alpha blending

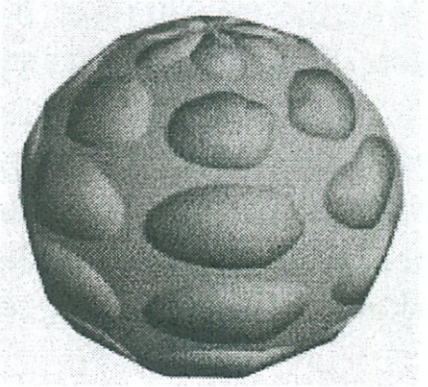
Ezt az eljárást átlátszó objektumok modellezésére alkalmazhatjuk. Ehhez egy úgynevezett Alpha-Buffer-ben tároljuk az átlátszó textúra mögötti kép textúráját. A képernyőn való megjelenítéskor az átlátszó textúra és az Alpha-Buffer-ben tárolt textúra színadatait átlagolják. Így például egy részben átlátszó vízfelületnél átmenetet képezhetünk a vízfelszín és a meder textúrája között.

Bump mapping

Ezzel az effektussal érhetjük el például, hogy egy kő érdes felszínű, vagy egy deszka szálkás legyen. Ezzel tudjuk modellezni például egy narancs felszínét egy narancsszínű gömbbel.

A Blium által kidolgozott eljárás szerint ehhez egy tárolóterületen letároljuk a megvilágítási modellben használt felületi normálvektorok irányításának „megzavarásához" szükséges adatokat (bump map). (Ezeket vagy konkrétan meghatározzuk, vagy egy véletlenszámgenerátorral állítjuk elő.) A képalkotáskor az eredeti textúrából kapott kép megvilágítási intenzitásait a módosított normálvektorokkal számítjuk ki.

Ennek az optikai trükknek a hatásaként a felületre merőlegesen a képen módosul a felszín: kiemelkedik vagy lesüllyed (lásd 250. sz. ábra).



250. sz. ábra
Bump mapping-al előállított göröngyös felszín

Environment Mapping, Reflection Mapping

! *Ezzel az eljárással textúrákkal modellezhetjük egy objektum környezetében lévő tárgyak visszatükröződését az objektum felszínén.* Ezt azzal lehet elérni, hogy *a visszatükrözött tárgy textúráját használjuk fel a tükröző felülettel rendelkező tárgy textúrájaként.*

Lightning Map

Ezzel a reális fényviszonyokat szimulálhatjuk textúrák alkalmazásával. Ehhez a környezeti megvilágításnak megfelelő intenzitásértékeket textúraként letároljuk és a megjelenítéskor a kép pixeljeit ezzel módosítjuk.

A 6.3.8.3. és 6.3.8.4. fejezetekben bemutatott eljárásokat a korszerű 3D-s kártyák többségében hardver úton is realizálták. A speciális effektusok kezelése külön műveleti igényt jelent az ehhez szükséges további textúrák feldolgozása miatt. Ezért, ha a grafikus kártya több textúra feldolgozó egységet tartalmaz, a képmegjelenítés sebessége jelentősen megnőhet.

6.3.9. Árnyékok kezelése

! *A láthatósági algoritmusok azt határozzák meg, hogy mely felületek láthatók a nézőpontból; az árnyékokat előállító algoritmusok pedig azt, hogy mely felületek „láthatók" a fényforrásból. Így a látható felület meghatározó és az árnyékképző algoritmusokat lényegében azonosnak tekinthetjük.* Ezért a következőkben bemutatásra kerülő algoritmusok a korábban már megismert, a látható felületek meghatározására szolgáló technikákat használják fel az árnyékok keresésére is.

A megvilágítási egyenletben az árnyékhatást úgy fejezhetjük ki, hogy a pontszerű fényforrásokra vonatkozó részt kiegészítjük egy S árnyékfaktorral, melynek értéke 1, ha a pont a fényforrásból „látható" és 0 egyébként.

Az elkövetkezendőkben néhány fontosabb árnyékképző algoritmust mutatunk be.

6.3.9.1. Árnyékképzés scan-line algoritmussal

Árnyékok képzésének legkorábbi eljárásai a scan-line algoritmusokat bővítették ki, összeolvasztva az árnyék és látható felület műveleteket. A fényforrást használva vetítési középpontként, ezek az algoritmusok a potenciális árnyékképző objektumok éleit rávetítették az aktuálisan vizsgált pixelsort metsző poligonokra. Amikor a konverziós folyamat elért egy ilyen árnyék élhez, a megjelenített pixelek színét megfelelően változtatták (ha a pixel egy árnyékos területre esett akkor sötétítették).

6.3.9.2. Árnyék testeket használó algoritmusok

Az egyes objektumok által árnyékolt térrészek, vagy „árnyék testek" meghatározásával is képezhetjük a jelenetek árnyékait. Az árnyék testek a térnek azt a részét töltik ki, melyeket egy-egy objektum eltakar a fényforrástól. Az árnyék testet egy objektum és egy fényforrás definiálja, határait pedig láthatatlan árnyék poligonok alkotják. (Az árnyék testeket a fényforrás felé néző felületelemekből generálhatjuk.) Azok a felületelemek, melyek benne vannak legalább egy árnyék testben, árnyékban lesznek. Több fényforrás esetében minden objektumhoz több árnyék testet kell készíteni.

6.3.9.3. A z-buffer algoritmus felhasználása az árnyékok meghatározására

A z-buffer algoritmussal az alaptechnika kétszeri alkalmazásával (a nézőpontra és a fényforrásra) is lehetőség van árnyékban levő felületek meghatározására. Az algoritmus a fényforrást használva vetítési középpontként, kitölt egy z-buffert, melyben fényforrástól mért távolságot rögzítik. Ezután a korábban megismert z-buffer algoritmus meghatározza a látható felületi pontokat, de az egyes pontok árnyalásakor a pontok térbeli koordinátáit átranszformálja a fényforrás középpontú rendszerbe, és a z koordinátákat összeveti az árnyék z-bufferben lévő távolságértékekkel. Ha a transzformált pont z koordinátája nagyobb, mint a bufferben tárolt érték, akkor a vizsgált pont távolabb van a fényforrástól, tehát árnyékban van.

6.3.10. Átlátszóság és tükröződés kezelése

6.3.10.1. Átlátszósági algoritmusok

Azok az objektumok, melyek teljesen vagy részben áteresztik a fényt átlátszóak, illetve áttetszőek. Az átlátszó tárgyakon tisztán átnézhetünk, az áttetsző tárgyak mögötti világ látványa viszont bizonyos mértékig elmosódott, például mint télen egy deres ablakon keresztül látható kép.

Az átlátszóság és áttetszőség modellezésére a következő módszerek alakultak ki:

- *törésmentes átlátszósági eljárások, melyek lehetnek interpoláló illetve szűrő algoritmusok,*
- *törő átlátszósági eljárások, melyek a fénytörés törvényei szerint kezelik a testeken áthaladó fénysugarakat.*

Az interpolációt használó algoritmusok az átlátszó felülethez és a mögötte lévő átlátszatlan felülethez tartozó sokszögekhez tartozó árnyalatokat lineárisan interpolálják.

Az átlátszóság mértékét egy k áttetszőségi együtthatóval veszik figyelembe, melynek értéke 0 és 1 között van. Ha k=0, akkor a felület nem átlátszó, ha pedig k=1, akkor a felület teljesen átlátszó így nem befolyásolja a felület mögötti tárgyak megjelenítését. A köztes esetekben k értékének megfelelően módosított intenzitásértékkel az áttetsző felület színértékét lineárisan interpolálják a mögöttes tárgy felületének intenzitásértékeivel.

Ehhez az algoritmus csoporthoz tartozik az átlátszóság textúrákkal való modellezését megvalósító alpha blending.

A szűrt átlátszóság algoritmusai az áttetsző felülethez tartozó sokszögeket szűrőként modellezik, melyek a különböző hullámhosszú fényt eltérő arányban engedik át.

Amennyiben nemcsak egy, hanem több átlátszó objektum is van a jelenetben, akkor ezeknek az átlátszó sokszögeknek a takarási viszonyait figyelembe véve az előző algoritmusokat rekurzív módon alkalmazhatjuk. Ez – különösen ha a modelltérben több fényforrás is található – rendkívül lelassíthatja a renderelés idejét.

Több látható felület meghatározó algoritmust is kiegészítettek oly módon, hogy az átlátszóságot is képesek legyenek kezelni. Ilyen változata létezik például az elterjedt scan-line és z-buffer algoritmusoknak is.

A z-buffer algoritmus kiegészítésekor sokszor az ún. képernyő maszk módszert alkalmazzák, mely egy átlátszósági bit maszk alapján indexelve esetenként az áttetsző felület néhány képpontját is kirajzolja.

A törő átlátszóságot és a teljes visszaverődést az eddig megismert algoritmusokkal csak nagy nehézségek árán lehetne kezelni. Erre a célra a fotorealisztikus képmegjelenítés algoritmusait (raytracing) használhatjuk.

6.3.10.2. Objektumok tükröződését kezelő algoritmusok

Objektumok közti tükröződésről beszélünk, ha egy felületen a jelenet egy másik elemének képe visszatükröződik. Ez a jelenség lehet nézőpontfüggő, tükörszerű visszaverődés, de lehet a nézőponttól független diffúz visszaverődés is. A raytracing és a radiosity módszerek tudják leghatásosabban modellezni a tükörszerű és a diffúz visszaverődéseket, de néhány korábbi technikával is lehet egészen elfogadható minőségű képet produkálni.

A tükörszerű visszaverődés modellezésére az ún. reflection mapping technikát dolgozták ki. Ennél megjelenítendő tükröző objektum köré olyan gömböt definiálnak, amelyre rávetítik az objektumot körülvevő modelltér képét. A gömb felületét ezután 2 dimenziós textúraként használják fel a tükröző felület képének kialakítása során. Alternatív megoldásként esetenként kockát használnak, és ennek oldalaira végzik el a vetítést. Természetesen a reflection map eljárás csak közelíti a valódi tükrözést, ennek ellenére sokszor jól használható effektusokat lehet előállítani vele.

ELLENŐRZŐ KÉRDÉSEK A 6.3. FEJEZETHEZ

320. Milyen esetekben van szükség a modelltér elemeinek képi megjelenítésére?
321. *Mi a huzalvázas képi megjelenítés lényege?*
322. *Mit értünk árnyalt megjelenítés alatt?*
323. *Mit jelent a renderelés?*
324. Mi a flat-shading?
325. *Mit jelent a fotorealisztikus renderelése?*
326. *Milyen követelményeknek kell megfelelnie egy 2D-s képnek, hogy térhatású legyen?*
327. *A felületeknek milyen tulajdonságait kell figyelembe venni realisztikus képek készítésekor?*
328. *Mivel tudjuk modellezni a különböző anyagú testeket?*
329. Mit jelent és mire használható a bump-mapping?
330. Miben különbözik a pontszerű és a kiterjedt fényforrások árnyéka?
331. *Mit kell figyelembe venni az átlátszóság modellezésénél?*

332. Mire használhatók a textúrák az átlátszóságmodellezésénél?
333. Mit jelent az alpha-blending?
334. *Hogyan modellezhető a köd és a füst?*
335. *Adja a modelltér jeleneteiből a raszteres kép előállításának lépéseit!*
336. *Mit értünk képgenerálási pipeline alatt?*
337. *Hogyan definiálhatunk egy kamerát?*
338. Mi a szerepe a kivágóablaknak?
339. *Hogyan állítjuk elő a jelenet képét az ábrázolósíkban?*
340. Mit értünk látótest (wiev volumen) alatt?
341. Miben különbözik a párhuzamos és a középpontos vetítés látóteste?
342. *Milyen célt szolgál a v-clipping algoritmus?*
343. Milyen célt szolgálnak a hidden-line, hidden-surface algoritmusok?
344. *Mi a tárgypontosság algoritmusok lényege?*
345. *Mi a képpontosság algoritmusok lényege?*
346. Hol működnek a láthatóság meghatározó algoritmusok?
347. *Mit nevezünk back-face cullingnak?*
348. Hogyan állapítható meg az objektumok „hátsó", nem látható lapjai poliéderek esetében?
349. *Mennyivel csökkenti a back-face culling a láthatósági algoritmusok műveletigényét?*
350. *Mire használhatók a befoglaló testek (bounding volume)?*
351. *Mi a min/max teszt alapgondolata?*
352. Mi az előnye a z-buffer algoritmusnak?
353. *Milyen puffereket használ a z-buffer algoritmus?*
354. *Ismertesse a z-buffer algoritmus működését!*
355. Mivel arányos a műveletigény a z-buffer alkalmazása esetén?
356. *Mennyiben függ a z-buffer algoritmus a modelltér objektumainak alakjától?*
357. Miért alkalmazzák együtt a z-buffer algoritmust más eljárásokkal?
358. *Mi a mélység rendező algoritmus lényege?*
359. Hogyan kezeljük a kölcsönösen takaró háromszögeket a mélység rendező algoritmusban?
360. *Milyen algoritmusokkal együtt alkalmazzuk a mélység rendező algoritmust?*
361. *Milyen célra használható a BSP fa algoritmus?*
362. Mi a BSP fa algoritmusok alapgondolata?
363. *Hogyan állítják össze a képet a scan-line algoritmusok!*
364. *Mi a scan-line algoritmusok lényege?*
365. Miért előnyös a scan-line algoritmus?
366. *Mi a területfelosztó algoritmusok alapgondolata?*
367. Mi a Warnock-algoritmus lényege?

368. *Ismertesse a sugárkövetéses algoritmust!*
369. Mi a raytracing és a z-buffer algoritmus közös előnyös tulajdonsága?
370. *Mitől függ a modelltér egy objektuma felületi pontjának színe?*
371. *Milyen típusú fényforrásokat adhatunk meg a vektorgrafikus rendszerekben?*
372. *Mit jelent a szórt háttérvilágítás (ambient light)?*
373. Mi jellemzi a távoli fényforrásokat?
374. *Jellemezze a pontszerű fényforrásokat!*
375. Hogyan definiáljuk a pontszerű fényforrásokat?
376. Hogyan adjuk meg a reflektor fényforrásokat?
377. *Mit nevezünk area light-nak?*
378. *Mi a szerepe a megvilágítási modelleknek a vektorgrafikában?*
379. *Milyen hatása van a renderelési időre, ha egy képelőállításnál több megvilágítási modellt is figyelembe veszünk?*
380. *Hogyan kapjuk meg egy pixel végleges színét?*
381. *Sorolja fel a megvilágítási modelleket!*
382. *Mit jelent az ambient light?*
383. *Milyen felületekre jellemző a diffúz fényvisszaverődés?*
384. *Mi a különbség a speculár és a total reflection között?*
385. Hogyan függ össze a felület minősége a megvilágítási modellekkel?
386. *Mit nevezünk lokális megvilágítási modellnek?*
387. *Mit nevezünk globális megvilágítási modellnek?*
388. *Adja meg a háttérfény megvilágítási egyenletét!*
389. *Adja meg a diffúz visszaverődés megvilágítási egyenlete!*
390. *Hogyan modellezhetők a színes fényforrások megvilágítási egyenletekkel?*
391. Milyen felületeken képződnek fényfoltok?
392. *Mikor kell alkalmazni a Phong-féle megvilágítási modellt?*
393. Mitől függ a fényfoltintenzitás csökkenése és ezt hogyan fejezzük ki a megvilágítási egyenlettel?
394. Miért nem alkalmazhatjuk felületi pontonként a megvilágítási egyenleteket?
395. *Miért egyszerűbb poliéderek megvilágítását kiszámolni?*
396. *Mutassa be a lokális megvilágítási algoritmusok lényegét?*
397. *Mit nevezünk shading-nak és melyek a legismertebbek?*
398. A különböző árnyalási algoritmusoknak milyen a műveletigénye és a képminősége?
399. *Mi a flat-shading lényege?*
400. *Milyen megvilágítási modellt használunk a flat-shading-nél?*
401. *Mi a Gourand shading lényege?*
402. *Ismertesse az intenzitás interpoláló árnyalás lépéseit!*

403. Milyen láthatóság algoritmussal együtt alkalmazzák a Gourand-árnyalást a grafikus kártyákban?
404. *Mi a lényege a Phong shading-nak?*
405. *Mi a legfontosabb különbség az előállított képen a Phong- és a Gourand-árnyalás között?*
406. *A fényfolthatástól eltekintve, melyik árnyalási algoritmus adja a legjobb képminőséget?*
407. *Melyik árnyalási algoritmusnak a legnagyobb a műveleti igénye?*
408. Hogyan lehet a Phong-árnyalást realizálni a hardverben?
409. *Hogyan használhatjuk a felületi részlet poligonokat?*
410. *Mit nevezünk texel-nek?*
411. Milyen típusú textúrákat ismer?
412. *Hogyan állítjuk elő a pixelek színértékeit textúrák alkalmazása esetén?*
413. *Mit értünk algoritmikus textúra alatt?*
414. Miért van szükség képjavító eljárásokra textúrák alkalmazása esetén?
415. *Mit értünk edge-antialiasing alatt?*
416. Mikor alkalmazunk anisotrop filteringet?
417. *Mi a hatása a bilinear filteringnek?*
418. *Mit jelent a mip-mapping?*
419. *Hogyan lehet perspektivikus hatást elérni textúrákkal?*
420. Mi a trilinear texture filtering?
421. *Mire használható az alpha-blending?*
422. *Milyen effekteket lehet a bump-mapping-al modellezni?*
423. Mi a bump-mapping algoritmus lényege?
424. *Mi az enviroment mapping lényege?*
425. *Miért használhatunk fel látható felület meghatározó algoritmusokat az árnyékok meghatározására?*
426. Hogyan kell módosítani a megvilágítási egyenletet, ha az árnyékokat is figyelembe kívánjuk venni?
427. *Sorolja fel az árnyékképzésben felhasználható fontosabb algoritmusokat!*
428. Hogyan határozzuk meg az árnyékokat a scan-line algoritmussal?
429. Mi jellemzi az árnyék testeket?
430. *Hogyan határozhatjuk meg az árnyékokat z-buffer algoritmussal?*
431. *Hogyan modellezhetjük az átlátszó és áttetsző testeket?*
432. *Milyen ismert láthatósági algoritmus kezeli az átlátszóságot?*
433. *Mit értünk objektumok közötti tükröződés alatt?*
434. Mit értünk reflection mapping alatt?

6.4. A FOTOREALISZTIKUS KÉPÁBRÁZOLÁS ALGORITMUSAI

Az eddig tárgyalt megvilágítási algoritmusok közös jellemzője volt, hogy csak a felületeket elsődlegesen, közvetlenül elérő fényt vették figyelembe. *A teljesen valósághű képekhez* – mivel a természetben a tárgyakat a fényforrás sok esetben nem direkt módon világítja meg – arra is szükség van, hogy *a jelenet objektumairól többszörösen visszaverődő, illetve az áttetsző objektumokon áthaladó fénysugarak által meghatározott fényviszonyokat is kezelni tudjuk. Ezeket az effektusokat figyelembe vevő eljárásokat a szakirodalom a globális megvilágítási modelleknek nevezi.*

A fotorealisztikus képábrázolásban a globális megvilágítás kezelésére, azaz

- a többszörös fényvisszaverődés,
- a többszörös tükröződés,
- a tárgyakon fénytöréssel áthaladó fény és
- a reális árnyékhatások (például kiterjedt fényforrások okozta elmosódott árnyékok az ún. penumbra)

ábrázolására az algoritmusok két csoportja alakult ki.

A korábbiakban a 6.3. fejezetben, a látható felület meghatározó eljárások között már bemutatott sugárkövetéses algoritmus továbbfejlesztett változata a *rekurzív sugárkövetéses algoritmus vagy raytracing több lépésben követi a visszaverődő vagy törő fénysugarakat és így képes a globális megvilágítási effektusok korrekt figyelembevételére.*

A fényterjedés fizikai törvényeit (fénysűrűség egyenlet) felhasználó radiosity algoritmusok ezzel szemben teljesen szétválasztják a látható felületek meghatározását és azok árnyalását. Az első, nézőpont független fázisban az objektumok és a rájuk eső fény összes kölcsönhatását képesek modellezni, majd ezt követően a hagyományos látható felület meghatározó és interpolációs technikákkal készítik el a megadott nézőpontból a képet.

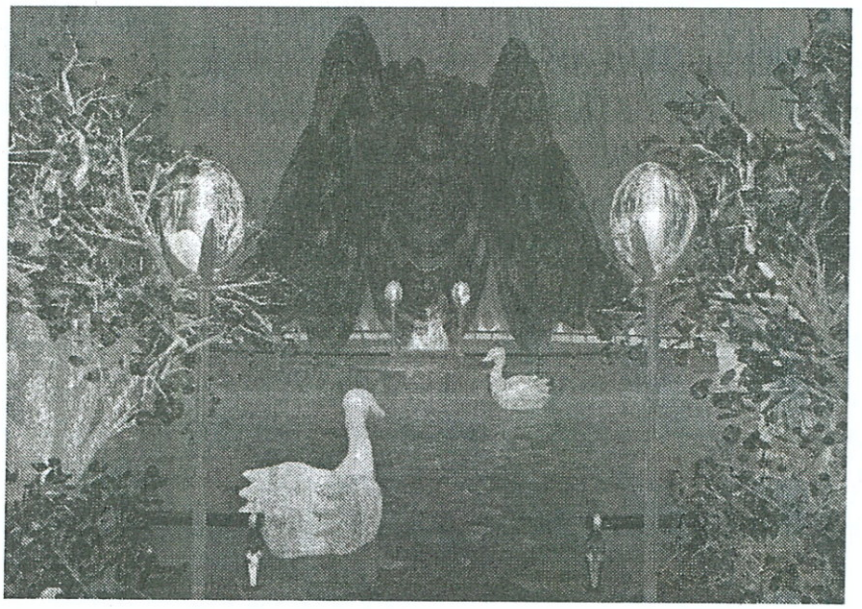
Fontos felhívni a figyelmet a nézőpontfüggő raytracing és a nézőpont független radiosity algoritmusok közötti legfontosabb különbségre. A nézőpontfüggő algoritmusok diszkrét egységekre bontják a képsík ablakát, előállítva ezzel a pixeleknek megfelelő olyan pontok halmazát, amelyekre aztán kiszámolhatják a megvilágítási értékeket. Ezzel szemben a nézőpont független algoritmusok a jelenetet tartalmazó teret osztják fel, majd a fényviszonyokat úgy dolgozzák fel, hogy elegendő információhoz jussanak bármely pont megvilágítási értékének meghatározásához.

! *A nézőpontfüggő raytracing algoritmusok könnyedén kezelik a tükröződéseket, de diffúz jelenségek esetén többletmunkára kényszerülhetnek. Ezzel szemben a nézőpont független radiosity algoritmusok hatékonyan modellezik a diffúz jelenségeket, de a tükröződés kíván tőlük többlet kapacitást.*

A következő két fejezetben a raytracing és a radiosity algoritmusokat mutatjuk be részletesebben.

6.4.1. Rekurzív raytracing

☝ A rekurzív raytracing a számítógépes grafika fotorealisztikus képábrázolásának legelterjedtebb algoritmusa, legintenzívebben fejlődő területe. Felhasználják a filmiparban (például Jurrasic Park, Terminator), a reklámkészítésben, a videoclip-gyártásban, tervek fotorealisztikus bemutatásában, sőt a művészetben is. Utóbbira jó példát mutatnak a WEB-en egyre terjedő művészi galériák (Artwork Gallery), melyekben számos művész publikálja alkotásait. Erre mutat egy példát a 251. sz. ábra.

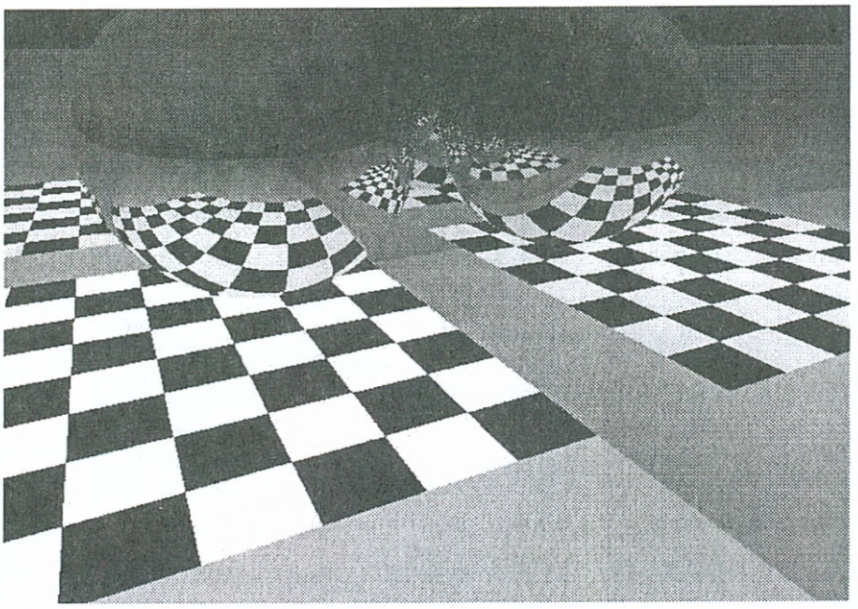


251. sz. ábra
Rekurzív sugárkövetéssel készített művészi kép

! A *raytracing algoritmussal készített képek jellegzetességei* könnyen felismerhetők. Ezek:

- *objektumok egymásra vetett árnyékai,*
- *többszörös tükröződések,*
- *átlátszóság kezelése fénytöréssel.*

Ezt tanulmányozhatjuk a 252. sz. ábrán.



252. sz. ábra
A rekurzív raytracing jellemző tulajdonsága
többszörös tükröződés ábrázolása

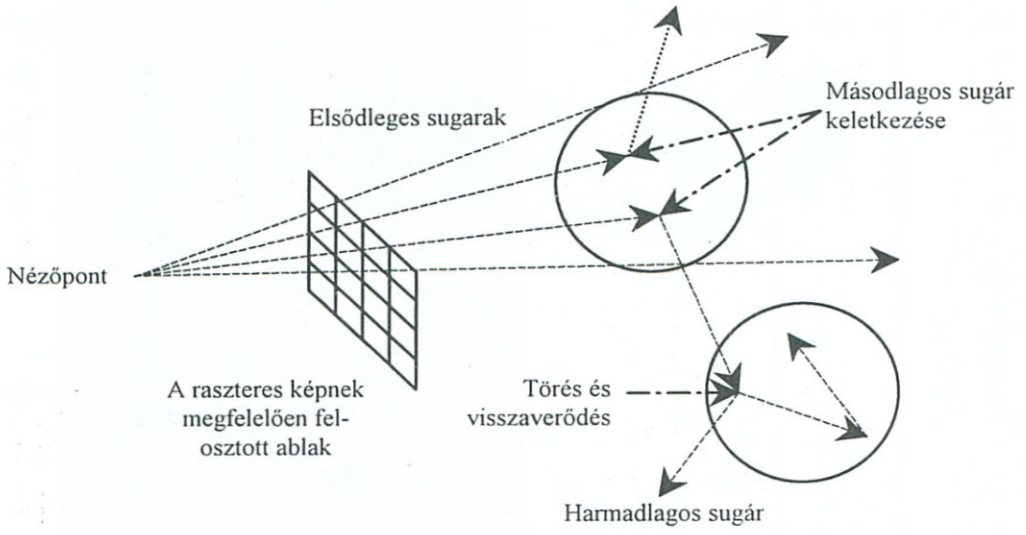
6.4.1.1. A rekurzív sugárkövetés algoritmusának elve

Az algoritmus alapelve egy mondatban megfogalmazható: a 3D-s modelltérben definiált objektumok és fényforrások leírása alapján egy nézőpontból a geometriai optika törvényei szerint követve a többszörösen visszaverődő és megtörő fénysugarakat egy fotominőségű képet számolunk ki és jelenítünk meg.

Ehhez *a 3D-s térben definiáljuk a megfigyelő nézőpontját és egy ablakot, melyet a képernyő felbontásának megfelelő számú „cellára" felosztunk.* (Tehát például 800x600-as felbontás esetén ez 480.000 cellát jelent.) !

Az ablak minden egyes cellája tehát a képernyő, illetve a frame buffer egy pixeljének felel meg. *A nézőpontból minden egyes cellán keresztül egy vetítősugarat indítunk és meghatározzuk ezek metszéspontját a modelltér objektumaival.* (Eddig a pontig az eljárás azonos a láthatóság meghatározó algoritmusok között tárgyalt egyszerű sugárkövetéssel.) Ezeket a sugarakat elsődleges vetítősugaraknak (Primary Rays) nevezzük. ! !

Az elsődleges sugarak az objektumok felszínéről visszaverődnek, illetve átlátszó testeken töréssel áthaladnak, így keletkeznek a másodlagos sugarak (Secondary Rays), *melyek szintén visszaverődhetnek, illetve megtörhetnek* (lásd 253. sz. ábra), *amivel újabb harmadlagos vetítősugarak keletkezhetnek és így tovább.* !



253. sz. ábra
A raytracing elve

! *Ez az eljárás akkor ér véget, ha a vetítősugár már nem ütközik egyetlen objektumba vagy fényforrásba sem, vagy az algoritmus lépésszáma már elérte a felhasználó által megadott maximális mélységet.* (Például ha ez 3, akkor az algoritmus már nem képez negyedleges vetítősugarakat.)

Az algoritmus elvéből a raytracing-nek három fontos tulajdonsága rögtön következik:

- a nézőpontból az ablak celláján keresztül indított vetítősugarak nyomkövetéséhez szükséges számítások cellánként egymástól függetlenek. Ez nagy lehetőséget biztosít az algoritmus hardver eszközökkel való párhuzamosítására;
- a raytracing több szempontból független az objektumok geometriai modellezésétől, végrehajtásához lényegében elegendő, hogy a felületi metszéspontok és normálisok meghatározhatók legyenek;
- programozástechnikailag a raytracing algoritmust könnyen lehet rekurzív módon (szubrutinok, altaszkok) implementálni, mivel például az elsőrendű és az „*n*"-edrendű sugarak követésekor lényegében azonos műveleteket kell leprogramozni.

! *Ha egy elsődleges sugár visszaverődve egy felület egy pontjáról „eltalál" egy fényforrást, és nem kell tükröződést illetve fénytörést figyelembe venni, akkor az elsődleges sugár képzésénél felhasznált cellának megfeleltetett pixel színe a megvilágítási modellből közvetlenül számítható. Ez azonban a pixelnek csak elsődleges színe lesz, mert előfordulhat, hogy a szóban forgó felületi pontot az algoritmus későbbi lépéseiben újabb sugár is érinteni fogja.*

A felületi pontoknak megfeleltetett pixel színének kiszámításához általános esetben több típusú másodlagos sugarat is nyomon kell követni. Ezek lehetnek

- visszatükröződött sugarak,
- megtörő sugarak, és
- a fényforrással összekötő sugarak.

Ha a fényforrással összekötő sugarak (ezekből annyit kell indítani, ahány fényforrás van definiálva a modelltérben), *melyeket árnyéksugaraknak (shadows ray) is neveznek, beleütköznek valamilyen objektumba mielőtt a fényforrást elérnék, akkor a felületi pont árnyékban van és a megfelelő fényforrást ki kell hagyni a felületi pontot árnyaló megvilágítási egyenletből.*

Az előzőekben bemutattuk a raytracing elvi működését, de egy nagyon egyszerű számítással rögtön láthatjuk, hogy ebben a formában az algoritmus viszonylag nagyobb kapacitású gépeken is reális idő alatt működésképtelen lenne.

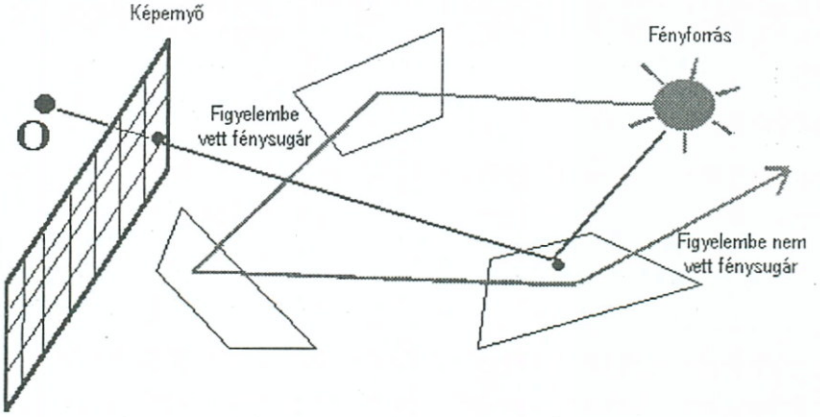
Nem csúcsminőségű monitorok felbontása is olyan, hogy a képenként kb. 1 millió pixelnek megfelelő számú vetítősugarat kellene indítanunk az algoritmus egy lépésében (!!). Ha a modelltérben csak 10 db objektumunk van, ez 10 milliós nagyságrendű metszéspontszámítást jelent.

Ezért nagyon fontosak azok a módszerek, melyekkel az algoritmus műveletigényét csökkenteni tudjuk.

6.4.1.2. A raytracing műveletigényét csökkentő eljárások

6.4.1.2.1. Backward Raytracing

Mivel a fényforrásokból kiinduló fénysugarak többsége olyan, hogy nem jut el az ablakon keresztül a nézőpontba, jelentősen csökken a számításigény, ha ezekkel nem foglalkozunk (lásd 254. sz. ábra).



254. sz. ábra
A raytracing csak a nézőpontba eljutó fénysugarakat veszi figyelembe

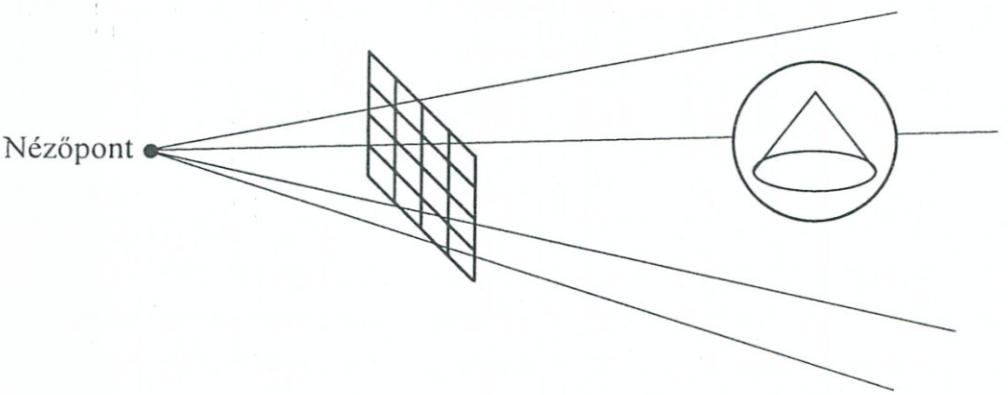
Ezért az algoritmus csak azokat a fénysugarakat veszi figyelembe, melyek az ablak celláin keresztül eljutnak a nézőpontba. Ezt úgy éri el, hogy *a fény terjedésével ellentétben nem a fényforrásokból kiinduló sugarakat kezdi vizsgálni, hanem azokat a nézőpontból indított vetítősugarakat elemzi, melyek visszafelé eljutnak a fényforrásba.* Ezt a megoldást „hátrafelé történő sugárkövetésnek", vagy *Backward Raytracing*-nek nevezik.

6.4.1.2.2. Bounding Volumes

A raytracing legműveletigényesebb része a vetítősugarak és az objektumok metszéspontjainak kiszámítása, célszerű ezek számát tehát csökkenteni.

Ezt célozza a raytracing során a befoglaló testek (Bounding Volumes) alkalmazása. Ennek során legtöbbször gömböket használunk fel. Ez azt jelenti, hogy olyan legkisebb méretű gömböket definiálunk a modelltérben, melyek objektumainkat teljes mértékben tartalmazzák.

A befoglaló gömbökkel úgy tudjuk csökkenteni a metszéspontok számításának mennyiségét, hogy csak azokat a vetítő sugarakat vesszük figyelembe a metszéspontszámításnál, melyek a befoglaló gömböket is metszik (lásd 255. sz. ábra).



255. sz. ábra
A vetítősugár metszéspontok számításának csökkentése befoglaló gömbökkel

Egy egyenes és egy gömb metszéspontja (nem metszi, metszi, érinti) matematikailag relatíve egyszerűen meghatározható, és ebből egy egyszerű teszttel kiküszöbölhetjük azokat a vetítősugarakat, melyek biztosan nem metszik a gömb által tartalmazott objektumot.

Legyen O a nézőpont, P az ablak egy cellájának megfelelő pont, R_0 egy r sugarú gömb középpontja 3D-ben és $\underline{o}$, $\underline{p}$, $\underline{r}_0$ az ennek megfelelő helyvektorok. Ekkor az O-ból kiinduló, P-n áthaladó egyenes paraméteres vektoregyenlete

$$\underline{r} = \underline{o} + t \cdot (\underline{p} - \underline{o})$$

a gömbbé pedig

$$(\underline{r} - \underline{r}_0)^2 = r_o^2$$

Az első egyenletet a másodikba helyettesítve és rendezve a döféspont t paraméter értékére egy

$$\alpha \cdot t^2 + \beta \cdot t + \gamma = 0$$

alakú másodfokú egyenletet kapunk, ahol

$\alpha = (\underline{p} - \underline{o})^2$

$\beta = 2 \cdot (\underline{p} - \underline{o}) \cdot (\underline{o} - \underline{r}_0)$

$\gamma = (\underline{o} - \underline{r}_0)^2 - r_o^2$

A diszkrimináns vizsgálatával az egyenes és a gömb metszése könnyen megállapítható:

- $\beta^2 - 4 \cdot \alpha \cdot \gamma > 0$ két metszéspont van
- $\beta^2 - 4 \cdot \alpha \cdot \gamma = 0$ érintési pont van
- $\beta^2 - 4 \cdot \alpha \cdot \gamma < 0$ nincs metszéspont

Ez alapján a befoglaló gömbökkel gyorsított raytracing esetén:

- először teszteljük, hogy melyik vetítősugár metszi a befoglaló gömböket, és a továbbiakban csak ezekkel a vetítősugarakkal foglalkozunk;
- csak a gömböt metsző esetben határozzuk meg a vetítősugár és a gömbben lévő objektum metszéspontját.

Ezzel az eljárással a gyakorlati tapasztalatok szerint a metszéspontszámítások mennyisége több nagyságrenddel csökkenthető.

6.4.1.3. A raytracing alkalmazásának korlátai

A raytracing-gal igen jó minőségű képeket állíthatunk elő, de az algoritmus alkalmazásának korlátai is vannak. E problémák közül mutatunk be néhányat.

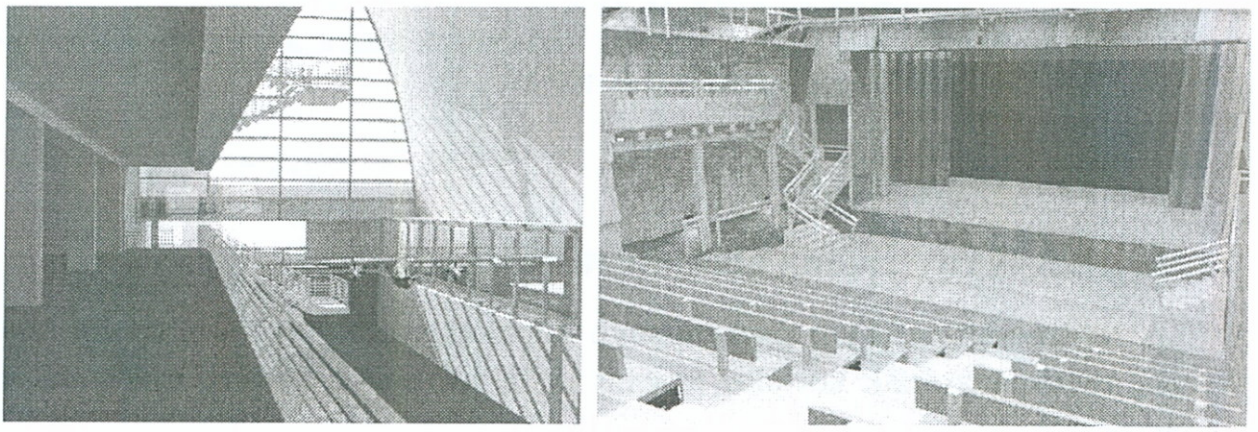
A raytracing algoritmusok a többlépcsős rekurzív eljárás miatt igen érzékenyek a kerekítési hibákra. Ez különösen akkor vezethet hibákra, ha nagy mélységben hajtjuk végre az algoritmust. Így például előfordulhat, hogy egy felületi pontból indított sugár többszöri visszaverődés után számítási hiba miatt visszajut az indító pontba és így önmagát árnyékolja. A kerekítési hibák miatt pöttyös felületű objektumok is megjelenhetnek a képeken. E nem kívánatos jelenségek ellen az algoritmus fejlesztői tűréshatárok bevezetésével próbálnak védekezni.

A raytracing két objektum közötti indirekt diffúz megvilágítást elvileg nem tudja kezelni. Erre a klasszikus példa egy világos felületű sík, mely fölött egy gömb, a fölött pedig egy fényforrás található. Ekkor a gömb alsó, síkfelőli oldalát indirekt fénynek kellene megvilágítani. Ennek kezelését raytracing esetében csak az ambient fénnyel tudjuk megoldani, ami nem mindig vezet realisztikus képhez.

A raytracing, mint képpontosság algoritmus raszterfüggő, emiatt a nézőpont kisebb megváltoztatása esetén is a teljes renderelést ismételten végre kell hajtani.

6.4.2. A radiosity algoritmus

A radiosity eljárás az egyik legmodernebb renderelési módszer, mely *a felületek közötti fényenergiacsere fizikai törvényszerűségeire alapozva dolgoztak ki. Ezért ezzel a módszerrel a kiterjedt fényforrások és a diffúz visszaverődés teljes értékűen, egzakt módon modellezhető.* Emiatt a radiosity eljárással teljesen valósághű, fotominőségű képeket készíthetünk a modelltér jeleneteiről (lásd 256. sz. ábra).



a) b)

256. sz. ábra
A radiosity eljárással készített fotominőségű képek
a) külső nappali fény; b) elektromos fény

Az eljárás alapja a fizikából ismert általános fénysűrűség egyenlet, melyből bizonyos egyszerűsítésekkel határozzák meg a felületek fénysugárzását. Emiatt a szakirodalomban a radiosity eljárást a fénysűrűséggel való renderelésnek is szokták nevezni (Rendering with Radiance).

A radiosity algoritmus alkalmazása esetén először a jelenet összes objektumának felületére meghatározzák a visszavert és a felület saját sugárzásából adódó fénymennyiséget, melyből származtatható a globális megvilágítási modell. Ennek erőforrásigénye és így a gépidőszükséglete általában a többi renderelési algoritmushoz képest többszörös. A fényviszonyokat ugyanakkor csak egy alkalommal kell kiszámítani. Így például *egy nézőpont* (kameraállás) *megváltoztatása esetén* – feltéve, hogy a jelenethez tartozó fényforrások és objektumok helyüket nem változtatják – *az új kép a megismert egyszerű láthatósági algoritmusokkal már nagyon gyorsan kiszámítható a változatlan megvilágítási modell miatt.* (Ez lényegesen gyorsabb, mint egy raytracelt kép előállítása.)

A korábbiakban vizsgált árnyalási algoritmusok a fényforrások és a megvilágított felületek egymásra gyakorolt kölcsönhatását (például a másodlagos, indirekt megvilágítást) nem vették figyelembe. *A radiosity eljárásnál viszont minden felületet fénysugárzónak tekintünk, és a fényforrások sem pontszerűek, hanem véges kiterjedésű felületekkel lesznek modellezve.*

Ennek megfelelően a jelenetet véges számú, diszkrét felületelemre (Patches) osszák fel. Minden felületelemnek véges mérete van, és a teljes felületén egyformán visszaver és ki is sugároz fényt.

Az algoritmust egyszerűsítve, azzal a feltételezéssel mutatjuk be, hogy minden felületelem a Lambert-törvények szerint sugároz ki és ver vissza fényt. Ekkor n darab felületelem esetén az „i"-edik felület által kisugárzott B_i fényenergia

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j \cdot F_{ij} \qquad 1 \le i \le n$$

B_i-t radiosity-nek nevezik, jelentése az idő és felületegységre vonatkoztatott saját és visszavert sugárzás energiája.

A képletben

E_i = az „i"-edik felületelem által kisugárzott saját fényenergia

ρ_i = a visszaverési tényezője az „i"-edik felületelemnek

F_{ij} = a „j"-edik felületelem által kisugárzott energiának az „i"-edik felületelemre eső része

Ha az előző lineáris egyenletrendszert megoldjuk, minden egyes felületelemre megkapjuk a B_i kisugárzás értékét. (A B_i ismeretlenek száma n db, és szintén n db egyenletünk van.)

A felületelemek sugárzásértékeinek meghatározását követően, ezeket mint a megvilágítási modellben felhasználandó intenzitásértékeket, bármelyik láthatósági és árnyalási algoritmussal elő tudjuk állítani a jelenet képét.

? ELLENŐRZŐ KÉRDÉSEK A 6.4. FEJEZETHEZ

435. *Milyen megvilágítási modellel kezelhetjük a többszörösen visszaverődő és megtörő fényt?*
436. *Mi a rekurzív sugárkövetés lényege?*
437. *Mi a radiosity eljárás legfontosabb jellemzője?*
438. Mi a raytracing és a radiosity algoritmusok közötti legfontosabb különbség?
439. *Melyik fotorealisztikus képábrázoló algoritmus modellezi hatékonyabban a tükröződést, illetve a diffúz visszaverődést?*
440. Milyen felhasználási területei vannak a rekurzív raytracingnak?
441. *Melyek a raytracinggel készített képek fontosabb jellegzetességei?*
442. *Ismertesse a rekurzív sugárkövetés algoritmusát!*
443. Milyen lehetőséget lehet kihasználni a raytracing algoritmus hardver realizációja során?
444. Mi jellemzi a raytracing programozás technikáját?
445. *Mikor lesz egy pixelnek elsődleges színe?*
446. *Milyen célt szolgálnak az árnyéksugarak?*
447. Miért nem alkalmazhatjuk a raytracing algoritmust „tiszta" formájában?
448. *Mit értünk backward raytracing alatt?*
449. *Miért alkalmazunk befoglaló testeket a raytracing során?*
450. Sorolja fel milyen problémák jelentkezhetnek a raytracing algoritmus alkalmazása során!
451. *Jellemezze a radiosity algoritmus felhasználhatóságát és fizikai elvét!*
452. *Mely esetben lassú és mikor gyors a radiosity eljárás?*
453. *Miben különbözik a radiosity eljárás az összes többi árnyalási algoritmustól?*

VII.

A SZÁMÍTÓGÉPES GRAFIKA SZABVÁNYAI

Az elmúlt évtizedekben a számítógépes grafika az informatika egyik legdinamikusabban fejlődő ágává vált. Egyre több és nagyobb teljesítményű hardver eszközt fejlesztettek ki a grafikus alkalmazások céljaira, az eszközöket egyre több cég gyártotta. Kezdetben a grafikus szoftvereket eszközfüggően kellett programozni, ez azonban egyre kevésbé felelt meg az igényeknek.

Egyrészt *a felhasználók már nem fogadták el függő helyzetüket az eszközgyártóktól, másrészt a grafikus szoftvereket fejlesztő szoftverházaknak is elemi érdekévé vált, hogy az egyre bonyolultabb és növekvő fejlesztési költségekkel előállított programcsomagokat minél több hardver plattformon értékesíteni tudják.*

Mindezek következtében a hardver portabilitás követelménye egyre inkább általánosan elfogadottá vált és ez egy szabványosítási folyamatot indított be a számítógépes grafikában. Ez kezdetben egyes nagyobb gyártók által rendszeresített házi szabványokban öltött testet, majd az 1980-as évek közepére már a nemzetközi szabványügyi szervezetek által elfogadott szabványok által teljesedett ki. (Például a 2D grafikát szabványosító GKS-t az ISO 1985-ben fogadta el.)

A számítógépes grafika szabványosításához jelentősen hozzájárult a grafikus felhasználói felület (GUI = Graphical User Interface) tömeges elterjedése is. (A Windows-os ablakozó felület legfontosabb szabványait 1985-ben és 1987-ben fogadták el.)

Az egységesítés egyre több területre is kiterjedt, ezek közül a fontosabbak ma a következők:

- *grafikus felhasználói felület szabványai,*

- *grafikus programcsomag szabványok,*
- *a felhasználói programok és a grafikus programcsomagok közötti interfészt, az API-t (Application Program Interface) meghatározó szabványok,*
- *a grafikus rendszerek közötti adatcserét biztosító fájlszabványok,*
- *a színkezelés, a betűformák szabványai,*
- *a különböző hardver csatolókra* (BIOS, driverek) *és grafikus eszközökre vonatkozó szabványok.*

Az INTERNET és az új hardver eszközök (például a grafikus kártyák) a 90-es években az API-k újabb szabványosítási hullámát indította el (OPEN-GL, DIRECT 3D, JAVA 3D).

Ebben a fejezetben egy rövid áttekintést adunk a legfontosabb raszter- és vektorgrafikus programcsomag szabványokról és a megfelelő API-król.

7.1. RASZTERGRAFIKUS SZABVÁNYOK

7.1.1. A grafikus felhasználói felület szabványa az X-WINDOWS

Az X11 szabványt, azaz az X-WINDOWS System v. 11-et a Massachusetts Institute of Technology adta ki 1987 szeptemberében. Ezt csaknem valamennyi számítástechnikai gyártó elfogadta és az 1990-es évek szoftver technológiájának egyik legfontosabb elemévé vált.

Az eredeti X11-et további release-k sorozata követte, jelenleg az X11 Release 6-nál tartunk.

Az X Windows System hardver független, a raszteres grafikára alapozott ablakozó megjelenítés szabványa.

A grafikus munkahely az X11 szerint képernyőt (screens), billentyűzetet (keyboard) és mutatóeszközt (egér, tablet) kezelő architektúra, melyen az X szerver program fut. (Felhívjuk a figyelmet, hogy a szabvány terminológiája eltér pl. a LAN-nál megszokottól.) Ez lehet például egy PC, mely az X szervert futtatja, vagy egy X-terminál, mely beégetett vagy programként futó X-szervert tartalmaz.

Az X-szerver feladata a grafikus megjelenítés (2D-s rajzolás) és a felhasználó inputjainak a fogadása. Az X-szerver hálózati kliensekkel, az úgynevezett alkalmazásokkal (applications) tarthat kapcsolatot, melyek számára hozzáférést biztosít a munkahelyhez.

A X11 felhasználói felülete a Windows-ból jól ismert ablakozó rendszer. Az ablakok méretezését és általában menedzselését a Windows Manager program végzi, mely az X-szerver számára szintén egy kliens.

Az ablakozó rendszer eseményvezérelt a mutató eszköz és a billentyűzet által. (Esemény lehet a mutató gomb lenyomása/felengedése, billentyű lenyomása stb.)

Az X11 architektúráját a 257. sz. ábra mutatja be.

257. sz. ábra
Az X11 architektúrája

Fontos megjegyezni, hogy az X11 R5 1992 szeptemberében fontos lépéseket tett a számítógépes grafika más szabványaival való kapcsolat megteremtésére:

- az X Colour Management System lehetővé tette az eszközfüggetlen színkezelést a nemzetközileg szabványos színterek használatával,
- bevezette a vektorgrafikus PHIGS szabvány (lásd 7.2. fejezet) minta implementációját (PHIGS-SI).

Az X11-es grafikus felület szabványt a „nyílt rendszerek" szabványai is átvették. Ezek:

- az OSF (Open System Foundation) által kiadott OSF-.Motif és
- az X Open Group által kiadott Open Look.

7.1.2. Az SRGP (Simple Raster Graphics Package)

Amikor a Windows-os felületű rajzoló programokkal (például PAINT, PHOTOSHOP stb.) egy kört rajzolunk a képernyőre, nem gondolunk arra, hogy tulajdonképpen egy rasztergrafikus szabványkörnyezetet használunk.

! Ezt *a szabványt egyszerű rasztergrafikus programcsomagnak, vagy az angol megnevezése után rövidítve SRGP-nak (Simple Raster Graphics Package) nevezik.*

! *Az SRGP tulajdonképpen a legalapvetőbb rasztergrafikus funkciókat megvalósító programegységekre vonatkozó szabvány, mely X11 alatt alkalmazható.*

Az SRGP szabványunk megfelelő programokat az összes elterjedt platformokon (IBM kompatíbilis PC-k, MacIntosh gépek stb.) megvalósították, ezek forráskódja az INTERNET anonymus ftp szervereiről is letölthető.

Az SRGP legfontosabb alkotóelemei a rasztergrafikus primitívek, melyek

- vonalak,
- ellipszisek,
- sokszögek és
- szövegek

lehetnek. A primitívek tulajdonképpen képelem generátorok, melyeket felhasználásukkor kell felparaméterezni (például kör esetén a középpont és a sugár megadásával).

Az SRGP a konkrét primitíveket egy raszteres felépítésű képmemóriában tárolja, melynek mérete az eszközfüggetlenség biztosítása érdekében nem függ a konkrét konfigurációban alkalmazott monitor felbontásától. (Ezt érzékelhetjük akkor, ha olyan rasztergrafikus objektumot rajzolunk, melynek csak egy része jelenik meg a képernyőn.)

Az SRGP egyes programrealizációi a szokásos rasztergrafikus funkcionalitást biztosítják. Ezek felsorolásszerűen:

- a grafikákhoz szín, vonalvastagság, vonalstílus, kitöltő szín és textúra rendelhető,
- lehetőség van bittérképes betűformák kezelésére,
- logikai eszköz- és eseménykezelés biztosított,
- a képernyőkezelés ismeri a vásznakat, a háttérszínt, a kivágó téglalapot,
- a rasztergrafikus objektumok között megengedettek a logikai rasztermüveletek.

7.2. VEKTORGRAFIKUS SZABVÁNYOK

E fejezetben a nemzetközi szabványügyi szervezetek által elfogadott GKS és PHIGS szabványokat és ezek továbbfejlesztett változatait mutatjuk be röviden.

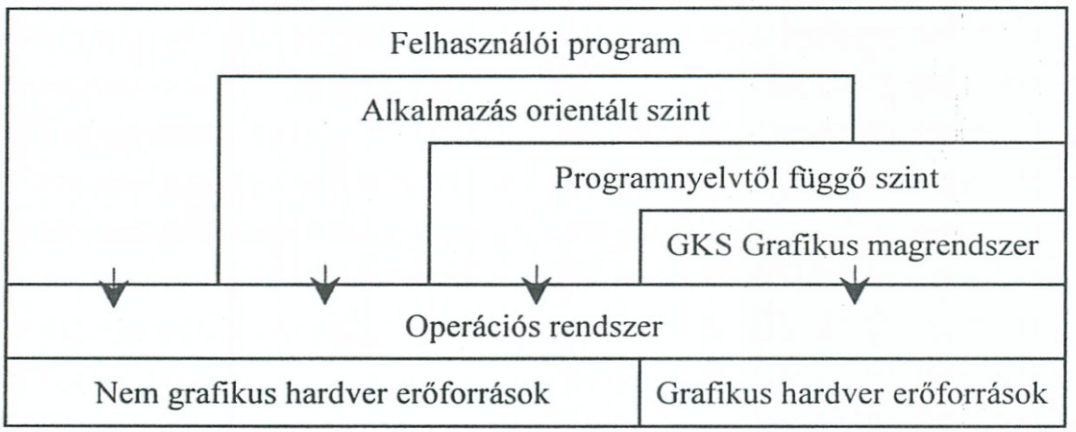
7.2.1. A GKS és a GKS-3D

Az első nemzetközi grafikai szabvány a német kezdeményezés alapján kidolgozott *grafikus magrendszer (Graphical Kernel System = GKS), mely a 2D-s vektorgrafikára vonatkozott. A GKS-nek ezt az első változatát 1985-ben fogadta el az ISO* (ISO 7942 – 1985).

A célkitűzések és követelmények, melyeket a GKS szabvánnyal el akartak érni a következők voltak:

- egységes illesztési helyet meghatározni a grafikus rendszerek és az egyedi alkalmazások között,
- a felhasználástól független, számítástechnikailag hatékonyan realizálható könyvtár definiálása a vektorgrafikus rendszerek fontosabb funkcióira,
- a grafika területén minél több általánosítható követelményt lefedni a szabvánnyal,
- elválasztani a grafikus rendszerek alap- (ún. mag) funkcióit a magasabb szintű modellezési funkcióktól,
- a szabvánnyal egy fejlesztési irányt mutatni a készülékgyártók számára.

A követelményeknek megfelelően a GKS egy rétegmodellt határozott meg a teljes grafikus rendszer számára (lásd 258. sz. ábra).



258. sz. ábra
A GKS rétegmodellje

A GKS-t a speciális programnyelvektől függetlenül definiálták, ezért a GKS és az alkalmazások közé be kellett iktatni egy programnyelvtől függő szintet (language binding). Ez átfordítja a GKS számára a programnyelv specifikusan közölt funkciókat és paramétereket.

Vegyük észre, hogy minden szint közvetlenül hozzáférhet – az alatta lévő szinteket „megkerülve" – az operációs rendszerhez (az ábrán ezt a ↓ szemlélteti). Ez viszont csak akkor megengedett, ha nem zavarja a rendszer hordozhatóságát és platform függetlenségét.

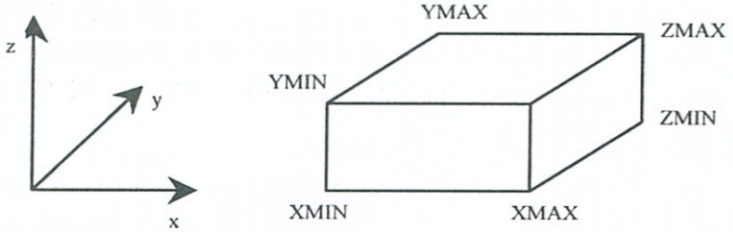
A grafikus hardver erőforrásokhoz pl. egy grafikus munkahelyhez a GKS egy munkahely illesztési hely szabványt határoz meg. Ez alapján készülékfüggő drivereket lehet írni, melyek lefordítják a GKS funkciókat a konkrét hardver speciális „nyelvére".

A felhasználói programokkal a GKS egy standard felhasználói interfészen keresztül tartja a kapcsolatot. A szabvány továbbfejlesztése a programnyelvtől függő részben ma már kezeli FORTRAN, PASCAL, ADA és C nyelvi kapcsolatokat (ISO 8651-4 – 1995).

! *Az egyre teljesítőképesebb hardverek eredményezték a GKS szabvány továbbfejlesztését. Így jött létre a GKS-3D szabvány, melyben a GKS-t 3D eljárásokkal és funkciókkal bővítették* (ISO 8805 (E) – 1988 szeptember).

A GKS szabványt több hardver és operációs rendszer platformon is megvalósították. (Ilyen például az Alpha processzorra a DEC GKS.) *Ezek* számítógépes megjelenési formájukat tekintve *legtöbbször egy alprogram könyvtárban öltenek testet. Ezért a grafikus szoftverfejlesztők a GKS-t tulajdonképpen egy felhasználói programozói interfész (API) formájában látják.*

Így például, ha egy 3D-s téglatestet akarnak definiálni koordinátasíkokkal párhuzamos oldalakkal, akkor egy alprogramot hívnak meg, melynek átadják a következő paramétereket: XMIN, SMAX, YMIN, YMAX, ZMIN, ZMAX.



259. sz. ábra
Téglatest definiálása GKS-ben

! *A GKS a GKS-3D funkciókkal kiegészítve a vektorgrafika következő területeit fedi le:*

- *színkezelés* (színpaletta definiálása),
- *transzformációk* (vetítés, 3D-s ablak definiálás stb.),
- *grafikus pimitívek* (sokszög, kitöltött terület, 2 és 3D-s szöveg stb.),
- *koordináta-rendszer, skálázás, rácsok,*
- *objektumok definiálása* (kör, téglatest, ellipszoid stb.),

- *térgörbék és felületek definiálása* (kontúrvonalak, háromszög-közelítésű felületek, függvénnyel megadott felület háromszögközelítéssel stb.),
- *láthatósági eljárások (huzalvázas megjelenítés, árnyalt felületek, poligonok rendezése fedettség szerint* stb.).

Ezeket a funkciókat a programozó a GKS könyvtárban lévő alprogramok felhívásával érheti el, mely alprogramokat fordítást követően bekapcsol a programjába.

7.2.2. A PHIGS, PHIGS+ és a PEX

A PHIGS vektorgrafikus programcsomag szabvány létrejötte volt a normakészítők másik válasza a GKS-3D mellett arra a helyzetre, hogy a GKS első változata csak a 2D-s vektorgrafika magfunkcióit szabványosította.

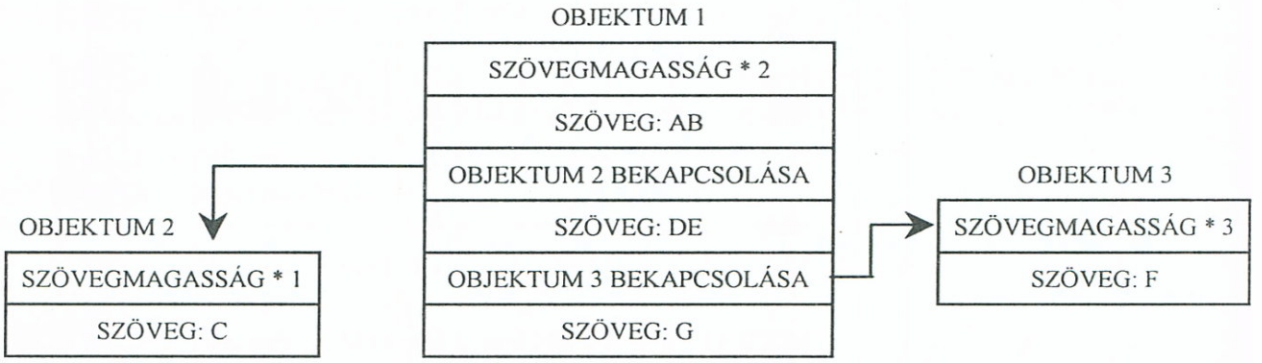
A PHIGS a szabvány angol elnevezésének: Programmers Hierarchical Interactive Graphics System rövidítéséből származik. *Ezek szerint a PHIGS a programozók hierarchikus, interaktív grafikus rendszere*. A szabványt az *ISO 1989-ben fogadta el* (ISO 9592).

A PHIGS szabvánnyal főként a tervezőszoftverek egységesítését szerették volna elérni, a norma funkcionalitását tekintve döntően megegyezik a GKS-3D-vel. A lényeges különbségeket a szabvány neve is kifejezi:

- a PHIGS támogatja a vektorgrafikus objektumok közötti hierarchikus kapcsolatok felépítését, ezeket egy modelladatbankban az ún. CSS-be (Central Structure Storage) a központi struktúra tárolóba integrálja;
- az adatbázisszerkezet a szabványban úgy került megfogalmazásra, hogy a programok a modelltér műveleteket interaktív módon is képesek legyenek végrehajtani.

A hierarchikus kapcsolatok létesítését a PHIGS-ben szöveges adatok példáján mutatja be a 260. sz. ábra.

A PHIGS szabvány – melynek kapcsolata a legfontosabb programnyelvekkel, így például FORTRAN, ADA, C is kidolgozásra került (ISO 9593 – 1990) – a programozó számára egy hatékony eszközt biztosított a 3D-s objektumok adatbázisban való feldolgozásához.



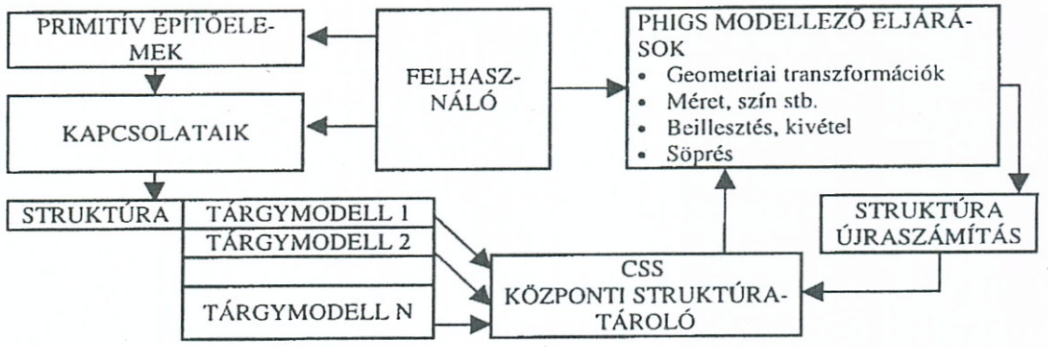
OBJEKTUM 1 MEGJELENÍTÉSKOR: ABcDEFG

260. sz. ábra
Példa a hierarchikus felépítésű objektumadatbázisra a PHIGS szabvánnyal

Az 1990-es évek elejére viszont a grafikus hardver teljesítményének növekedése és a realisztikus képelőállítást biztosító renderelés igénye egyaránt szükségessé tette a szabvány továbbfejlesztését. *A PHIGS+ szabványt* (ISO 9592-4 PHIGS Plus Lumiere and Surfaces 1992) *1992 végén adták ki. Ez már a túl absztrakt huzalvázas megjelenítést meghaladva kitér a különböző megvilágítási és árnyalási modellekre és eljárásokra is, és beépíti a szabványba a térbeli görbék és felületek modellezésének legújabb eredményeit.*

A PHIGS a PHIGS+ bővítéssel a következő elemekből épül fel:

- A primitívek egyik csoportja geometriai, ide tartoznak a szokásos poligonok, poliéderek kitöltött felületek, NURBS (csak a PHIGS+-tól) stb. Emellett a szabvány ismeri a szöveges és raszteres primitívek mellett a különböző mennyiségi (matematikai, statisztikai) primitíveket és a speciális szervezési primitíveket is (például egy úthálózat gráfja). A primitívekhez egyedi és csoporttulajdonságok is hozzárendelhetők, például színmodellek.
- *A primitívekből tárgymodelleket állíthatunk össze és ezeket struktúrákba szervezhetjük. A CSS lehetővé teszi a primitívek, tárgymodellek és struktúrák adatbázisszerű feldolgozását.*
- A CSS objektumaira a szokásos adatbázisműveletek (visszakeresés, létesítés, törlés, csoportosítás) mellett végrehajthatjuk a vektorgrafikában szokásos transzformációkat is (skálázás, mozgatás, koordináta-rendszer választás stb.).



261. sz. ábra
A CSS alkalmazása a PHIGS-ben

- A modelltér jeleneteit különböző nézőpontokból ábrázolhatjuk, ehhez a szabvány biztosítja a szükséges 3D–2D-s vetítéseket és kivágást.
- A megjelenítéshez a PHIGS ismeri a láthatósági és megvilágítási és árnyalási modelleket (Flat shading, egyes implementációkban Gouraud shading PHIGS+-tól).
- A felhasználóval való kapcsolattartás eszközeként alkalmazhatók a lokátor eszközök, a poligon rajzolás, a kijelölés, értékadás stb.

Az egyes platformokon a PHIGS, mint API egy programkönyvtár formájában áll a programozók rendelkezésére, mely több mint 400 funkcionális egységet tartalmaz.

A PHIGS implementációi a legkülönbözőbb számítógéptípuson működnek a PC-től a mainframekig. Ilyenek például

- SUN PHIGS a SUN cég UNIX gépein,
- graPHIGS az IBM AIX platformján.

A PEX (PHIGS Extension on X) az X-Windows-System bővítése a PHIGS-hez, illetve a PHIGS+-hoz. Ez *egy illesztési helyet ad meg a PHIGS és a X11 között, ami lehetővé teszi az X szerver szolgáltatások igénybevételét a PHIGS-en belül.* A PEX által támogatott elemek:

- képernyő,
- pixeles bitmap-ek,
- billentyűzet és egér,
- CSS.

A PEX nélkül is lehet a PHIGS-et X11-ben használni, de ekkor az X-Server csak egyszerű, az X11-ben meghatározott 2D-s primitíveket tud ábrázolni, ezért ez esetben a PHIGS 3D-s megjelenítési funkciókat programmal kell átkonvertálni az X-Server számára.

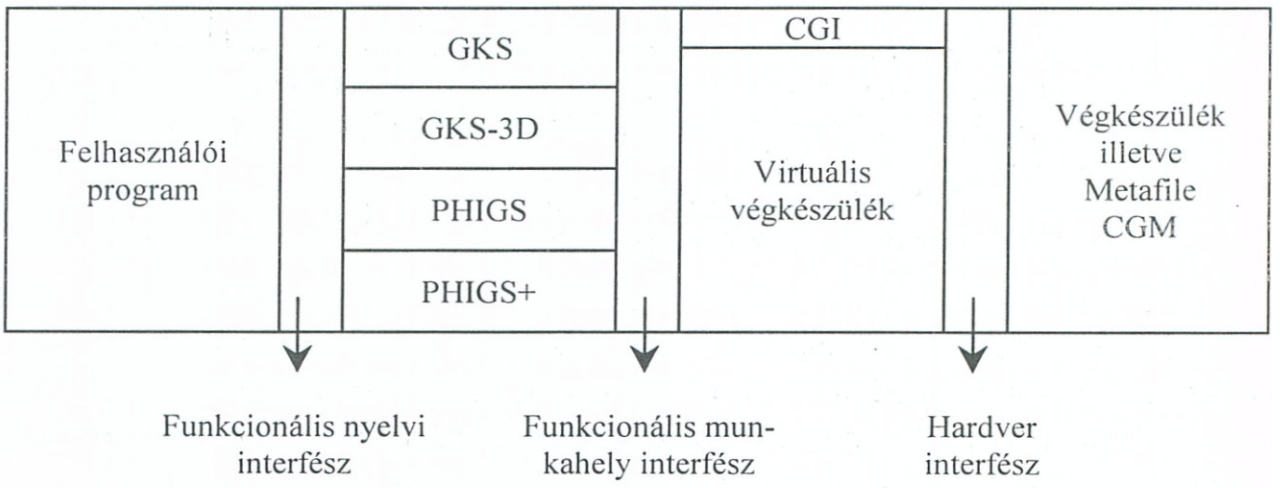
7.2.3. A CGM és a CGI

A CGM (Computer Graphics Metafile) szabvány 1987 óta létezik (ISO 8632) és a grafikus információk úgynevezett Metafile-ba történő archiválásának fájlformátumát rögzíti. Ez kiterjed

- egy funkcióhalmazra,
- az egyes funkciókra való programkódolási és paramétermegadási konvenciókra, mely a különböző architektúrák közötti információcserét biztosítja,
- a struktúra adatokra és
- a grafikus adatokra.

A CGI szabvány (Computer Graphics Interface) egy virtuális végkészülékhez való illesztési helyet határoz meg. Alkalmazása esetén a PHIGS, GKS programok a CGI szabványban rögzített funkcionális „munkahely" illesztési helyen keresztül cserélnek adatot a hardverrel. (A CGI nélkül minden grafikus programcsomaghoz speciális készülékdriver-eket kellene írni és működtetni.)

A CGI, CGM és a PHIGS(+), valamint a GKS-3D viszonyát mutatja be a 262. sz. ábra.



262. sz. ábra
A vektorgrafikus szabványok összefüggései

7.3. A SZÁMÍTÓGÉPES GRAFIKA LEGÚJABB SZABVÁNYAI

A GKS, GKS-3D, PHIGS és PHIGS+ lényegét tekintve egy grafikus szoftverkernel leírását adja meg, ezt a CGI még egy hardver interfésszel

egészíti ki. Az 1990-es években, különösen a 3D-s gyorsító chipek fejlődésével egyre nagyobb gondot okozott, hogy a hardverben realizált láthatósági, megvilágítási és árnyalási algoritmusokra ezek a szabványok nem vonatkoztak. Ez a korábbi grafikus szabványok alapján kifejlesztett szoftverek hordozhatóságában is egyre nagyobb gondot jelentett: egy-egy konkrét készülékkonfigurációhoz és ablakozórendszerhez a szabványos szoftvereket csak pótlólagos ráfordítással lehetett illeszteni. Emiatt alakult ki a számítógépes grafikában a szabványosítás újabb hulláma az 1990-es években.

7.3.1. Az OpenGL

A SiliconGraphics fejlesztői a fenti problémákra kerestek megoldást, amikor a korábbi IRIS GL grafikus szoftverük tapasztalatainak felhasználásával *egy szoftver interfészt specifikáltak OpenGL néven a grafikus kártyák hardveréhez. Ez hamarosan defacto szabvánnyá vált és egyre több platformon létrehozták az OpenGL-nek megfelelő szabványos 3D-s grafikus programkönyvtárakat.*

A C programnyelvhez való illesztéssel (language binding) az OpenGL kb. 250 bázisrutint kínál fel az alkalmazásfejlesztők számára. Ezek egy részével 2 illetve 3D-s geometriai objektumokat vagy raszterprimitíveket (pixelmezők) lehet definiálni, másrészük a primitívek képének a hardver framebuffer-re számára való átadását biztosítja. Ez utóbbiak közé tartoznak:

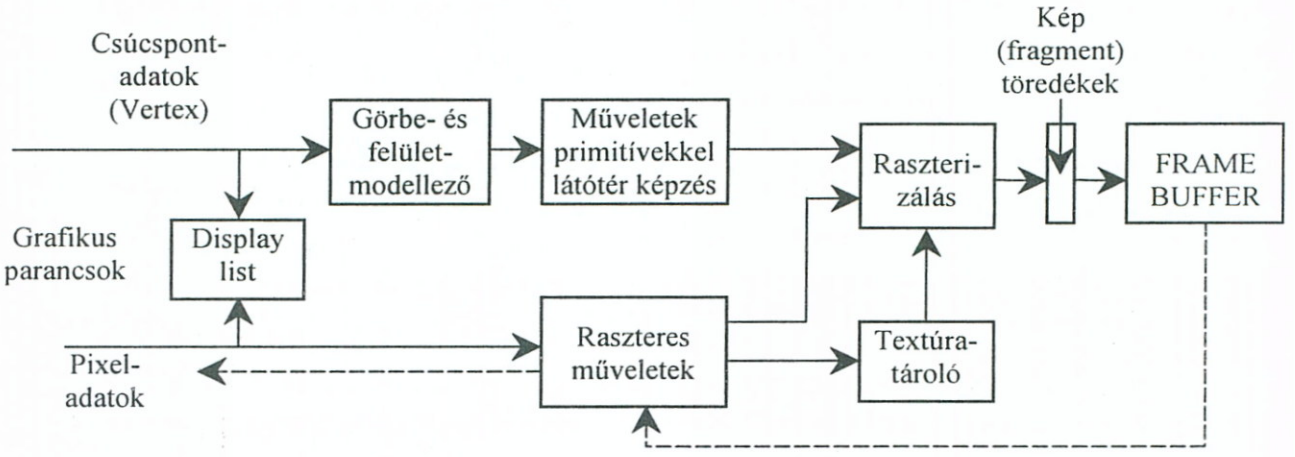
- a modellező transzformációk (eltolás, forgatás stb.),
- a látótér képző transzformációk (vetítések),
- a láthatósági és árnyalási algoritmusok.

Az OpenGL (a GKS és PHIGS-el szemben) ezen túlmenően a „magasabb szintű" renderelő funkciókat is tartalmaz. Ezek:

- *texture mapping,*
- *alpha blending* (átlátszóság),
- *légköri effektek* (köd, pára, füst),
- *anti-aliasing.*

Az OpenGL konkrét inplementációit az előbbieken túlmenően további GL-Utility (GLU) könyvtárak egészítik ki, melyek például további renderelő és objektumkezelő funkciókat (anyagi tulajdonságok, fényforrás definíciók, jelenet-adatbank, speciális testek: gömb, henger, torusz, NURBS stb.) támogathatnak.

A 263. sz. ábra mutatja be az OpenGL alapvető felépítését és működését.



263. sz. ábra
Viewing pipeline az OpenGL-ben

Ezek szerint az *OpenGL által megkapott parancsok egy viewing pipeline-n futnak végig. A parancsokat a rendszer vagy azonnal végrehajtja, vagy egy átmeneti tárolóba kerülnek (display list) ahonnan később kiolvasva végrehajthatók. A grafikus információkat a rendszerrel csúcsponti adatok (vertex) formájában közölhetjük. Ha ezek térbeli görbe vagy felületre vonatkoznak (tartópontok), akkor a görbe és felületmodellező ebből OpenGL primitíveket állít elő. A következő lépésben az OpenGL a primitíveket egy vetítéssel leképezi a látótérbe. Ebben a fázisban történik a fényforrások kihatásainak figyelembevételéhez szükséges számítások elvégzése is.*

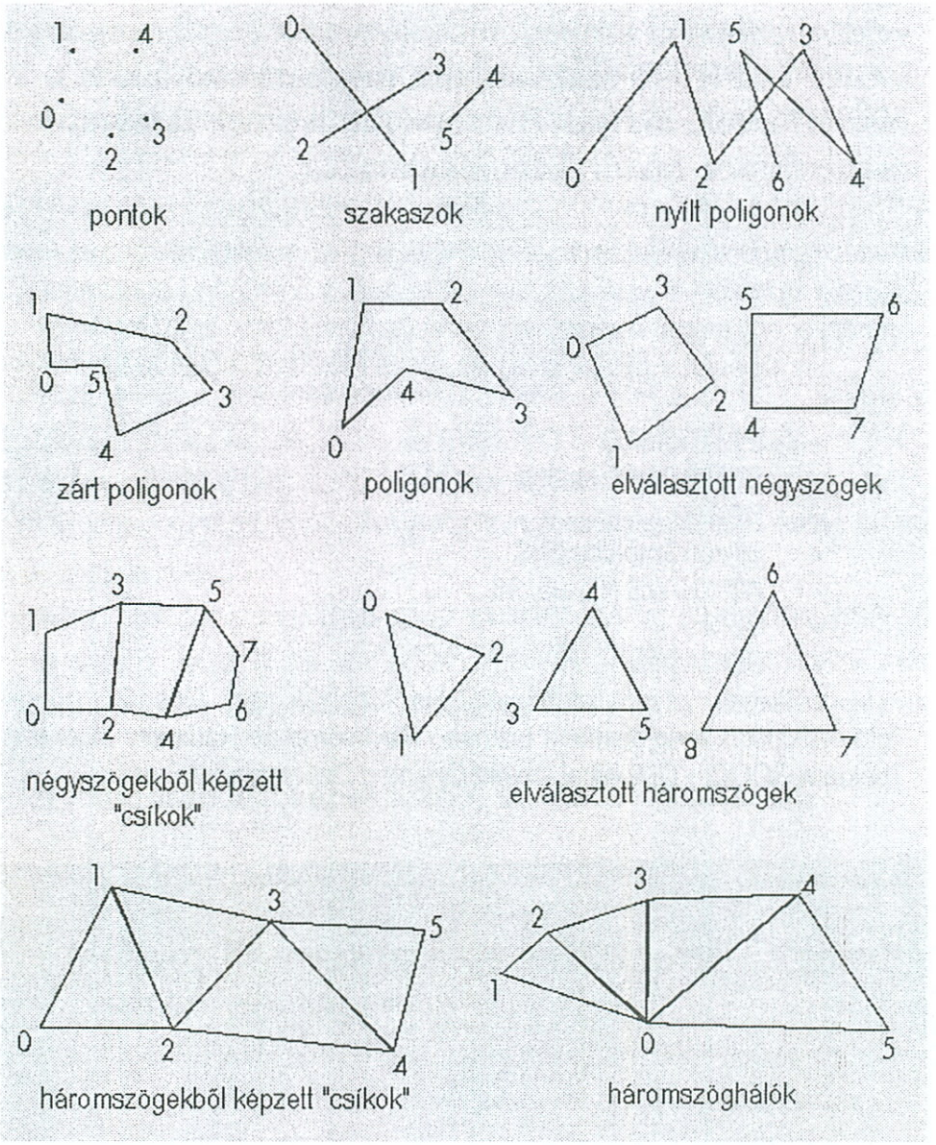
Az úgynevezett raszterizálással megtörténik az átalakítás 2D-s egész koordinátákra. Ekkor veszi a rendszer figyelembe a különböző Bitmap és pixelmezőket, melyek közé számítják a textúrákat is. A raszterizálás során kapott pontokhoz a színértékek és a pont z értéke is hozzá lesz rendelve.

A raszterizálás után kapjuk a képtöredékeket (fragment), amelyek alapján *z-buffer eljárással határozzák meg a végső láthatóságot.*

Az OpenGL (lásd az ábrán a szaggatott vonallal jelölteket) megengedi, hogy a frame-bufferből adatokat kiolvassunk, illetve egyes tartományokat kimásoljunk.

A viewing pipeline utolsó lépését a frame-bufferből való képkirajzolást a monitor képernyőjére nem az OpenGL, hanem az alkalmazott ablakozó rendszer hajtja végre. Ez összhangban van az OpenGL alapkoncepciójával, mely szerint a konfiguráció és grafikus input készülék vezérlése a Windows feladata. Így például a Windows kezeli a képernyő ablakait, a színtáblázatot stb. (Az ezekkel kapcsolatos információkat viszont az OpenGL természetesen felhasználja a renderelés során.)

Az OpenGL a 264. sz. ábra szerinti primitíveket különbözteti meg.



264. sz. ábra
Az OpenGL primitívjei

A primitívek geometriai formájából következik, hogy ezeket a csúcspontjaik (Vertex) koordinátáinak megadásával specifikálhatjuk, erre az OpenGL 24 funkciót biztosít. A koordináta adatok mindig 4D homogén koordinátákban értelmezendők, így például a „gl Vertex 0" az $x = 0$, $y = 0$, $z = 0$ és a $w = 1$ koordinátákat jelenti.

!

Az alapvetően pontokból és szakaszokból felépülő primitívek mellett analitikus görbéket és felületeket is lehet konstruálni az OpenGL segítségével. Ezeket kontrollpontjaikkal kell megadni, amelyből a görbe és felületmodellező egy térbeli pontrácsot állít elő (Bezier modellezéssel), mely megadja a felület vagy görbe sokszög illetve egyenesszakasz közelítését.

!

Az OpenGL háromfajta font típust kezel:

- egyeneszakaszokból előállított fontok (Hershey-Fonts), melyek vektorbetűnként szabadon skálázhatók (ez lényegét tekintve azonos a GKS és PHIGS szabványban meghatározottakkal),
- raszterfontok, melyek Bitmap-ként lesznek tárolva,
- raszterfontok textúraként alkalmazva.

A modelltranszformációkat, a kivágást és a látótérvetítést az OpenGL-ben a csúcspontok mátrixokkal való szorzásával kezelik a homogén koordinátás 4D térben. Ehhez a rendszer egy mátrix veremtárolót vezet (gl Push Matrix és gl Pop Matrix utasítások).

Az OpenGL-t gyakorlatilag az összes fontosabb platformra implementálták. A fontosabbak:

- IBM RS/6000
- SUN SPARC station
- HP-700
- SiliconGraphics IRIS
- WINDOWS NT, 95, 98
- OS/2
- DEC/Alpha

Léteznek olyan rendszerek is, melyek az OpenGL renderelő funkcióit a fotorealisztikus megjelenítést biztosító Raytracing és Radiosity algoritmusokkal integrálják (például GOOD = Graphical Objekt Oriented Development System).

7.3.2. Direct X, Direct 3D és a Fahrenheit projekt

A Windows operációs rendszer alatt futó multimédiás és grafikus alkalmazásokat nagyon lelassította, hogy a Windows-Kernel egy kvázi terelőútként működött a felhasználói szoftver és a hardver (mindenek előtt a grafikus kártya) között. Ez motiválta a Microsoft céget, hogy kidolgozza a Direct X „házi" szabványt, mely lehetővé teszi a hardver erőforrásokhoz történő közvetlen és gyors hozzáférést.

A Direct X multimédiás és grafikus komponensekből áll, ezek közül a

Direct 3D a 3D-s vektorgrafikus modellezés és a 3D-s jelenetek képi ábrázolásának szabványa.

A Direct 3D API (Low-Level) funkcionálisan rendkívül hasonlít az OpenGL-re. Így például a 3D-s objektumokat poligonfelületekkel modellezi, azonosak a modelltér és megjelenítés transzformációs lehetőségei, hasonló a láthatósági, valamint a megvilágítási és árnyalási algoritmusok kezelése, a speciális effektek (pl. köd) biztosítása.

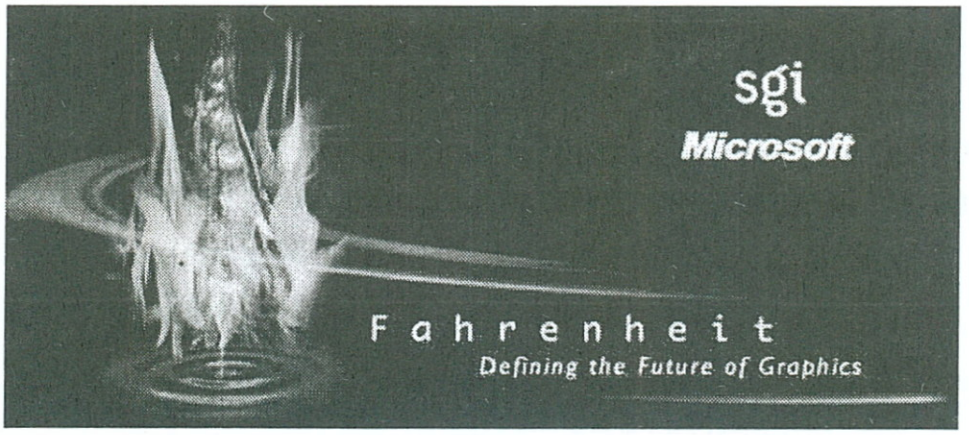
A Direct 3D jó animációs lehetőségeket biztosít, erre is visszavezethető, hogy főként a PC-s játékprogramoknál használják elterjedten.

A Direct 3D grafikai alkalmazásában jelentős hátrányt jelent, hogy lényegében csak a Windows operációs rendszerek alatt működőképes, emiatt a *Direct 3D-re alapozott szoftverek gyakorlatilag más platformokon nem futtathatók.*

Az OpenGL-el összehasonlítva a Direct 3D-nek viszont sokkal jobb a hardver támogatása. (Például a Pentium III processzor megjelenését követően pár hónap alatt optimalizálták a Direct X-et az új 3D-s utasításkészletre.) Vélhetően ez is hozzájárult ahhoz, hogy a *SiliconGraphics és a Microsoft* – bár e téren sokáig egymás konkurensei voltak – egy *közös projektet szervezett „FAHRENHEIT" néven. Ennek keretén belül egy olyan új szabványos API kidolgozását irányozták elő, mely az OpenGL és a Direct 3D továbbfejlesztése* lenne, oly módon, hogy e két korábbi szabvánnyal való lefelé kompatibilitást megtartaná.

A két cég által kötött megállapodás szerint az új szabványnak megfelelő felhasználói program interfész a WINDOWS 2000 operációs rendszerrel együtt kerül kiadásra, és *alkalmazhatósága átfogja a számítógépes grafika összes területét.*

A projektet szimbolizáló képet a 265. sz. ábra mutatja be.



265. sz. ábra
A Fahrenheit projekt

A két cég együttműködése során kifejlesztett API-val a számítógépes képfeldolgozás és grafika következő területeit célozták meg:

- Internet,
- számítógépes játékok,
- üzleti grafika és prezentáció,
- digitális képgenerálás,
- CAD, CAM

- orvosi és tudományos alkalmazások.

A Fahrenheit projekt a következő részekből áll:

- **Farenheit low-level API** lesz az elsődleges grafikus API a kommersz és a professzionális WINDOWS alkalmazások számára, és lényegében a Direct 3D, Direct Draw és az OpenGL funkcionalitását foglalja magába. Kompatíbilis lesz a korábbi alkalmazásokkal és hardver driverekkel, melyeket a Microsoft Direct 3D-re írtak meg;
- **Fahrenheit Scene Graph API** a magas szintű programnyelveken fejlesztők számára készül. Ez az SGI Scene Graph API-n alapul és egy magas szintű (high-level) grafikus adatbáziskezelést is biztosít;
- **Fahrenheit Large Model Visualization Extensions** az OpenGL Optimizer és Hewlett-Packard és Microsoft Direct Model API-k továbbfejlesztésével jön létre. Ezt az olyan igen nagyméretű grafikus alkalmazásokra és képi adatbáziskezelésre optimalizálják, mint például az autógyártás CAD/CAM rendszerei.

Az SGI és a Microsoft abban is megegyezett, hogy együttműködnek egy új grafikus készülékdriver fejlesztő eszköz (3-D Graphics Device Driver Kit) létrehozásában is.

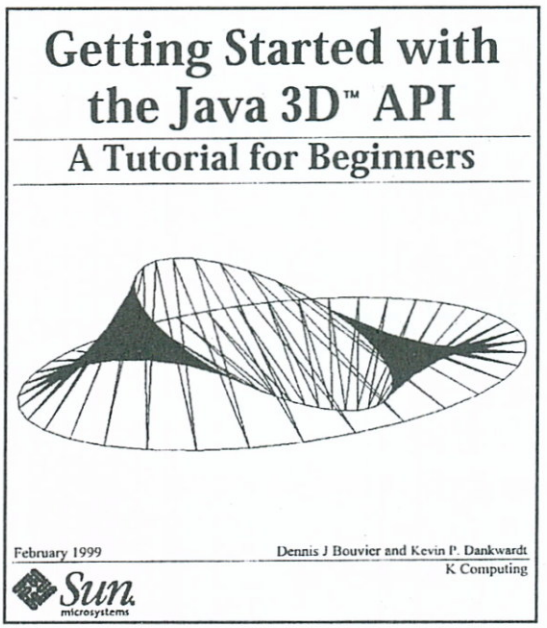
7.3.3. A Java 3D

Az 1996 májusában megtartott első JavaOne konferencián a JAVA nyelv programozói interfészének fejlesztésére a következőket fogadták el:

- Java 2D, a kétdimenziós grafikai interfész szabványa,
- Java 3D, a háromdimenziós grafika és virtuális valóság interfész szabványa.
- Java Animation, az animációk és ahhoz kötődő grafikus objektumok kezeléséhez szükséges programozói felület.
- Java Sound, a hang manipulációs felület szabványa.

Ezt követően a SUN cég 1998 decemberében jelentette be a Java 3D programozói interfész szabaddá tételét a Java programozók számára.

A Java 3D-hez többfajta leírás, oktatóprogram készült, ezek pl. a SUN cég szerveréről letölthetők a WEB-en (lásd 266. sz. ábra).



266. sz. ábra
A Java 3D oktatóprogram kezdőlapja a SUN honlapján

A Java 3D egy magas szintű (High-Level) programozói interfész, melynek segítségével a Java 3D „virtuális világegyetemben" (virtual universe), azaz a modelltérben interaktív módon 3D-s objektumokat definiálhatunk és renderelhetünk. Erre az ún. jelenet gráfok (scene graph) szolgálnak, melyeket képi és hang objektumok

- *geometriai,*
- *hang,*
- *fény,*
- *hely és irány,*
- *külső megjelenítés*

jellemzőivel definiálunk és állítunk össze.

A Java 3D-ben a szokásos gömb, kocka, kúp, henger testek definiálása mellett 3D-s csúcspontokkal reprezentált, egyenesszakasz és háromszög primitívekkel közelített objektumokat modellezhetünk. (Az eljárás lényegét tekintve megegyezik az OpenGL-nél bemutatottakkal.)

Lehetőség van a térbeli objektumok eltolására, forgatására skálázására, valamint egy nézőpontból és egy láthatósági ablaknak megfelelő perspektív vetítésre.

Különböző típusú fényforrásokat (pontszerű, fényszóró, irányított) határozhatunk meg, az árnyalásnál a Flat vagy a Gouraud shading alkalmazható. A Java 3D kezeli a textúrákat, az átlátszóságot és néhány speciális effektust is (pl. köd).

! *Az OpenGL és Direct 3D API-kkal való funkcionális különbségeket vizsgálva, mindenképpen meg kell említeni a Java 3D jó animációs képességeit és a sztereo hangforrások kezelését.*

? ELLENŐRZŐ KÉRDÉSEK

454. *Mi okozta, hogy a hardver portabilitás ma már általánosan elfogadott követelmény a számítógépes grafikában?*
455. *Sorolja fel, hogy a szabványosítás milyen területekre terjed ki a számítógépes grafikában!*
456. *Mi az X Windows System?*
457. *Az X11 szabvány szerint mit tartalmaz a grafikus munkahely?*
458. *Mi az X-szerver feladata?*
459. *Mi a feladata a Windows Manager kliensnek?*
460. Mutassa be az X11 architektúráját!
461. *Mi az SRGP?*
462. Milyen primitívek vannak az SRGP-ben?
463. Milyen funkciókat biztosítanak a SRGP egyes programrealizációi?
464. *Mi a GKS?*
465. Milyen követelményeket határoztak meg a GKS szabvánnyal szemben?
466. Ismertesse a GKS rétegmodelljét!
467. *Hogyan használhatják a szoftverfejlesztők a GKS-t?*
468. *Milyen funkciókat valósít meg a GKS és a GKS-3D?*
469. *Mi a PHIGS?*
470. Mi a közös és mi az eltérés a PHIGS és a GKS-3D között?
471. Milyen programnyelvekhez biztosít kapcsolatot (language binding) a PHIGS?
472. *Milyen új lehetőségeket biztosít a PHIGS+ a PHIGS-hez képest?*
473. Milyen primitíveket használhatunk a PHIGS+-ban?
474. *Mit biztosít a CSS?*
475. *Mi a PEX?*
476. *Mi a CGM?*
477. *Mi a CGI?*
478. Milyen problémát kellett a szabványosítással megoldani a 90-es években?
479. *Mi az OpenGL?*
480. *Milyen „magasabb szintű", új renderelő funkciókat tartalmaz az OpenGL a korábbi szabványokhoz képest?*
481. Mit tartalmaznak a GLU könyvtárak?

482. *Milyen lépésekből áll az OpenGL viewing pipeline?*
483. *Hogyan kezelhetők a térbeli görbék és primitívek az OpenGL-ben?*
484. *Mi történik a raszterizálás során az OpenGL-ben?*
485. *Milyen algoritmussal állítjuk elő a képernyőképet az OpenGL-ben?*
486. *Milyen a feladatmegosztás az OpenGL és a Windows között?*
487. *Hogyan adjuk meg a primitíveket az OpenGL-ben?*
488. Milyen fonttípusokat kezel az OpenGL?
489. Miért alakította ki a Microsoft cég a Direct X házi szabványt?
490. *Jellemezze a Direct 3D „házi" szabványt!*
491. *Milyen feladatot kíván megoldani az SGI és a Microsoft a „FAHRENHEIT" projektben?*
492. Milyen felhasználási területeket kívánnak lefedni a „FAHRENHEIT" projektben kidolgozandó API-kkal?
493. Milyen programozói interfészeket fogadtak el a JavaOne konferencián?
494. Hogyan juthatunk hozzá a Java3D oktatóprogramjához?
495. *Jellemezze a Java3D-t!*
496. *Milyen funkciókat biztosít a Java3D a megjelenítéshez?*

VIII.

A GRAFIKUS HARDVER ÉS SZOFTVER

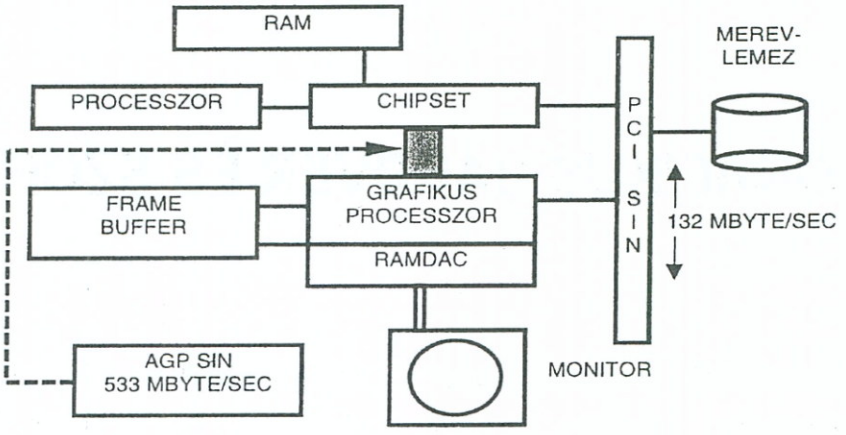
E fejezetben először a számítógépes grafika hardver követelményeit ismertetjük (processzor, grafikus buszok, memóriák, háttértárak és csatolóik). A grafikus perifériák közül részletesebben foglalkozunk a monitorral és a monitor vezérlő és 3D-s gyorsítókártyákkal és egy összefoglaló áttekintést adunk a számítógépes grafikában használatos további perifériákról. Ezt követően a legelterjedtebb professzionális grafikus programcsomagokat tárgyaljuk és néhány fontosabb freeware programcsomagot is bemutatunk.

8.1. A SZÁMÍTÓGÉPES GRAFIKA HARDVER KÖVETELMÉNYEI

E fejezetben a grafikus hardvert IBM kompatíbilis PC-s környezetben vizsgáljuk és nem foglalkozunk a nagyteljesítményű speciális grafikus „munkaállomásokkal" (Workstation), melyekről az érdeklődő például a Silicon Graphics cég honlapjáról (lásd irodalomjegyzék) könnyen beszerezhet információkat. A hardver követelmények tárgyalását a vektorgrafikára koncentrálva végezzük el, de ez a vektor- és rasztergrafika közötti különbség figyelembevételével értelemszerűen az utóbbira is alkalmazható. (Így például csak rasztergrafikus alkalmazásnál nyilvánvalóan nem kell odafigyelnünk a monitorvezérlőkártya 3D-s gyorsítóképességeire.)

A vektorgrafika hatékony alkalmazása és a számítógép hardver egységeinek teljesítménye között szoros kapcsolat van. Az egyik legfontosabb követelmény, hogy a vektorgrafikus objektumok feldolgozása, tárolása és megjelenítése során a számítógép hardver részegységeinek szorosan együtt kell működniük, ezért teljesítményjellemzőiknek is összehangoltaknak kell lenniük.

A vektorgrafikában meghatározó jelentőségű hardver rendszerelemeket a 267. sz. ábra mutatja be.



267. sz. ábra

A vektorgrafika hatékonyságát leginkább befolyásoló hardver elemek

Az egyes rendszerelemekkel szemben támasztott hardver követelmények meghatározásához, röviden tekintsük át a vektorgrafikus objektumok definiálásával kezdődő és a képernyőn való megjelenítésükig tartó teljes renderelési folyamatot, az úgynevezett *Viewing-Pipeline egyes lépéseit:*

1. Vektorgrafikus objektum definiálása és transzformációk a modelltérben, fényforrás és kameradefiníciók (modell- és világkoordináta-rendszer) vektorgrafikus adatbáziskezelés.

2. Nézet ablak definiálása (Windowing). Befoglaló testek (Bounding Volumes) kezelése (modell- és világ-koordináta-rendszer).

3. A jelenet leképezése a normalizált látótérbe (világkoordináta-rendszer és NPC tér) és kivágás (clipping) és hátsó felületek eltávolítása (Back-Face Culling). E lépésben az objektumokhoz tartozó poligonokat szukcesszíven ki kell számolni és deformálni *(modell és világkoordináta-rendszer, NPC tér).*

4. *Láthatósági és árnyalási algoritmusok (világkoordináta-rendszer, vagy NPC tér és/vagy képernyő koordináta-rendszer).*

5. Háromszögekre illetve vonalakra bontás (képernyő koordináta-rendszer).

6. Megjelenítés a képernyőn a frame-bufferből.

Nyilván az 1., 2., 3. (esetleg 4.) lépésekben meghatározó hardver elemek a processzor, a rendszermemória és a buszrendszer. Ha vektorgrafikus adatbáziskezelés is történik akkor szerepet kap a háttértár és a csatolója is. Mivel az alkalmazott koordináta-rendszerek lebegőpontosak, e lépések gyorsasága főként a processzor lebegőpontos műveletvégző képességétől függ.

Az 5. (esetleg 4.) lépésben a feladatokat a monitorvezérlő és a 3D-s gyorsítókártyák, a 6.-ban pedig a monitorvezérlőkártya és a monitor hajtja végre.

Ennyi bevezetés után tekintsük át a számítógépes grafikára alkalmas hardver legfontosabb jellemzőit.

8.1.1. Processzorok

Ha képeket, illetve mozgóképet (például számítógépes grafikával előállított animációt) akarunk feldolgozni, akkor a processzornak multimédia képességekkel kell rendelkeznie.

A számítógépes grafika és képfeldolgozás igényei miatt fejlesztették ki a Pentium MMX és vele kompatíbilis processzorokat. *Az MMX képességű processzorok kifejlesztésének célja tehát az volt, hogy a hardver képes legyen a multimédia egyre inkább növekvő igényeit kiszolgálni.*

Ehhez a processzor utasításkészletét bővítették 48 új MMX utasítással, melyekkel ún. MMX adattípusokat

- Packed byte = nyolc 8-bites adat
- Packed word = négy 16-bites adat
- Packed doubleword = két 32-bites adat
- Quadword = egy 64-bites adat

lehet feldolgozni.

Ezeknek az adatoknak a kezelését az MMX processzorokban egy új MMX pipeline oldja meg, melyhez Pentium MMX esetén a lebegőpontos regiszterstack 8 regisztere van hozzárendelve „új" MMX regiszterekként.

Ezek a processzorok képesek egy utasítással vektoros műveleteket (például 2D-s vektorok előjel nélküli összeadása) végrehajtani, ezért ún. SIMD (Single Instruction Stream Multiple Data Stream) architektúrájúak.

Az Intel következő processzora, a Pentium II is MMX képességű és más cégek is gyártanak MMX kompatíbilis processzorokat (AMD K6, CYRIX 6x86, IDT Centaur C6+).

Fontos hangsúlyozni, hogy az MMX processzorok csak a multimédiás képfeldolgozást gyorsítják és ezt is csak akkor, ha a megfelelő grafikus szoftvert optimalizálták az MMX utasításkészletre.

Ezért jelentőségük a számítógépes grafikában csak a raszteres képfeldolgozásra korlátozódik. Ez alól csak az AMD K6 processzor kivétel, melynek utasításkészlete az MMX utasítások mellett 21 további utasítást is tartalmaz a 3D-s transzformációkhoz.

A processzorok vonatkozásában az igazi áttörést a számítógépes grafika területén a Pentium III. processzor hozta meg. *A Pentium III. utasításkészletét kifejezetten a számítógépes grafika igényei szerint bővítették.* Ezt az új utasítások megoszlása is mutatja :

- *50 új lebegőpontos SIMD utasítás,*
- 12 új multimédia utasítás,
- 8 új cache-tár kezelő utasítás.

Ez jelentősen gyorsíthatja a számítógépes grafika egyes alkalmazásait a 3D-s vektoros műveletek hardver szintű végrehajtásával.

Gondoljunk arra, hogy például egy poligonokkal modellezett test mozgatásakor a sokszögek (például háromszögek) minden egyes lebegőpontos koordinátáját ki kell számítani az új pozícióban lényegében azonos műveletekkel. Ezért e műveletek párhuzamosítása, azaz vektoros végrehajtása (egy utasítással több számon azonos aritmetikai művelet elvégzése) jelentősen gyorsíthatja az új utasításkészletnek megfelelően átírt grafikus programcsomagokat, például a grafikus animációk megjelenítését.

Az előbbi tényezők mellett a 3D-s grafika szempontjából az is fontos, hogy a lebegőpontos számításokat az új utasításkészlet mellett 8 db új 128 bites lebegőpontos regiszter is támogatja.

Természetesen a Pentium III. előnyeit akkor lehet igazán kihasználni, ha az új processzor utasításkészletét az operációs rendszerek és a grafikus programcsomagok is alkalmazzák.

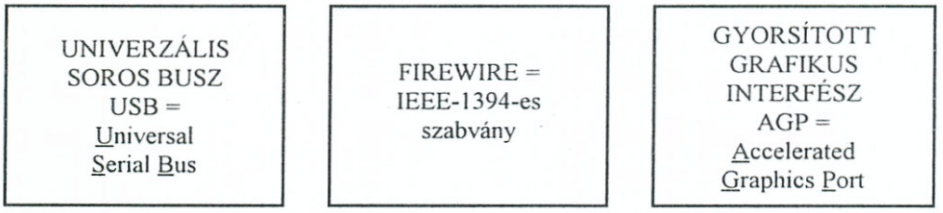
E téren – a korábbi processzorokkal kapcsolatban tapasztalatokkal ellentétben – felgyorsult a fejlesztés: az új processzort már támogatják a WINDOWS 98 és NT és a NOVELL újabb verziói és a DIRECTX API-t is optimalizálták az új 3D-s grafikus utasításokra.

8.1.2. Grafikus buszok és csatolók

A számítógépes grafika és képfeldolgozás gyors terjedése miatt új típusú buszokat és csatolókat is kifejlesztettek.

A képi információk, például textúrák adatátviteli igénye miatt meg kellett növelni a számítógépek buszának sebességét. Ezt a grafika mellett a nagy adatmennyiséget feldolgozó input/output eszközök (nagyfelbontású nyomtató, videokamera) terjedése is szükségessé tette.

Az új busztípusok a szabványosítás és felhasználóbarátság (plug and play) követelményeinek is jobban meg kívántak felelni. Az elmúlt években megjelent új buszokat a 268. sz. ábra foglalja össze.



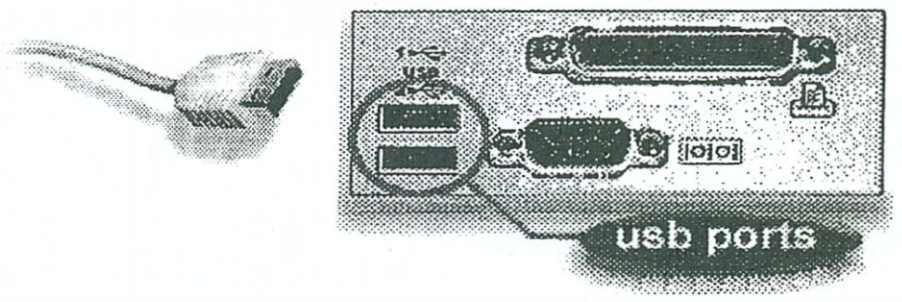
268. sz. ábra
Az új grafikus buszok

8.1.2.1. Az USB (Universal Serial Bus)

Az univerzális soros busz, azaz az USB egy olyan szabványosított csatlakozóaljzat és összeköttetés, amely

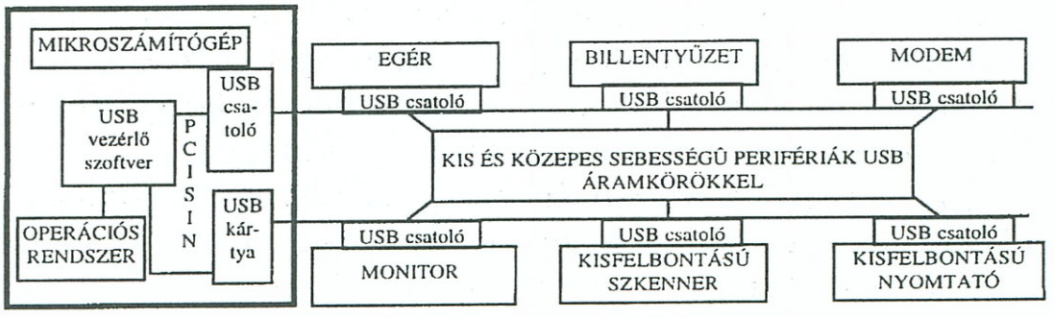
- *a billentyűzetcsatlakozót,*
- *az egércsatlakozót,*
- *a soros és párhuzamos portot*

egyetlen nagysebességű, soros átvitelt biztosító összeköttetéssel helyettesíti.



269. sz. ábra
Az USB port

Az USB felépítését és csatlakozását a mikroszámítógéphez mutatja be a 270. sz. ábra.



270. sz. ábra
Az USB csatlakozása a mikroszámítógéphez és a perifériákhoz

Az USB legfontosabb tulajdonságai a következők:

- 12 Mbit/sec átviteli sebesség,
- maximum 127 USB eszköz kiszolgálása.
- soros adatátvitel,
- elvileg az eszközök sorosan felfűzhetők (csak ha két kaput tartalmaznak),
- lehetővé teszi a plug és play perifériatelepítést,
- az eszközök tápellátása USB kábelen keresztül lehetséges.

8.1.2.2. A FIREWIRE

A FIREWIRE (IEEE-1394-es szabvány) létrejöttének oka, hogy multimédiában és grafikában egyre nagyobb szerephez jutnak a közepes teljesítményű átvitelt igénylő perifériák, azaz

- *a videokamerák,*
- *a nagy felbontású nyomtatók,*
- *a nagy kapacitású streamerek.*

A FIREWIRE jellemzői a következők:

- átviteli teljesítmény 400 Mbit/sec,
- max. 16 eszköz kezelését teszi lehetővé,
- a plug and play telepítést támogatja.

8.1.2.3. Az AGP (Accelerated Graphics Port)

A 3D-s renderelés évről-évre több memóriát igényelt, mellyel a grafikus kártyák frame bufferének kapacitása egyre nehezebben tudott lépést tartani.

Mivel a frame buffernek alkalmas memóriák jóval drágábbak a központi tár RAM memóriamoduljainál, ezért önként adódott a következtetés, hogy a RAM-ot használjuk fel az egyre inkább fotorealisztikussá váló grafikus megjelenítéshez szükséges textúra, háromszög stb. adatok tárolására.

Ez viszont csak úgy lehetséges – gondoljunk például mozgófilmek megfelelő sebességű megjelenítésére –, ha valamilyen eszközzel nagy teljesítményű adatátvitelt biztosítunk a monitorvezérlő-kártya és a főtár RAM memóriája között.

E gondolatok jegyében született meg *a gyorsított grafikus port, az AGP, mely közvetlen összeköttetéssel 266, illetve 533 Mbyte/sec átviteli sebességet biztosít a grafikus kártya és a főtár RAM között (1x, 2x-es AGP).*

Látható, hogy 2x AGP esetén az új sín átviteli teljesítménye kb. kétszerese a PCI sínének. Ezen túlmenően a grafikus rendszerek teljesítményét az AGP azzal is növeli, hogy ez a busz kizárólagosan csak a grafikus információk átvitelére szolgál. (A PCI rendszersínen a grafikus adatok mellett még számtalan más jellegű adat is áramlik.)

Az AGP alkalmazásához a RAM-ból természetesen megfelelő mennyiségű memóriát rezerválni kell. Ez a grafikus hardvertől, és az alkalmazástól függően 24–64 Mbyte is lehet. Ez tovább növeli a grafikus programcsomagok egyébként is jelentős memóriaigényét.

8.1.3. A grafikus rendszerekhez szükséges memóriák

A grafikus szoftverrendszerek legújabb verzióinak hatékony és elfogadható sebességgel történő működtetésében meghatározó a memóriaigény:

- *a grafikus szoftverek futtatásához „házi" használat esetén általában 16–32 Mbyte RAM szükséges,*
- *professzionális alkalmazások esetén a központi memóriaigény legalább 64–128 Mbyte.*

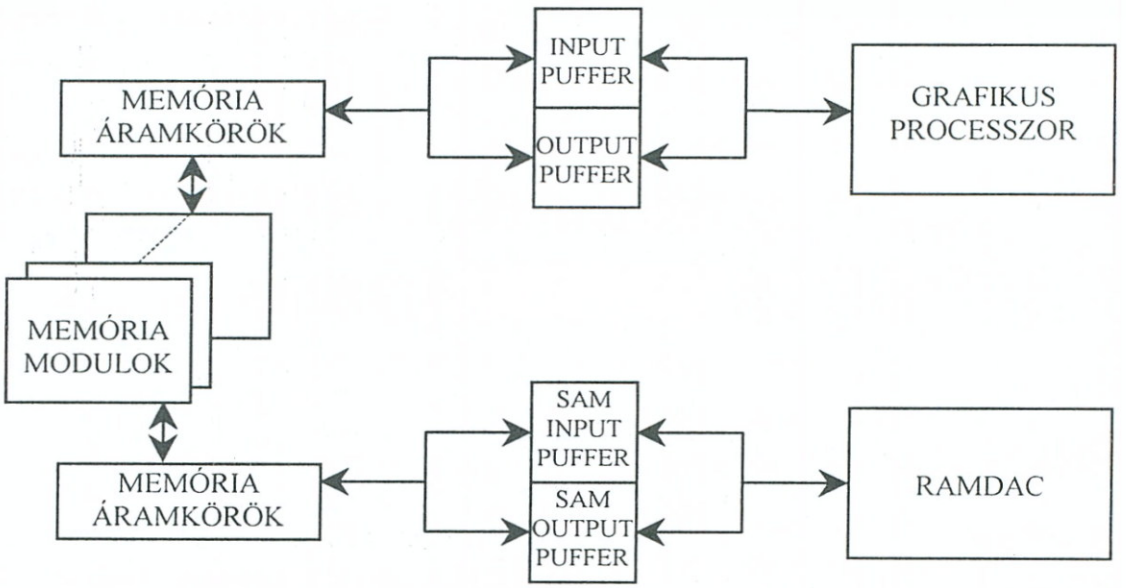
A grafikus szoftverek használata során memóriakapacitás mellett az sem közömbös, hogy gépünk korszerű, elég gyors memóriatípust tartalmaz-e.

A grafikus adatfeldolgozáshoz a piacon kapható memóriatípusok közül az SDRAM és a BEDO RAM a legmegfelelőbb. Az SDRAM (Synchrones DRAM) és a BEDO RAM (Burst Extended Data Output RAM) a képi adatokhoz gyors hozzáférést biztosítanak, melyet a „burst", azaz a csoportos hozzáférési technika bevezetésével és a processzor és a memória működésének szinkronizálásával érnek el. (A SDRAM-nál a memória írási és olvasási műveleteket a processzor működését meghatározó órajelhez szinkronizálják. A burst üzemmódban az adatokat megfelelő áramkörökkel blokkokba szervezik.)

! *Különösen a gyors renderelés miatt töltenek be fontos szerepet a grafikus kártyák memóriái. Ezek töltik be a frame buffer szerepét, ezért kétportosak, azaz egyidejűleg a grafikus processzorral és a monitor vezérlését ellátó digitális analóg konverterrel, a RAMDAC-al* (lásd 8.2.2.1. pont) *is képesek kommunikálni.*

A számítógép monitorán látható kép ebben az ún. videomemóriában (VRAM) kerül tárolásra, ennek minden egysége a képernyő egy képpontját határozza meg. A VRAM-oknak soros hozzáférésű (SAM = Serial Acces Memory) pufferjeik vannak, melyek a RAMDAC frekvenciájával működnek. Ezek és a VRAM között az adatok folyamatosan kétirányban automatikusan cserélődnek. (A soros működésnek az az értelme, hogy a monitorképernyőre való kirajzolás is sorosan történik.) A VRAM a monitor-vezérlőkártya grafikus gyorsító processzorával egy másik be/kimeneti pufferrel tartja a kapcsolatot (lásd 271. sz. ábra).

A speciális „kétirányú" működés jelentős számú pótlólagos áramkört igényel, ami ezt a memóriatípust megdrágítja.



271. sz. ábra
A VRAM sematikus felépítése

A legegyszerűbb VRAM-ok, az áramkör típusát tekintve, dinamikus RAM-okból épülnek fel. Ezek 500 Kbyte–2 Mbyte méretben a legtöbb általában szokásos számítógép felhasználásra (például programfejlesztés, szövegszerkesztés) megfelelőek, de a professzionális grafika igényeit nem elégítik ki. Ezért fejlesztették ki a frame buffernek és a speciális renderelő algoritmusoknak jobban megfelelő video RAM típusokat. Ezek közül néhány fontosabb:

- SGRAM = Synchrones Graphical RAM jellemzője, hogy működése a grafikus processzor órajelével összehangolt, így azt hatékonyabban képes kiszolgálni.
- A WRAM a saját belső buszától eltekintve lényegében VRAM. A belső busz viszont segíti a kommunikációt a grafikus processzor és a memória között.
- A 3DRAM-nak szokták nevezni azt a VRAM-ot, melyet speciális képfeldolgozó feladatokat ellátó hardver elemekkel egészítenek ki. Így például:
  - műveletvégrehajtó ALU-val a Z-pufferelés hardveres végrehajtására,
  - a képi információkat átmenetileg tároló cache-el.

A grafikus kártyák memóriaszükséglete a színkódolástól és a monitor felbontásától függ. Ezt mutatja be a 272. sz. ábra.

Színsíkok/ színek száma	Felbontás			
	640x480	800x600	1024x768	1280x1024
4 bit/16	256 kB	256 kB	512 kB	1 MB
8 bit/256	512 kB	512 kB	1 MB	1,5 MB
16 bit/>65 000	1 MB	1 MB	2 MB	3 MB
24 bit/>16 millió	1 MB	1,5 MB	2,5 MB	4 MB

272. sz. ábra

A memóriaszükségletet bitben úgy kaphatjuk meg, ha a pixelszámot összeszorozzuk a színkód bitszámával. Például: 800 x 600-as felbontás és 24 bites (3x8 bit) színkódolás esetén 800 x 600 x 24 = 5 760 000 bitre van szükségünk.

Meg kell jegyezni, hogy ez a számítás csak a monitor képernyőjén megjelenített kép tárolószükségletét tartalmazza. Ha a grafikus kártya hardverét olyan algoritmusokra is alkalmassá tették (pl. z-buffer, texture mapping), melyek további memóriát igényelnek, akkor a memóriaszükséglet ennek arányában növekedhet.

8.1.4. Háttértárak és csatolóik

A mágneses vagy optikai jelrögzítést alkalmazó háttértárakat a számítógépes grafikában adattárolásra, mentésre és továbbításra használjuk. Fontos felhívni arra a figyelmet, hogy egy konkrét grafikus alkalmazáshoz szükséges háttértároló eszközök fajtája és kapacitása nagyon függ a felhasználás jellegétől. Néhány ezzel kapcsolatos fontosabb szabály:

- *Ha egy vektorgrafikus rendszerrel úgy dolgozunk, hogy az előállított képeket és filmeket nem akarjuk tárolni, akkor ennek háttértárkapacitásigénye nem túl jelentős*, mivel az objektumok vektorgrafikus tárolásának csak minimális a helyigénye. *Ezért ez esetben a mai PC-ken általában rendelkezésre álló néhány Gbyte kapacitás bőven elegendő.*
- Ha a vektorgrafikus rendszerben jelentős mennyiségű objektum, építőelem, szimbólum, textúra könyvtárat kívánunk felhasználni, akkor az ehhez szükséges tárolókapacitást is figyelembe kell vennünk.
- *Ha a vektorgrafikus rendszerben előállított képeket tárolnunk kell*, illetve a munka során, pl. textúrák formájában raszteres képeket akarunk felhasználni, *akkor már lényegesen több közvetlen elérésű tárolókapacitásra (5–10 Gybte) van szükség*. Ennek mérlegelésénél különösen azt kell figyelembe venni, hogy *a fotorealisztikus rendereléssel készített képek tárolóigénye igen jelentős*. Ebben az esetben már egy komolyabb archiváló eszközről (például CD-RW vagy DVD-RAM meghajtó) is célszerű gondoskodni.
- *A tárolókapacitás igény akkor a legnagyobb, ha a grafikus rendszerrel filmeket akarunk készíteni és tárolni.*

Egy grafikus hardver konfiguráció kialakításánál figyelembe veendő eszközök a következők:

- merevlemez (Whinchester) 2–20 Gbyte kapacitással,
- CD-ROM, lemezenként 650 Mbyte kapacitással,
- CD-RW (írható–olvasható CD),
- DVD-ROM, lemezenként 18,8 Gbyte kapacitással,
- DVD-RAM (írható–olvasható DVD).

A CD és a DVD technika közötti választást az archiválandó adatmennyiség figyelembevételével dönthetjük el. Ha több gigabájtnyi mennyiséget kell időről időre elmentenünk, akkor célszerű a professzionális alkalmazásoknál az új DVD technikára alapoznunk.

A háttértár konfiguráció kialakításakor a megfelelő csatolót is ki kell választanunk. A szóba jöhető SCSI és EIDE eszközök fontosabb jellemzőikkel a következők:

SCSI 2:	• WIDE SCSI = az adatbusz 16 bites • Full WIDE SCSI = az adatbusz 32 bites • Bus Master üzemmód = nagy teljesítményű átvitel DMA-val az SCSI és a főtár között
SCSI 3:	• ULTRA WIDE SCSI-nek is nevezik (= szinkron BURST üzemmód) • üvegszál (fiber channel)
SCSI 3-RAID:	• hibatűrő rendszer pl. lemezcsere működés közben, melegtartalék
EIDE MODE4 és EIDE ULTRA DMA:	• megnövelt teljesítményű EIDE csatoló • BURST üzemmód

Egy grafikus rendszerben alkalmazott csatoló kiválasztásában döntő szempont az átviteli sebesség és a periféria csatlakoztatási lehetőségek. Az ezzel összefüggő fontosabb adatokat foglaltuk össze a 273. sz. ábra táblázatában.

Interfész	Sebesség	Csatlakoztatható perifériák és háttértárolók
soros port RS-232	115 kbit/sec	egér, modem
párhuzamos port Centronics	800 kbit/sec	nyomtató, szkenner
USB	12 mbit/sec	egér, modem, nyomtató, szkenner
EIDE MODE-4	133 mbit/sec	merev lemez, cd rom
EIDE ULTRA DMA/331	264 mbit/sec	merev lemez, cd rom
SCSI WIDE ULTRA	320 mbit/sec	merev lemez, cd rom, steamer, nagy felbontású szkenner
FIREWARE	400 mbit/sec	merev lemez, cd rom, steamer, szkenner, videoeszközök
SCSI WIDE ULTRA 2	640 mbit/sec	lásd SCSI WIDE ULTRA

273. sz. ábra
Az interfészek összehasonlító táblázata

Professzionális grafikus felhasználásra megbízhatósága, átviteli teljesítménye és saját buszrendszere miatt az SCSI csatolók ajánlhatók.

8.2. A MONITOR ÉS A MONITORVEZÉRLŐ ÉS GYORSÍTÓ KÁRTYÁK

A számítógépes grafikában az egyik legfontosabb hardver eszköz a monitor és a monitorműködést meghatározó monitorvezérlőkártya, melynek meghatározó szerepe van a 3D-s jelenetek megfelelő minőségű képi megjelenítésében.

8.2.1. A monitorok képminőségét meghatározó paraméterek

! *A monitorok által alkotott kép minőségét meghatározó legfontosabb műszaki paraméterek a következők:*

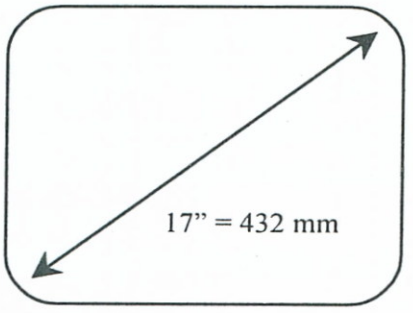
- *A monitor felbontása adja meg a raszteres képen megjeleníthető pixelek számát.* Például 800x600 felbontás esetén a kép 800 pixelsorból és 600 pixeloszlopból, azaz összesen 480.000 képpontból áll. Nyilvánvaló, hogy a grafikus kép részletgazdagsága akkor a legjobb, ha a monitor felbontása minél nagyobb. Egyszerűbb grafikai alkalmazások (például játékprogramok) esetében, ha beletörődünk a nem tökéletes képminőségbe, megfelelő lehet a 800x600-as felbontás is. *A professzionális grafika azonban ennél nagyobb felontású monitort igényel, így például*

  1024x768, 1280x1024, 1600x1200-as

  felbontású monitort választhatunk a pénztárcánk függvényében.

- A grafikus kép észlelésében fontos szerepe van annak, hogy a monitor képernyője milyen méretű (lásd 274. sz. ábra). A monitorok képátmérője általában 14", 17", 19" vagy 21" lehet. *Professzionális alkalmazások esetében minimum 17" vagy nagyobb képátmérőjű monitort kell alkalmaznunk.*

!



274. sz. ábra
A monitor képátmérője

- A képpont átmérő a képernyőn beszínezhető pixelek nagyságát adja meg. Ez az adat nem független a képernyő méretétől és a felbontástól, mint azt a 275. sz. ábra táblázata mutatja.

Képátló/ képméret (mm)	Felbontás			
	640x480	800x600	1024x768	1280x1024
13" / 264x198	0,41	0,33	0,26	–
14" / 284x213	0,44	0,35	0,28	–
15" / 305x228	0,49	0,39	0,30	0,24
17" / 345x259	0,53	0,43	0,34	0,26
19" / 386x290	0,61	0,48	0,38	0,30
21" / 426x320	0,67	0,53	0,42	0,33

275. sz. ábra
A képméret, a felbontás és a képpont-átmérő összefüggése

- *A videosávszélesség* az elektronika (például az elektronágyúk) állapotváltozásainak maximális számát adja meg másodpercenként, és így meghatározza a másodpercenként kirajzolható pixelek számát is. Lényeges, hogy ez *megfeleljen a RAMDAC teljesítményének.* !
- Az előzővel összefüggő adat a *képfrissítési frekvencia*, mely a másodpercenként kirajzolt teljes képernyők számát adja meg. Ha ez túl alacsony, akkor a kép „villog" és nagyon fárasztja a szemet. Ezért, ha huzamosabb ideig dolgozunk egy grafikus rendszerrel, *olyan monitort kell választanunk, melynek képfrissítési frekvenciája 75 Hz-nél nagyobb.* !
- A monitorok váltott soros (interlace) vagy folyamatos (non-interlace) képkirajzolással működhetnek. Az interlace üzemmód esetében a monitorok úgy működnek, mint a TV; az egyik frissítési lépésnél a pixelsorok közül csak az 1., 3., 5. ... stb. sort rajzolják ki, a következő ütemben pedig a 2., 4., 6. ... stb. sort. *Nyilvánvaló, hogy korszerűbb a non-interlaced üzemmód és a számítógépes grafikában ilyen működésű monitort célszerű választanunk.* !

Jelenleg a katódsugárcsöves (CRT) monitorokat alkalmazzák általában, ezek súlya elérheti a 25 kg-ot is. Ezért várhatóan nagy jövőjük van a folyadékkristályos (LCD) kijelzőknek, melyek tömege kb. ötödrésze, áramfelvételük pedig a fele a ma használatos monitorokénak és képminőségük is jobb. Ezek elterjedésében igazi áttörés akkor várható, ha a grafikus kártya és az LCD kijelző gyártóknak sikerül az interfészt szabványosítani. (Előrejelzések szerint ez 2000-ig megtörténik.)

A monitorok megválasztásában legfontosabb tényező a grafikus kártya és a monitor műszaki paramétereinek összhangja. Például a kártya a képmemóriájának mérete határozza meg a kép logikai felbontását, melynek összhangban kell lennie a monitor fizikai felbontásával. Ezen túlmenően a grafikus kártya digitális-analóg konvertere, a RAMDAC teljesítményének és a monitor képfrissítési frekvenciájának és felbontásának is meg kell felelnie egymásnak.

8.2.2. Monitorvezérlő és gyorsító kártyák

Az elkövetkezendőkben a monitorvezérlőkártyák és a 3D-s gyorsító chipek legfontosabb jellemzőit tekintjük át.

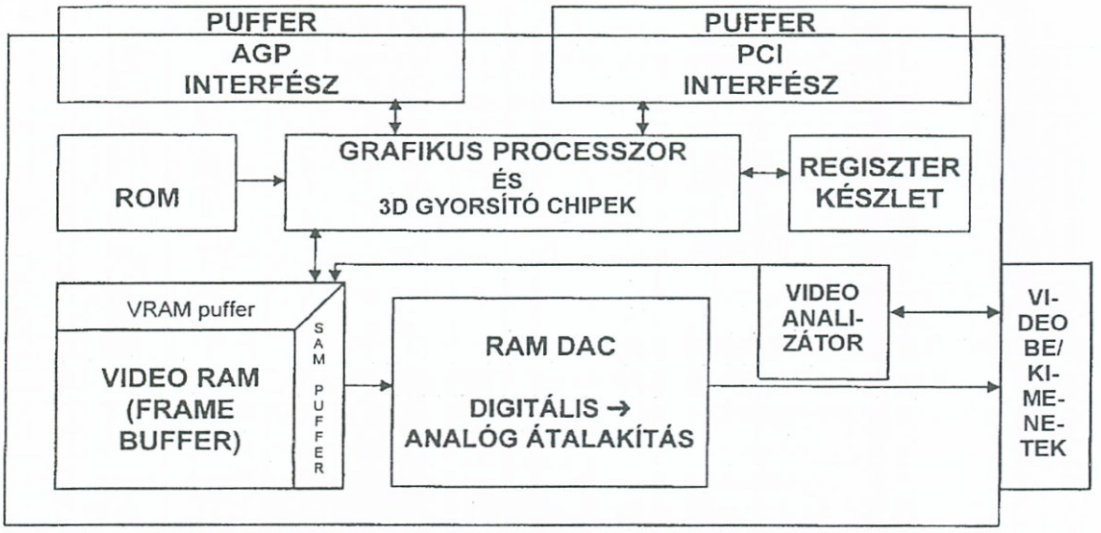
A korszerű grafikus és 3D-s gyorsító kártyákat lényegében a fotorealisztikus megjelenítés és a 3D-s virtuális valóságok (és az ennek megfelelő számítógépes játékok) igénye hozta létre.

A 2D-s és 3D-s gyorsító chipek vagy a grafikus kártyán helyezkednek el (ma már egyre inkább ez a jellemző), vagy külön kártyák, melyeket videokábellel kötünk össze a grafikus kártyával. Ezek általában önálló processzort és memóriát is tartalmaznak.

A monitorvezérlőkártyák teljesítményét frame/sec (FPS)-ben mérjük, mely a másodpercenkénti teljes raszteres képernyő megjelenítések számát jelenti. Fontos megemlíteni, hogy 25 FPS feletti sebesség kell a folyamatos mozgókép érzékeléshez. *A monitorvezérlőkártyák teljesítményében a műszaki paraméterek mellett meghatározóak a driverek* (ezek a kártyagyártó WEB-lapjáról általában legaktuálisabb változatban letölthetők). *A 3D-s gyorsítók teljesítményét a másodpercenként kirajzolt háromszögek számával szokták jellemezni.*

8.2.2.1. A monitorvezérlő és gyorsító kártyák felépítése

A grafikus és 3D-s gyorsító kártyák felépítését és működését mutatja be leegyszerűsítve a 276. sz. ábra.

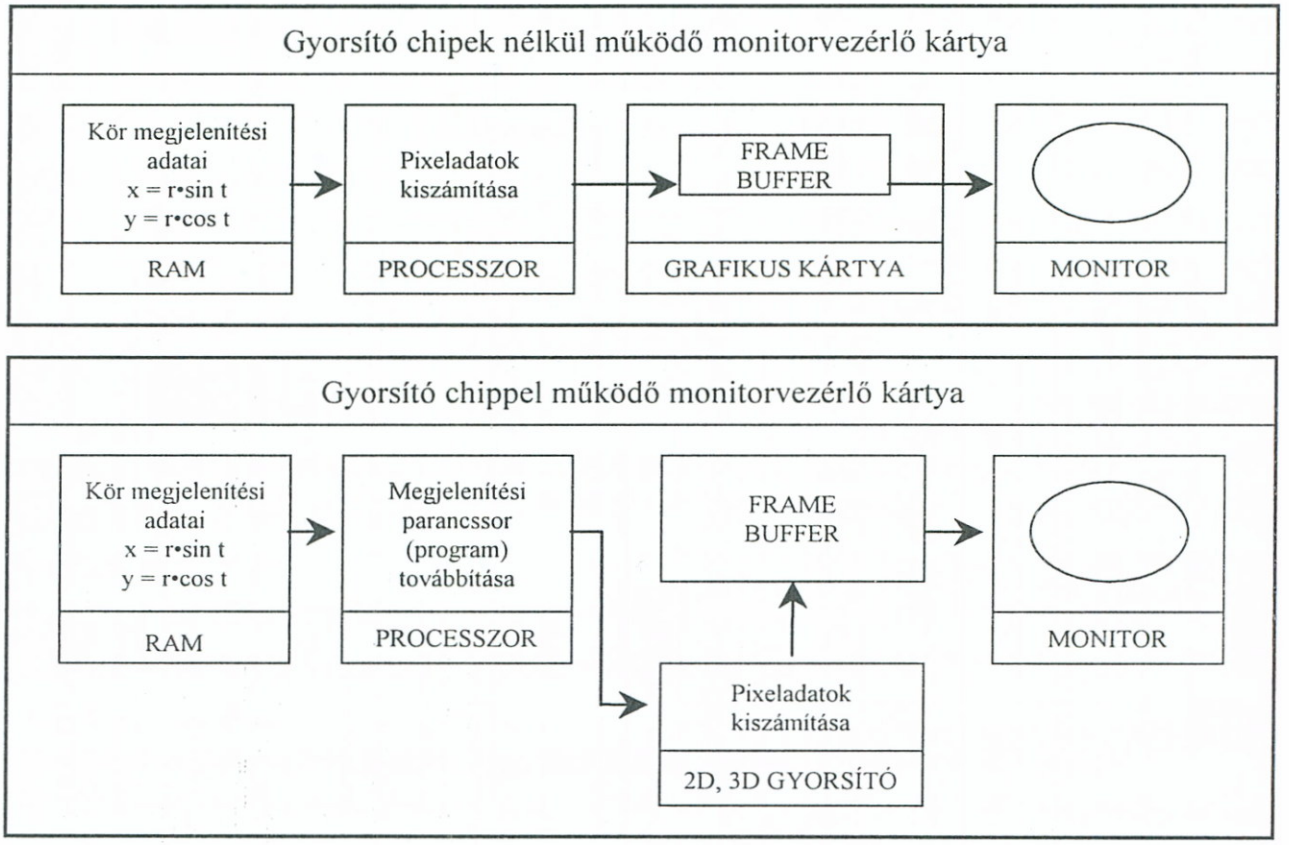


276. sz. ábra
A monitorvezérlő kártyák felépítése

A legfontosabb – eddig még nem tárgyalt – részegységek feladata a következő:

- *A pufferek* (AGP, PCI) *biztosítják a kártya kapcsolatát az AGP és PCI sínekkel.* !
- *A video ROM általában a kártyához tartozó video ROM BIOS-t tartalmazza* (karakteres képernyőkezelés esetén itt található meg a karaktergenerátor).
- *A grafikus processzorok vezérlik a frame buffer feltöltését, ezek a chipek hardver alapon egyre több grafikus algoritmust is realizálnak.* !
- *A RAMDAC a képernyőn történő megjelenítéshez szükséges digitális-analóg konverziót végzi, azaz a frame-bufferben tárolt digitális pixelinformációkat átalakítja az elektronágyúkat vezérlő analóg jelekké.* !
- *A kártya videoanalizáló egységének* feladata a RAMDAC-éval fordított: *a videokábelen érkező analóg jeleket alakítja át digitálissá.* !

A 2D-s és 3D-s gyorsító chipek a processzor tehermentesítése érdekében alakultak ki. *Ezek feladatát egy példán keresztül mutatjuk be* (lásd 277. sz. ábra). !



277. sz. ábra
A grafikus gyorsító chipek működésének elve

8.2.2.2. A korszerű monitorvezérlő és gyorsító kártyák tulajdonságai

Napjainkban a grafikus gyorsító chipek egyre több feladatot átvesznek a processzortól. Ezt mutatja be a következő példa, melyben egy konkrét, korszerű grafikus kártya jellemzőit soroljuk fel:

Funkcionális tulajdonságok hardverben megvalósítva

- Gouraud árnyékolás
- z-buffer
- Alpha-blending
- Nagyteljesítményű anti-aliasing
- Texture mapping
  - bilinear és trilinear filtering
  - perspektíva korrekció
  - mip-mapping
- Köd kezelése

Funkcionális tulajdonságok (szoftver támogatás)
- Open GL
- Direct X
- Windows NT

Memória a kártyán
20 Mbyte, mely 32 Mbyte-ig bővíthető. Ezen belül:
- 15 Mbyte 3D RAM, 96 bit/pixel 1280x1024 felbontásnál
- 4–16 Mbyte textúra tároló (8, 16, 32 bit textúrák)
- 1 Mbyte RAM a frame buffernak

Színkezelés
640x480-tól 1280x1024-ig 24 bites (16,7 millió színárnyalat)

Grafikus teljesítmény
- 2 millió 3D-s háromszög kirajzolás másodpercenként
- Textúrák:
  - 60 millió pixel/sec bilineár filteringnél
  - 30 millió pixel/sec trilinear filteringnél

Egy korszerű grafikus kártyát napjainkban a következő tulajdonságok jellemzik:

- *Multitextúrakezelés* (250 millió pixel/sec)
- *Bump-mapping* (125 millió pixel/sec)
- *Anisotrop filtering* (125 millió pixel/sec)
- *z-buffer* (32 bit)
- *Max. textúra nagyság (2048x2048)*
- *Max. felbontás (2048x1536)*
- *Busz csatlakozás (PCI, AGP 4X)*
- *Max. memória kiépítés (32 Mbyte)*
- *Gyártástechnológia (0.25 μ)*
- *RAMDAC teljesítmény (300 MHz)*
- *API támogatás* (Open GL, Direct 3D)
- *Operációsrendszer támogatás* (WIN95, 98, NT, LINUX)
- Egyebek, például: enviroment mapping textúra tömörítés, dualis pipeline stb.

A fentiekben bemutatott műszaki paraméterek – főként a teljesítmény-jellemzők – a felgyorsult fejlődés miatt szinte hónapról hónapra változnak. Felhasználói szempontból a megadott értékek napjainkban viszonylag bonyolult jeleneteket tartalmazó grafikus animációk élethű, filmszerű megjelenítését biztosítják.

8.3. GRAFIKUS PERIFÉRIÁK

A számítógépes grafikában a jól ismert input/output eszközök döntő részének van valamilyen szerepe. Az elkövetkezendőkben a perifériákat grafikus rendszerekben való felhasználásuk szerint tekintjük át és nem foglalkozunk az egyes eszközök működésének részleteivel.

8.3.1. Adatbeviteli (input) eszközök

A számítógépes grafikában az ember–gép kapcsolat helyzetinformációkat közlő input eszközei két alapvető típusba sorolhatók:

- *A relatív koordinátás eszközökhöz tartozik*
  - *az egér, a hanyattegér (track ball),*
  - *a joystick,*
  - *a touchpad,*
  - *a fényceruza.*

  Ezek relatív helyzete határozza meg a grafikus rendszerrel közölt koordinátaértékeket.
- *Abszolút koordinátás eszközökkel, a digitális táblával (tablett) és a klaviatúrával közvetlenül tudunk koordinátaértékeket megadni a vektorgrafikus rendszernek.*

A relatív koordinátás eszközöket és a tablettet, mivel ezekkel pozicionális adatokat közlünk a grafikus rendszerrel, lokátoroknak is nevezik.

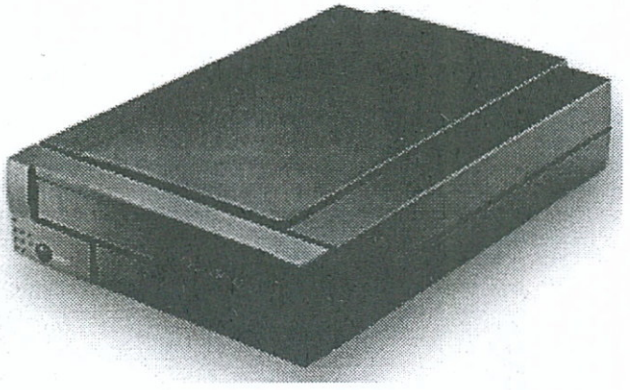
Képbeviteli eszköz lehet még egy grafikus rendszerben

- *a szkenner,*
- *a CD-olvasó,*
- *a digitális kamera és a digitális fényképezőgép,*
- *a videobemenet,*
- *a hálózat (például csoportmunka esetén egy LAN, vagy az INTERNET).*

Az elkövetkezendőkben a szkennert, a digitalizáló táblát és a digitális kamerát és fényképezőgépet mutatjuk be részletesebben.

8.3.1.1. A szkenner

A számítógépes grafikában szkenner segítségével fekete-fehér vagy színes képeket olvashatunk be raszteres formában (lásd 278. sz. ábra).



278. sz. ábra
A szkenner

A beolvasott kép minősége szempontjából a szkenner legfontosabb jellemzője a felbontás, mely az eszköz által érzékelt képpontok számát jelenti inchenként (dpi = dot/inch) és meghatározza a beolvasott kép részletdússágát.

A szkennerek felbontóképességénél meg kell különböztetni a fizikai vagy optikai felbontást, mely az olvasófej műszaki teljesítményét jellemzi és a megnövelt logikai felbontást, melybe a beolvasott képpontok közé interpolációval beszúrt, programmal generált képpontokat is beszámítják (a logikai felbontás a többszöröse is lehet a fizikainak).

A szkennerek alapvetően két teljesítménykategóriában szerezhetők be.

A kisteljesítményű szkennerek

Ezek könnyen hordozható berendezések, mechanikájuk hasonlít a nyomtatóéhoz: a papírt görgőkkel, motorosan továbbítják. Használhatóságukat korlátozza, hogy csak önálló lapokat lehet velük beolvastatni. Fizikai felbontásuk 400 dpi körüli, a soros vagy párhuzamos porthoz csatlakoztathatók.

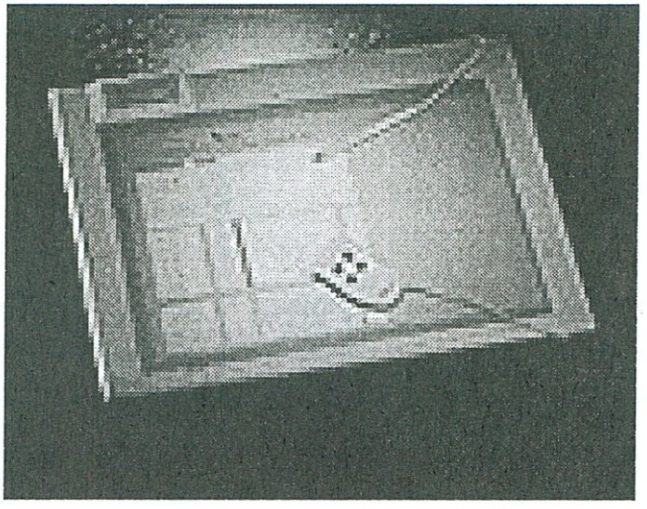
A nagyteljesítményű szkennerek

Ezek a síkágyas szkennerek általában komoly méretűek, felhasználási területük általában a kiadványszerkesztés.

Használatukkal 24 bit színmélységű is lehet a beolvasott kép, fizikai felbontásuk 1200–2400 dpi között helyezkedik el. Előnyük, hogy segítségükkel nemcsak A4-es lapokat, hanem például kihajtott könyvlapokat is be lehet olvastatni. Ezek a szkennerek célszerűen a SCSI csatolón keresztül kapcsolódhatnak a számítógéphez.

8.3.1.2. Digitalizáló táblák

A digitalizáló táblák vagy tablettek segítségével például térképekről, tervrajzokról vihetünk be koordinátaadatokat a grafikus rendszerbe (lásd 279. sz. ábra).



279. sz. ábra
Digitalizáló tábla

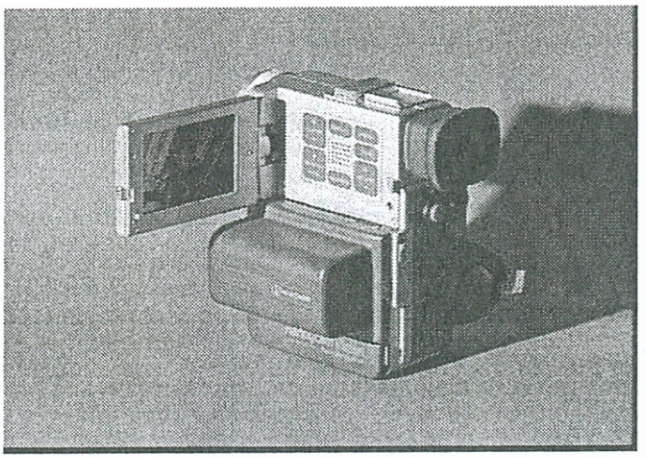
! *A téglalap alakú munkaterület mérete A0-tól A3-ig változhat. Ezen egy szélkereszttel ellátott leolvasót mozgathatunk, melynek helyzete közvetlenül, mint koordinátaérték kerül értelmezésre a grafikus rendszerben. Így tudunk vektoros információkat, például poligon csúcspontokat közölni egy grafikus programcsomaggal.*

A tablettek kiválasztása során a következő paraméterek értékelése fontos:

- érzékenység,
- a munkaterület nagysága,
- a kezelőszoftver funkcionális lehetőségei.

8.3.1.3. Digitális kamera és fényképezőgép

! *A digitális kamerával és fényképezőgéppel a környezetünkben lévő objektumokról közvetlenül előállíthatunk digitalizált raszteres képet* (lásd 280. sz. ábra).



280. sz. ábra
Digitális kamera és fényképezőgép

A digitális kamerákat és fényképezőgépeket sokszor egybeépítve gyártják, de kapható külön digitális fényképezőgép is. Jellemző műszaki paramétereik:

- Felbontás: ez általában a képmérettől is függ, például:
  - nagyméretű kép 1536x1024 képpont
  - kisméretű kép 768x512 képpont
- CCD képpontok száma, például 1,68 millió
- CCD érzékenység, például ISO 100, 200, 400
- Memóriaméret, például 15 Mbyte
- Objektív fókusztávolsága, például 2,0–2,4
- A CCD-ről kapott képadatok feldolgozásának módja: hardver vagy szoftver
- A kamerához tartozó LCD monitor jellemzői

Az előzőekben bemutatott eszközök (szkenner, digitalizáló tábla, digitális kamera) közös jellemzője, hogy felhasználásuk hatékonyságát a hardver mellett az eszközökhöz biztosított szoftver is meghatározó módon befolyásolja.

Ezek közül legfontosabbak az eszközmeghajtók, a különböző képkorrekciós, kalibrációs programok, valamint a professzionális grafikai programcsomagokhoz való adatátadáshoz szükséges plug-in-ek.

8.3.2. Az ember–gép kapcsolat output eszközei

A számítógépes grafikában output eszközként a következőket kell figyelembe vennünk:

- *a monitorok*
- *a nyomtatók, melyek típusai*

- *a tintasugaras nyomtató*
- *a laser nyomtató*
- *a hőnyomtató*

- *a nyomdai levilágítók*
- *a plotterek vagy vektorgrafikus rajzgépek.* Típusaik a következők:
  - *Tollplotterek* (sikplotter, dobplotter)
  - *Elektrosztatikus plotterek*
- *a hálózat.*

Ezek közül röviden a professzionális nyomtatókat és a plottereket mutatjuk be.

8.3.2.1. Professzionális nyomtatók

Ezekkel a nyomtatókkal grafikus rendszerekben (például CAD vagy GIS képek készítése esetén) *fényképminőségű, szükség esetén poszter méretű nyomtatást is végre tudunk hajtani, egyenletes finom tónusokkal, pontos színábrázolással.* Fontos követelmény, hogy a nyomtatáshoz többfajta papír is felhasználható legyen, a festékanyag fényálló és tartós legyen. *Ezek az eszközök általában feltételezik a Postcript nyelv használatát* (például Adobe PostScript Level 3) *és beépítve tartalmaznak egy RIP* (Raster Image Processor) *processzort.* A nyomtatandó képelőállításhoz az eszközöknek megfelelő tárolókapacitással is rendelkezniük kell (például beépített 2 Gbyte-os merevlemez).

A nyomtatás felbontása fényképminőségnél a 600 dpi-nél kezdődhet, az ún. *nyomdai levilágításra alkalmazott eszközöknél a szokásos szabvány 1270 dpi, de ez elérheti akár a 2540 dpi-t* is. A nyomdai levilágítókkal az úgynevezett nyomási minták (speciálisan bevont fémlemezek) előállítását lehet jelentősen megkönnyíteni.

8.3.2.2. Rajzgépek (plotterek)

A plotterek, vagy rajzgépeknek többfajta típusa létezik.

- *A tollplotterek lehetnek:*
  - *síkágyasak (flat bed plotter), ekkor a toll X és Y irányban egyaránt mozog,*
  - *a dobplotterek esetében a toll egyik koordináta irányban (oszlop), a papír pedig a másik koordináta irányában* (sor) *mozog,*
- *Az elektrosztatikus plotterek a negatív feltöltésű papírra pozitív töltésű tintasugarakat fecskendezve állítják elő a képet.*

Egy korszerű plotter (lásd 281. sz. ábra) alkalmas

- A0, A1 méretű rajzok elkészítésére,
- legalább 40 Mbyte memóriával és postscript áramkörökkel rendelkezik,
- 600 dpi-nál nagyobb felbontást képes kezelni,
- működhet soros, párhuzamos portról és hálózati interfészről is.

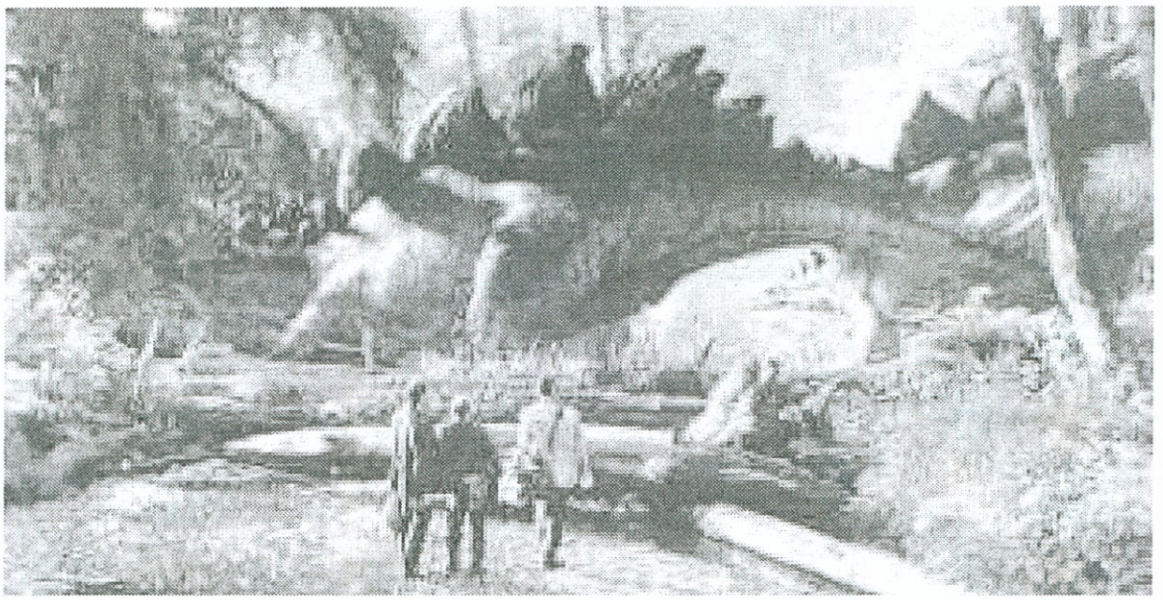
281. sz. ábra
Korszerű színes plotter

8.4. A PROFESSZIONÁLIS GRAFIKUS PROGRAMCSOMAGOK

A professzionális grafikus programcsomagok legjellemzőbb felhasználási területei a következők:

- *A számítógéppel segített tervezés (CAD), melynek legfontosabb részterületei a géptervezés és az építészet. Ilyen szolgáltatásokat nyújtó programcsomagok, például az AutoCAD, a CADKEY és az ARCHIECAD.*
- *A különböző grafikák, például reklám célra történő számítógéppel történő előállítása.* Ezek lehetnek különböző *termékismertetési,* vagy *oktatási célú prezentációk, de lehetnek képzőművészek által készített képek* is. A vektoros rajzolóprogramok közül a legnagyobb múlttal rendelkezik és a legismertebb a *COREL DRAW.*

- *A nyomdai kiadványszerkesztés* ma már elképzelhetetlen a számítógépes grafika felhasználása nélkül. E területen *jól ismert és alkalmazott programok az ADOBE PHOTOPSHOP és a PAGEMAKER.*
- *A szórakoztató és a filmipar is egyre inkább hasznosítja a számítógépes grafikát* (lásd 282. sz. ábra). *Ezekkel például elképzelt, sci-fi világokat lehet alkotni, vagy 3D-s virtuális "szörnyeket" lehet készíteni. E területen ismert programok közül a 3D Studio MAX, a SOFTIMAGE 3D és TRUESPACE említhető meg.*



282. sz. ábra
Számítógépes grafikával generált jelenet a Jurassic Park című filmből

A grafikus programcsomag gyártók közötti piaci verseny az elmúlt években rendkívül kiéleződött. Ennek egyik következménye, hogy a szoftvergyártó cégek a grafikus programcsomagok szolgáltatásaival "minden" igényt ki akarnak elégíteni.

Ennek érzékeltetésére tekintsük át például a COREL cég által készített programcsomag – mely eredetileg csak rajzolóprogram volt – fontosabb elemeit:

- COREL DRAW = vektoros rajzoló programcsomag,
- COREL PHOTO-PAINT = raszteres szkennelő, retusáló programcsomag,
- COREL VENTURA = nyomdai kiadványszerkesztő programcsomag,
- COREL CHART = üzleti grafikakészítő programcsomag,
- COREL MOVE = animációkészítő programcsomag,
- COREL MULTIMÉDIA MANAGER = multimédia készítő programcsomag,
- COREL MOSAIC = vizuális file kezelő programcsomag,

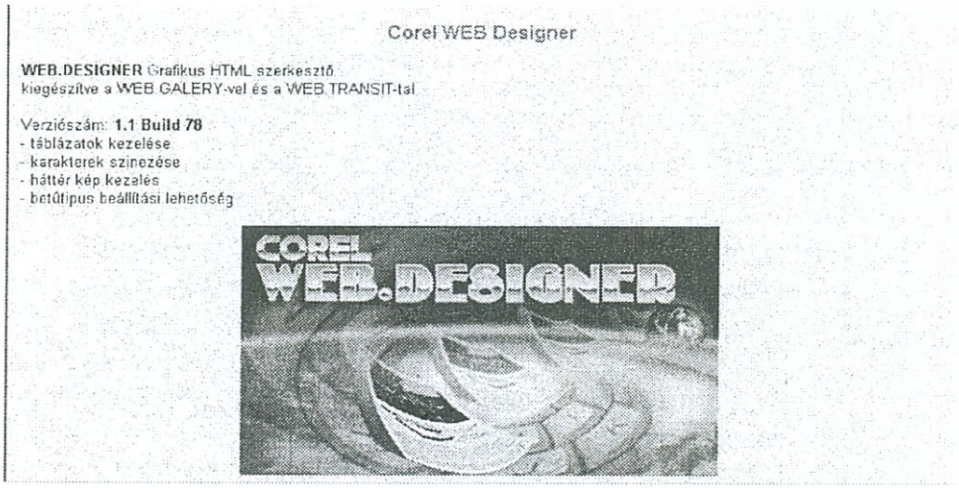
- COREL TRACE = vektor-raszter konverziót végző és OCR karakterfelismerő programcsomag,
- COREL PHOTO CD = CD készítő programcsomag,
- COREL GALLERY = képi adattár.

Fontos megjegyezni, hogy ez a felsorolás nem teljes, és azt is, hogy a programcsomagok „duzzadása" és erőforrásigénynövekedése nemcsak a COREL-re jellemző.

Mindezek eredményeképpen a programcsomagok igen nagyra nőttek és hardver erőforrás igényük is egyre nagyobb. Például a COREL DRAW 8 verziójának futtatásához – felhasználóbarát működtetés esetén – 64 Mbyte RAM szükséges.

A gyakorlati szakembereknek például az építész tervezőnek, vagy a reklámgrafikusnak – akik a programcsomagok szolgáltatásainak csak egy töredékét használják – a véleménye erről a folyamatról meglehetősen negatív.

Ezt a tendenciát illusztrálja az is, hogy a COREL már grafikus HTML szerkesztő programcsomagot (lásd 283. sz. ábra) vagy tervező programot (CORELCADD) is ajánl.



283. sz. ábra
A COREL cég WEB oldalszerkesztője

Az elkövetkezőkben a professzionális grafikus programcsomagok közül egy-egy elterjedten használatos és a szakmai közvélemény szerint az adott jellemző felhasználási területen vezető programcsomagot fogunk bemutatni. Ezek a következők:

- 3D Studio MAX vektoros animációkészítő, fotorealisztikus képmegjelenítő speciális effektusokkal,
- AutoCAD vektoros mérnöki tervezőprogram és kiegészítései,
- CorelDraw vektoros rajzolóprogram, melyet például a reklámgrafika készítésben alkalmaznak széleskörűen,
- Adobe Photoshop, mely raszteres professzionális képszerkesztő programként a kiadványszerkesztésben a legelterjedtebb.

8.4.1. A 3D Studio MAX

E programcsomagot leginkább az animációs film és videokészítésben hasznosíthatjuk, így például nagyon hatékonyan alkalmazható oktatófilmek készítésére.

A programcsomag az ún. parametrikus, matematikai formulákkal való modellezést ötvözi a poligoniálissal. Ez azt jelenti például, hogy a felületek és térgörbék előállítása a spline-okkal kapcsolatban megismert matematikai modellek (lásd III. fejezet) alapján történik, de a megjelenítési algoritmusok a testek poligonális közelítését alkalmazzák a renderelésnél. A program képszámítási algoritmusai lehetővé teszik ún. szerkesztő nézetek készítését, amivel még a kép tervezése során folyamatosan ellenőrizhető a képernyőn a kép kialakulása. A fényforrások hatását, a fényviszonyokat szintén interaktív módon követhetjük.

A programcsomag fejlesztő készlete és dokumentációja publikus, ami jelentősen hozzájárult a plug-in modulok számának növekedéséhez és lehetővé teszi megfelelő „C" programnyelvi ismeretek birtokában saját kiegészítő modulok kifejlesztését is a 3D Studio MAX-hoz.

A programcsomag lehetőségeit a következő példák érzékeltetik:

- lehetőség van az objektumokkal CSG műveleteket végezni;
- az animációkészítésnél a morphing-ot is alkalmazhatunk;
- a NURBS alapú görbe és felületmodellezés lehetővé teszi a görbült felületű objektumok nézetfüggő, jó minőségű megjelenítését. Ennek során a felhasználó a felületek kontrollpontjait közvetlenül is módosíthatja,
- a raytracing a modelltérben szelektíven alkalmazható, azaz ez leszűkíthető például a legfontosabb elemekre. A sugárkövetéssel számolt visszaverődések és fénytörések a felhasználó által befolyásolhatóak. Így például beállítható a visszaverődések száma, az elmosódás, az élsimítás stb. Megfelelő plug-in modullal a programcsomagban a radiosity eljárást is alkalmazhatjuk;
- az animációkban az objektumok a fizika törvényeinek (gravitáció) megfelelően mozoghatnak, ütközhetnek (DINAMIX). Lehetőség van olyan részecskeeffektusok alkalmazására, mint a robbantás, széthullás, szikrázás, folyadékmozgás;
- a programcsomag számtalan fényeffektust képes kezelni. Néhány példa: lencsecsillogás, füst, csillagköd, tűz stb.;
- a karakter animációt a programcsomag széleskörűen támogatja;
- a felületek reális anyagi mintáknak megfelelő textúrázására (fa, márvány stb.) lehetőség van. Már olyan kiegészítő modul is létezik a 3D Studio MAX-hoz, amivel szőrös felületű állatfigurákat is előállíthatunk. (Ez legtöbb grafikus rendszernek igen komoly feladatot jelent.)

Érdemes megemlíteni a 3D Studio MAX-ban plug-inként alkalmazható újabb komplex renderelési eljárást, az ún. RadioRay-t, mely mint a nevéből is kitűnik, a raytracing és radiosity algoritmusok ötvözésével keletkezett.

Ez megőrizte a radiosity eljárásnak azt az előnyét, hogy a modelltérben kiszámított fényintenzitásokat elmentjük adatbázisba és különböző nézőpontból való újabb és újabb renderelésnél ismételten felhasználjuk. (Ez rendkívül felgyorsítja az olyan realisztikus filmek készítését, melynél például azt kell visszaadni filmkockánként, hogy egy ablakokon keresztül diffúz fénnyel megvilágított szobát folyamatosan bejárunk. Ekkor minden egyes filmkocka előállítása egy-egy újabb fotorealisztikus renderelést kíván meg.) Ugyanakkor a RadioRay eljárásnál a raytracing algoritmust alkalmazzák minden olyan esetben, amikor tükröző, fényáteresztő és -törő objektumokat kell megjeleníteni.

Az OpenGL és Direct 3D API-kat a programcsomag támogatja, az ADOBE PHOTOSHOP-hoz plug-in filter rendelkezésre áll. !

A 3D Studio MAX programcsomag RadioRay plug-injével előállítható kép minőségét a 284. sz. ábra mutatja be.

284. sz. ábra
A 3D Studio MAX programcsomaggal előállított
RadioRay eljárással renderelt kép

8.4.2. Az AutoCAD és kiegészítő programcsomagjai

Az AutoCAD áll az első helyen a CAD tervező programcsomagok piacán, részesedése különösen a PC alapú rendszereknél meghatározó. !

Az alapprogramot kezdetben kifejezetten 2D-s műszaki rajzok elkészítésére fejlesztették ki, ma már azonban teljes körűen támogatja a 3D-s modellezést és eszközöket biztosít a tervek megfelelő minőségű rendereléséhez is.

! *Fő felhasználási területe az építészeti, belső építészeti és gépészeti tervezés, de alkalmazzák például az elektronikus áramkör tervezésben vagy a térképészetben is.*

Az alapverziót a különböző szakmáknak megfelelő speciális CAD szoftverek egészítik ki. Ezek lehetővé teszik, hogy az általános geometriai objektumok mellett egy olyan elemkészletet is felhasználjunk, amely az adott szakma igényeihez igazodik (például építészeti 3D-s fal-, födém-, tető- és nyílászáró-elemek vektorgrafikus objektumai). *Ilyen kiegészítő tervező-*

! *rendszer a gépészetben a Mechanical Desktop, vagy az építészetben az Architectural Desktop.*

Az AutoCAD-nak, mint tervezőprogramnak tehát a felhasználó szempontjából talán az a legfontosabb tulajdonsága, hogy több szakmaspecifikus „kiegészítő" programcsomag épül rá, melyek összességében az adott szakterületen komplex szolgáltatásokat biztosítanak. Ezeket nem néhány kiegészítő modulként kell elképzelnünk, hanem önmagukban is meglehetősen bonyolult programrendszer (emiatt természetesen számukra a szükséges erőforrásokat is biztosítani kell) és objektumkönyvtár együtteseként. A szakmaspecifikus kiegészítő rendszerek lehetőségeit most csak néhány funkció felvillantásával van módunkban érzékeltetni.

☝ **Az AutoCAD Architectural Desktop építészeti programcsomag néhány lehetősége**

- Falhálózat generálása a helyiségterv kontúrjaiból.
- Ajtók és ablakok képzése automatikus falnyílás szerkesztéssel.
- Lépcsők és korlátok képzése a tervezési szabályok figyelembevételével.
- Bútorok, berendezési tárgyak behelyezése a tervbe objektumkönyvtárból.
- Kameravezérléssel, a terv alapján az épületről a bejárást szimuláló fotorealisztikus filmkészítés.

☝ **Az AutoCAD Mechanical Desktop gépészeti programcsomag néhány lehetősége**

- Csavarvonalképzés (rugók, tekercsek modellezéséhez) egyetlen paranccsal.
- Alkatrészmetszéssel az objektum felbontása két független paraméteres testmodellé.
- Műhelyrajz, darabjegyzék készítés.
- Lekerekítés, héjképzés, a modellek renderelt állapotban történő mozgatása.

A tervek vizuális bemutatása megrendelőnek

A műszaki terveket elkészíthetjük egy CAD programmal, de emellett arra is szükség lehet, hogy a megrendelőnek például a megtervezett épületet valódi fényviszonyok mellett majdani környezetével együtt realisztikus képen is bemutassuk, akár egy 3D-s virtuális valóságban bejárhassuk. A műszaki látványtervek előállításához is léteznek olyan programcsomagok, melyek szervesen illeszkednek az AutoCAD-hoz. Ilyen például a 3D Studio VÍZ (melynek funkcionalitása nevének megfelelően nagyon hasonlít a 3D Studio MAX-hoz, a fejlesztő cég is azonos), mely lehetővé teszi az AutoCAD-el elkészített tervek (DWG fájlok) gyors fotorealisztikus képi megjelenítését. *Ennek fontos tulajdonsága, hogy az AutoCAD tervekben való módosítás esetén, nagyon rövid idő alatt képen is be tudjuk mutatni a változtatás kihatásait.*

8.4.3. A CorelDraw

A Corel rajzoló, illusztrációkészítő programja már a 9-es verziónál tart (Magyarországon ezt 1999 áprilisában mutatták be). E programcsomag többek között tartalmazza:

- *CorelDRAW: vektoros illusztrációs és szerkesztő-rajzolóprogramot,*
- *Corel PHOTO-PAINT: raszteres képszerkesztő programot,*
- *Font Navigator: fontkezelő programot,*
- *Corel TRACE: vektorizáló programot,*
- *Corel TEXTURE: procedurális textúragenerátort* (pl. kő, márvány, felhő stb. szimulációjára.)

A CorelDRAW eredetileg egy 2D-s vektoros rajzolóprogram volt, mellyel viszonylag komfortos módon és gyorsan lehetett szerkeszteni szabadkézi rajzzal, illetve 2D-s primitívek és a fontkészlet felhasználásával színes grafikákat, ezért különösen a reklámgrafikusok körében volt kedvelt.

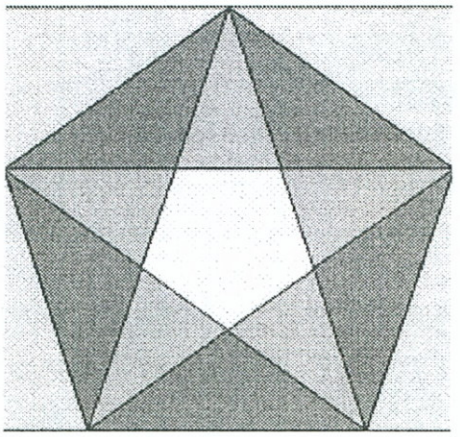
A program rajzolási képességeit néhány példával illusztráljuk:

- Bézier görbéket, poligonokat, lekerekített sarkú négyszögeket rajzolhatunk egy-két paranccsal,
- az objektumokat mozgathatjuk, tükrözhetjük, megdönthetjük, a rajzelemekből csoportokat képezhetünk, ezeket „kombinálva" egy utasítással például képkivágásokat képezhetünk,
- betűket görbévé transzformálhatjuk és ezt követően Bézier görbékként szerkeszthetjük, szövegeket görbékre illeszthetünk (lásd a 285. sz. ábra), betűtípusokat készíthetünk,
- a rajzok rétegekből (layers) épülhetnek fel, ezek együtt megjeleníthetők, ezek között az objektumok mozgathatók,

- a rajzelemeket perspektivikusan ábrázolhatjuk, két objektum egymásba történő átalakítását is megvalósíthatjuk a Blend (áttünés) paranccsal.

285. sz. ábra
Szöveg görbére illesztése
a CorelDRAW-ban

A programmal történő rajzszerkesztés hatékonyságot a 286. sz. ábra mutatja, melynek megrajzolása egy profi felhasználónak CorelDRAW-al kb. 3 percet igényel.



286. sz. ábra
Egy kb. 3 perc alatt előállítható rajz CorelDRAW-al

A programcsomag újabb verzióinak néhány lehetősége:

- *néhány speciális effektus: örvénylés (twister), áttetsző árnyékképzés (Drop Shadow),*
- *az elkészítet képek INTERNET kompatíbilis előállítása* (HTML forma, GIF, JPEG vagy Jáva program),
- *IXLA csatoló a digitális kamera képek fogadására,*
- *Batch processing lehetősége* (ez animációkészítésnél, vagy képek sorozatán végrehajtott azonos módosítás esetén lehet hasznos).

Sajnos a programcsomag konfigurációigénye az újabb verziókkal tovább nőtt. Így például:

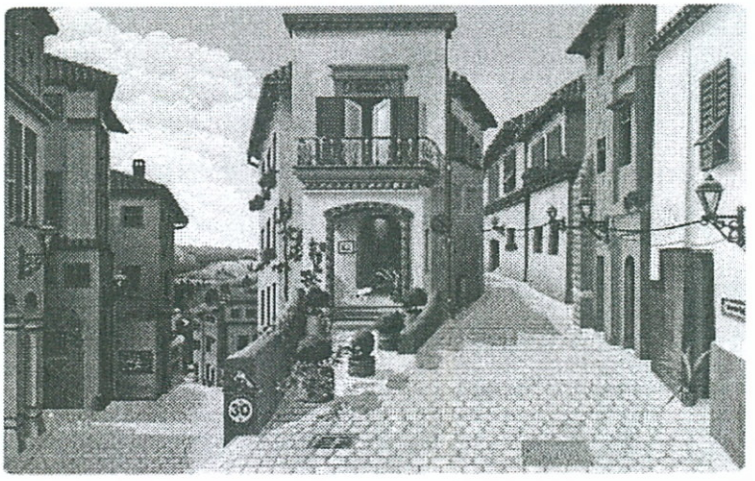
- 8 verzióban min. 16, 9 verzióban pedig min. 32 Mbyte RAM a megadott érték (a gyakorlott felhasználók viszont tudják, ezzel nem érdemes kísérletezni),

- a professzionális, megfelelő gyorsaságú napi munkavégzéshez 8 verzióban 64, a 9-es verzióban pedig inkább 128 Mbyte ajánlható.

A Corel programcsomag készítőinek egyik fontos törekvése a gyors és felhasználóbarát grafika készítés. Ezt a programcsomaggal együtt szállított objektumkönyvtárak bőséges tartalma is mutatja: !

- 25.000 *clipart és szimbólum,*
- 1000 *Type 1 és True Type font,*
- 1000 *nagyfelbontású fotó.*

A Corel programcsomaggal természetesen komplex képeket, speciális effektusokkal is előállíthatunk, igaz jóval nagyobb munkaráfordítással. Erre példát a Corel évről-évre megrendezett rajzolóversenyeinek győztes képei szolgáltatnak (lásd 287. sz. ábra).



287. sz. ábra
A CorelDRAW rajzolóverseny egyik győztes képe 1997-ben

8.4.4. Az ADOBE PHOTOSHOP

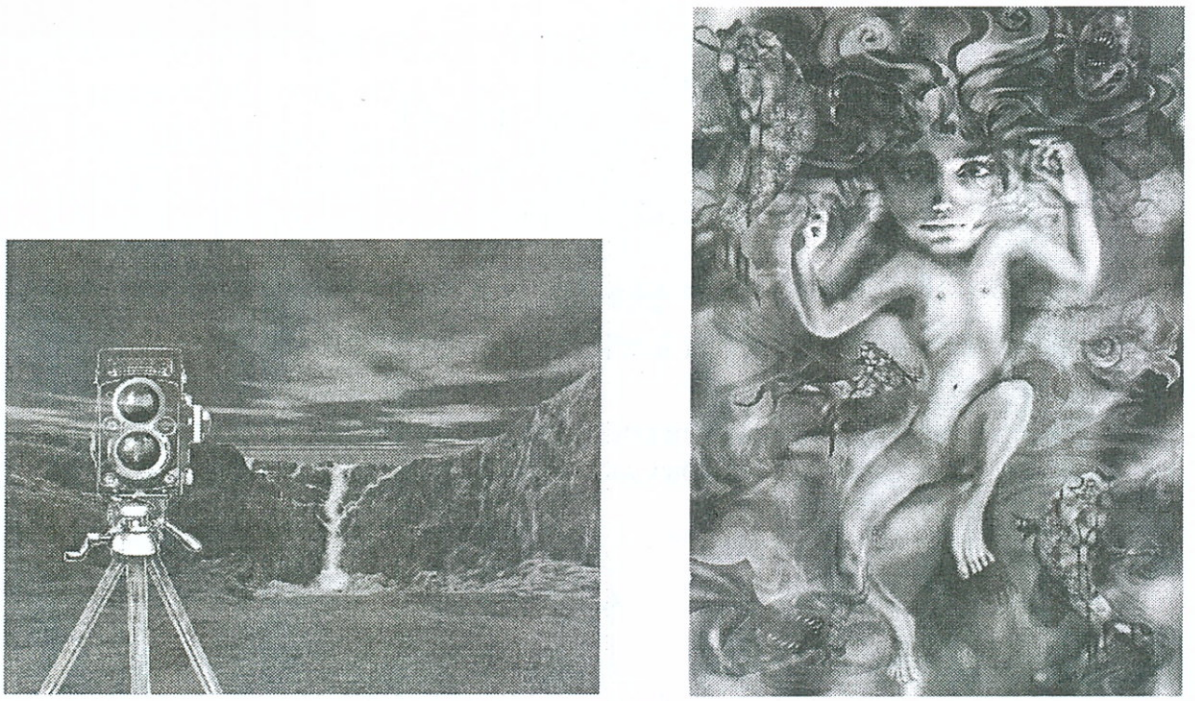
A nyomdai munkában, a kiadványszerkesztésben mindig kiemelt jelentőségű részterület volt a vizuális adatok, képek kezelése. E téren a számítógépes képfeldolgozás az elmúlt években a fototechnikának komoly konkurensévé vált

A DTP e szakterületére számos programcsomag specializálódott, de a legelterjedtebb és a szakma értékítélete szerint legsokoldalúbb mindenképpen az *ADOBE cég PHOTOSHOP nevű raszteres képfeldolgozó programja.* !

A programcsomag lehetőségeit néhány példával érzékeltetjük

- *a szokásos színterek (RGB, CMYK, HSB) mellett a PHOTOSHOP beépített színmintákat is felkínál, melyeket a nyomdai festékgyártók színkódjai alapján is megválaszthatunk;*
- *a kép jellemzőjeként 0–255 közötti értékkel a kontrasztot* (contrast) is megadhatjuk, mellyel a TV beállításához hasonlóan a kép világos és sötét részei közötti tónusátmenetet változtathatjuk meg (minél nagyobb a kontraszt értéke, az egyes képrészek határai annál élesebbek);
- a nyomdatechnikában a színes képeket a CMYK színtér komponensei szerint fel kell bontanunk (color separation). Ehhez a *PHOTOSHOP lehetőséget biztosít olyan szürkeárnyalatú képek előállítására, melyek a CMYK színtér alapszíneinek megfelelő színek árnyalatait adja vissza (színkivonatok = color plate).* Ezeket a nyomda az ún. színnyomatok készítésénél használhatja fel;
- a PHOTOSHOP *külön szolgáltatásokkal támogatja a nyomdai kép előkészítéséhez szükséges kalibrációt*;
- a raszteres képek részeit kijelölhetjük és önállóan módosíthatjuk a képrészek jellemzőit;
- a képek manipulálását *és speciális effektusok előállítását speciális eszközök az ún. filterek segítik. Így például a blur filter lágyítja a színek közti átmenetet.* Ennek mértékét beállíthatjuk és akár koncentrikus körök mentén is alkalmazhatjuk. Utóbbi esetében például a Spin (perdület) paranccsal a képen a forgó mozgást szimulálhatjuk. A *distort filterrel eltorzíthatjuk a képet például egy másik kép adatai alapján, a ripple filterrel pedig vízben tükröződő képet* állíthatunk elő;
- a *PHOTOSHOP-ban is lehetőségünk van layerek alkalmazására.* Ezekben önállóan speciális effekteket is alkalmazhatunk: például izzás, domborítás stb.

A PHOTOSHOP lehetőségeit próbálja érzékeltetni a 288. sz. ábra.



288. sz. ábra
Az ADOBE PHOTOSHOP-al készített raszteres képek

8.5. A VIRTUÁLIS VALÓSÁG MODELLEZŐ NYELVE (VRML)

A számítógépes grafikában egyre fontosabb szerepet tölt be az úgynevezett látszólagos valóság (vagy virtual reality) modellezése.

A virtuális valóság alatt olyan, a számítógépen létrehozott mesterséges háromdimenziós világot értünk, melyet a felhasználó bejárhat, felfedezhet, valóságosként érzékelhet. !

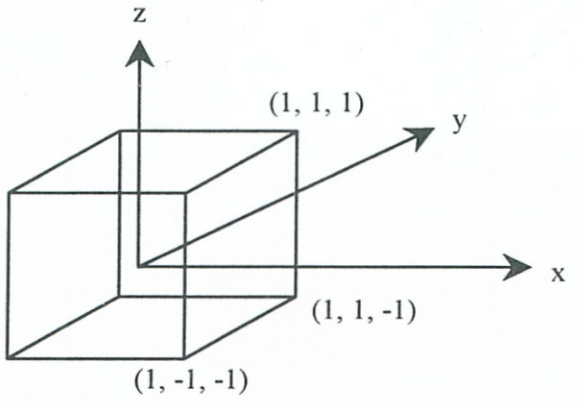
A virtuális valóság modellező nyelve a VRML (Virtual Reality Modeling Language), melynek 1.0 verziója 1995 februárjában vált szabvánnyá. E területen nagyon gyors a fejlődés, például az utóbbi évben kb. 500.000-ről közel 800.000-re nőtt a VRML-re hivatkozó WEB lapok száma, és több mint 15.000 3D-s virtuális valóság érhető el a világhálón.

A VRML 1-es verziójával alapvetően térgeometriai objektumokat definiálhatunk. Ezek lehetnek például kockák, kúpok, gömbök stb., melyeknek meg kell adni a virtuális térbeli koordinátáit és jellemzőit (szín, méret stb.). ! Ezt egy ASCII szövegfájlban tehetjük meg, például a következőképpen:

```
#VRML V1.0 ascii
Separator       { # A főblokk
  Cube { } #  Alapértelmezett kocka
} # A főblokk <<vége>>
```

289. sz. ábra
Alapértelmezett kocka definiálása VRML-ben

Példánk egy alapértelmezett kockát definiál, melynek középpontja a modelltér koordináta-rendszerének origójában van, oldala 2 egység csúcskoordinátái (1, 1, 1), (1, 1, -1) ... stb. (lásd 290. sz. ábra).



290. sz. ábra
A VRML-ben alapértelmezett kocka helyzete a koordináta-rendszerben

A VRML objektumokat definiáló fájlok kiterjesztése a konvenció szerint *.wrl.

A VRML nyelv az alakzatok színeit az RGB alapszínek keverésével állítja elő. A három szín arányát, telítettségét egy számhármassal adhatjuk meg:

```
Material {
        diffuseColor 0  1  0  # zöld
}
```

291. sz. ábra
Zöld szín definiálása VRML nyelven

A fenti tulajdonságblokkal egy zöld színt állítottunk elő. A diffuseColor arra utal, hogy a felület matt, azaz a megvilágítási modellben megjelenítés-

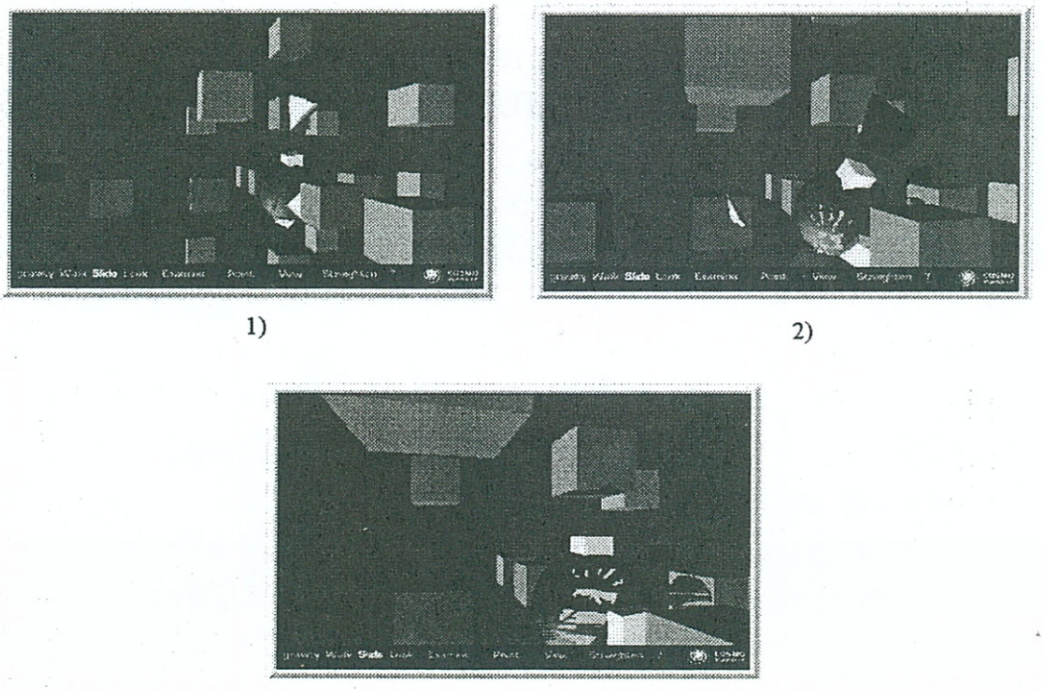
kor diffúz fényt kell alkalmazni. A 0,1,0 érték az RGB alapszínek intenzitását adja meg: vörös = 0, zöld = maximum, kék = 0.

A virtuális térben lámpákat (fényforrások) és kamerákat (3D-s nézőpont) is meghatározhatunk és lehetőség van az objektumok affin transzformálására is.

Ha a virtuális valóságban a 3D-s modelltérben már definiáltuk az objektumokat (testek, kamera, fényforrások), akkor lehetőségünk van sétát tenni a virtuális térben. *A séta (walk) a virtuális valóságban azt jelenti, hogy egy program igénybevételével – melyet lejátszó programnak nevezünk – „bejárhatjuk" a 3D-s modellteret*, melynek során a képernyőn valódi 3D-s térhatású képeket ad vissza a lejátszó program. Lehetőségünk van például az előzőekben definiált kockát „körbejárni" és különböző nézeteit megszemlélni.

Mi tehát a leglényegesebb különbség az eddig megismert 3D-s grafikus programcsomagok és a VRML fájlok lejátszása között?

A VRML fájlok lejátszásakor a virtuális térbeli helyzetünket, vagyis a nézőpont meghatározását interaktív módon, például egérrel folyamatosan vezéreljük. Ennek hatására a lejátszó program a helyzetünknek megfelelő térhatású képét a modelltér jelenetének a képernyőn folyamatosan kirajzolja (lásd 292. sz. ábra).



1) 2)

3)

292. sz. ábra
A VRML fájlok lejátszásának bemutatása képernyőn; fokozatosan az egérrel vezérelve „közelítünk" a testek felé

Figyeljük meg, hogy az ábra 1) nézetétől a 2)-n keresztül a 3) nézetig fokozatosan 3D-ben behatolunk a „kockák közé".

A VRML 1 fájlok lejátszóprogramja a LIVE 3D, mely ma már az INTERNET Browser-ekben standard módon be van építve (például NETSCAPE 3.x). Ez azt jelenti, hogy ha a WEB-en egy 3D-s honlapra tévedünk, akkor a WRL kiterjesztésű fájl a Browser automatikusan lejátsza.

A VRML1-et követően alig több mint egy éven belül, 1996-ban megjelent a *virtuális valóság modellező nyelv 2. verziója*, mely a modelltérben definiálható objektumokkal kapcsolatos *lehetőségeket* jelentősen kiszélesítette.

A lehetőségek érzékeltetésére csak néhány példát sorolunk fel:

- *a virtuális térben szenzorokat helyezhetünk el*, melyekkel meghatározott eseményeket figyelhetünk és ezek bekövetkezésekor akciókat kezdeményezhetünk;
- *a virtuális térben már valós fizikai viszonyokat is modellezni lehet, például gravitáció, repülés* stb.;
- *a VRML fáljban hangforrásokat is elhelyezhetünk, melyet a lejátszó sztereo hanghatásokkal visszaad;*
- *a virtuális térben ütközést is modellezhetünk*, át nem járható tárgyak is létezhetnek;
- *a virtuális teret a 3D-s grafikában már ismert effektusokkal is felruházhatjuk, például speciális textúrák, köd* stb.

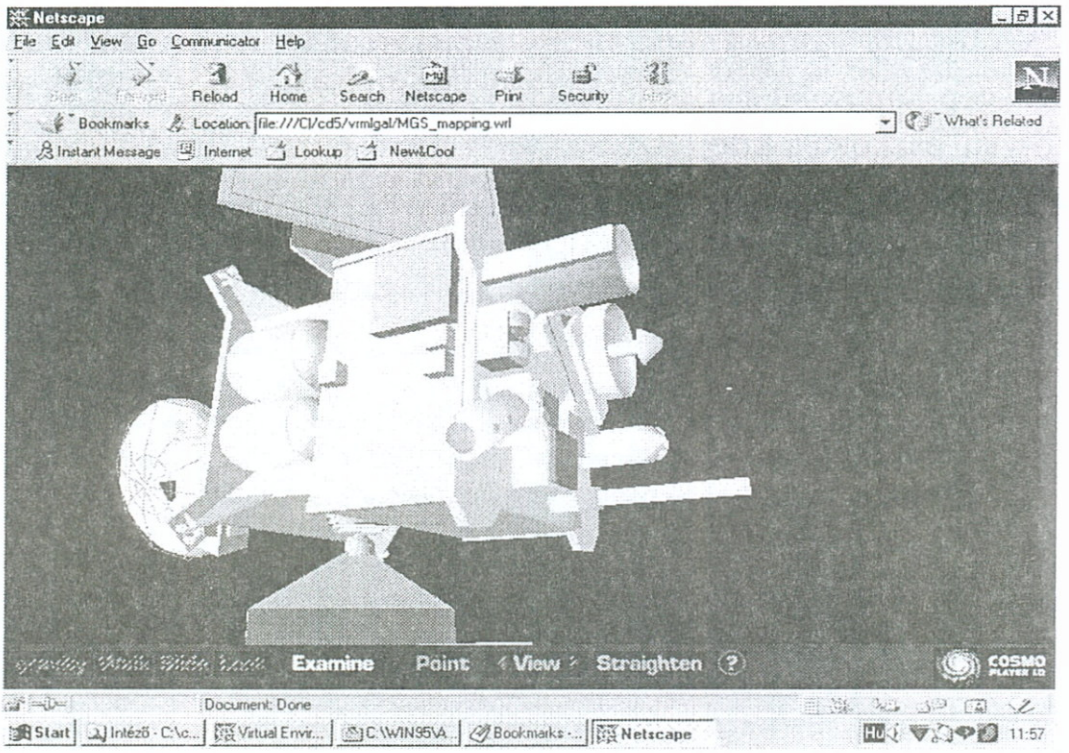
Mindezek eredményeként a VRML 2-ben definiált virtuális valóságok látványa a képernyőn ma már közelíti a 3D-s fotorealisztikus megjelenítést.

A VRML 2 megjelenésével együtt a Silicon Graphics cég az új, hatékonyabb lejátszót is kifejlesztette, a COSMO PLAYER-t, mely 1997-ben az év legjobb virtuális valóság lejátszó programjának címét is elnyerte.

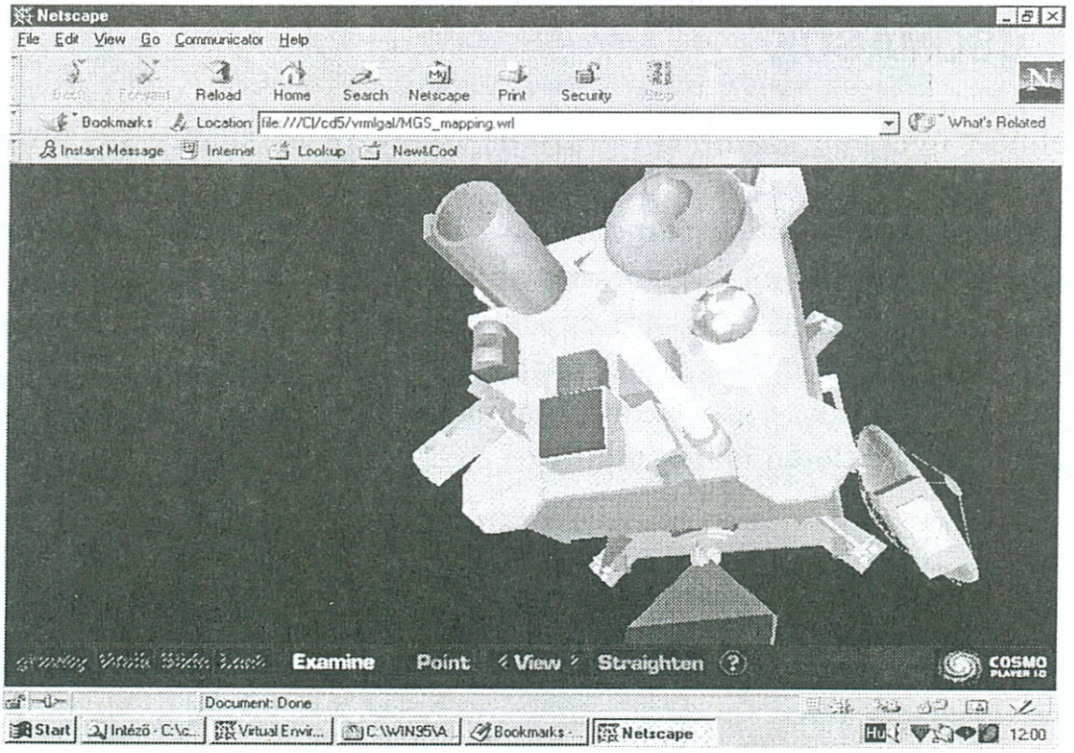
A COSMO PLAYER egy új, sokkal hatékonyabb kezelőfelületet is biztosított a 3D-s virtuális valóságok bejárásához. A COSMO PLAYER 1. verziójának kezelőfelületét mutatják be a 295., 294. sz. ábrák, melyen látható virtuális valóság egy űrszondát tartalmaz.

A VRML felhasználási területei nagyon gyorsan bővülnek:

- egyre több 3D-s honlapot találhatunk az INTERNETEN,
- a virtuális valóság modellezését játékok és filmek készítésénél is felhasználják,
- fontos szerephez jut a veszélyes helyzetek szimulációjával kapcsolatos oktatásban.



293. sz. ábra
Űrszonda VRML 1-es nézőpont



294. sz. ábra
Űrszonda VRML 2-es nézőpont

A virtuális valóság modellezés felhasználását a jövőben várhatóan az ember–gép kapcsolat perifériáinak továbbfejlesztése fogja meghatározni. A cél olyan eszközök kifejlesztése, melyekkel a virtuális valóság az ember összes érzékszervére hatni képes.

Ehhez hozzájárulhat, hogy 1997-ben az ISO (ISO/IEC 14772) a 3D-s multimédia INTERNET szabványaként elfogadta a VRML-t.

A fejlesztés e gyorsan fejlődő területen ezt követően sem állt le. 1998-ban indult be a VRML-NG (VRML Next-Generation) projekt, mely a VRML szabvány továbbfejlesztését tűzte ki célul. Ezek közül néhány szabványosítani kívánt terület

- prototípus könyvtárak,
- API,
- WEB integráció (XML, külső hozzáférések),
- renderelés szabványai,
- média integráció szabványai,
- hálózati elosztott szimuláció.

8.6. GRAFIKUS FREEWARE ÉS SHAREWARE PROGRAMCSOMAGOK

Az elmúlt években számtalan grafikus freeware és shareware programcsomag is megjelent, ezek döntő többsége a WEB-en is hozzáférhető. Ebben a fejezetben részletesebben bemutatjuk a POVRAY freeware programcsomagot, mely a raytracing algoritmussal fotorealisztikus renderelést biztosít. Röviden elemezzük a freeware és shareware 3D-s modellezőket és néhány további programcsomagot is bemutatunk felsorolásszerűen. Az e fejezetben tárgyalt programok, ezek oktatási anyagai, leírása, objektumkönyvtárai a könyv CD mellékletén is megtalálható.

8.6.1. A POVRAY (Persistence of Vision Raytracer)

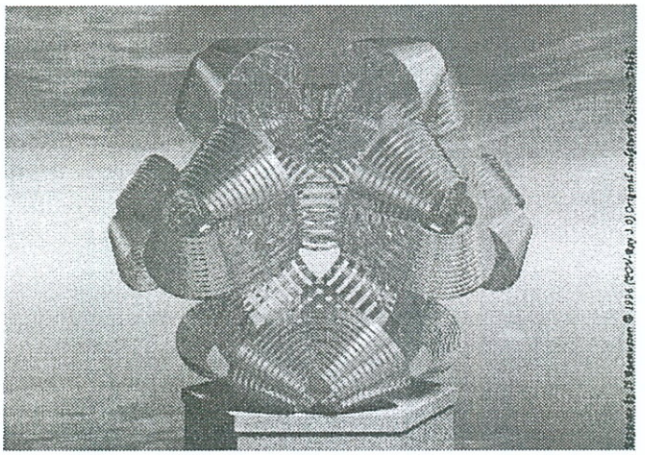
A POVRAY-t David K. Buck és Aaron A. Collins által írt DKBTrace programból kiindulva szakemberek egy csoportja szabad idejében, azt is mondhatnánk hobbyból fejlesztette ki és fejleszti azóta állandóan tovább. (A program C forrásnyelven is hozzáférhető.)

A POV-Team-ként ismert csapat tagjai világ különböző országaiban élnek és például a CompuServe POVRAY fórumán keresztül tartják egymással a kapcsolatot. Emellett még

számtalan országban léteznek POVRAY klubok a WEB-en, melyek raytracelt képeket publikálnak, textúra adatbázisokat építenek és cserélnek, segéd- és oktatóprogramokat írnak stb. Egy ilyenre példa a németországi Deutsche Povray FAQ (D.P.M.).

*A Povray egy speciális leíró nyelvet használ, ezzel kell definiálni a modelltér objektumait, a kamerát, a fényforrásokat és az effektusokat. A leíró-nyelven megfogalmazottakat *.pov kiterjesztésű fájlokban tároljuk.*

A Povray, mint raytracer az előállított képek minőségét tekintve felveszi a versenyt a professzionális 3D-s programcsomagokkal is (lásd 295. sz. ábra).

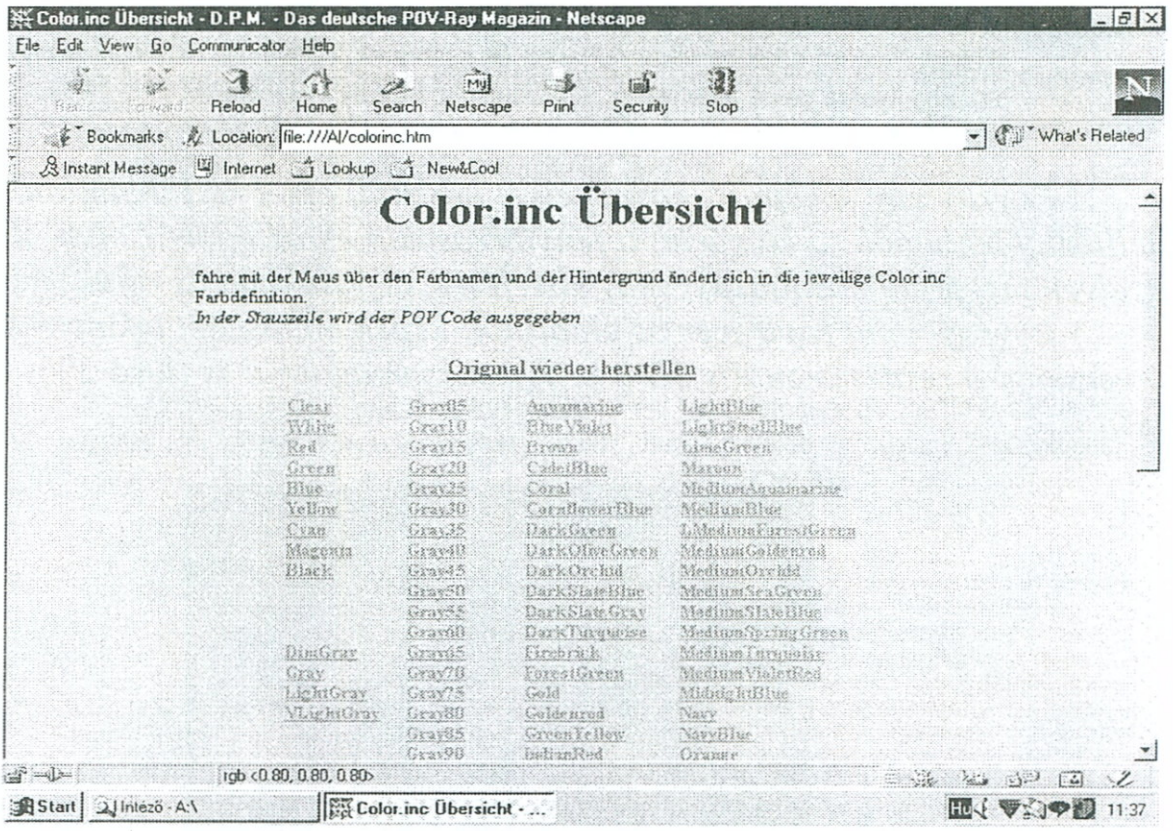


295. sz. ábra
A POVRAY programcsomaggal renderelt kép

A Povray renderelési algoritmusa a PC-s világban sokszor gyorsabb még a professzionális programcsomagokénál is. További előnye, hogy a Povray-val hobbyként foglalkozó nagyszámú szakember *igen bőséges textúra, szín és objektumkönyvtárat, standard építőelem-adattárakat is* létrehozott a programcsomaghoz (erre példát a 296. sz. ábra mutat).

A POVRAY hátránya viszont, hogy például az objektumok elhelyezkedését a térben előre meg kell adnunk a formális leírónyelven és ennek eredményét csak – a sokszor igen időigényes – *a renderelést követően szemlélhetjük meg.*

Ezért próbálkoztak meg a *Povray-hez olyan 3D-s modellezőprogramok kifejlesztésével, melyekkel interaktív módon követni tudjuk a modelltérben az objektum definíciókat és ezek kihatásait a képernyőn is megszemlélhetjük.* Ezeknek a modellezőknek a teljesítménye, komfortossága viszont meg sem közelíti a professzionális programcsomagokét (pl. 3D STUDIO MAX).



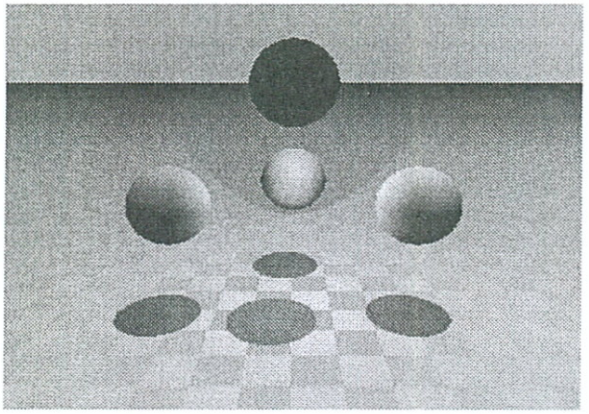
296. sz. ábra
A POVRAY-hez kifejlesztett RGB színkódoló HTM oldal
A színre klikkelve az oldal háttérszíne felveszi az adott színárnyalatot és egyúttal az RGB kódokat is leolvashatjuk az alsó sorban

A *.pov kiterjesztésű leírónyelv fájlokra nézzünk egy mintapéldát.

```
#include "colors.inc" //szindefiniálás
light_source {<0, 0, 0> color White}    //egy fényforrás a kamera előtt
light_source {<4, 50, 4> color White}  //egy fényforrás valamivel magasabban
camera { location <0, 3, -10>
          look_at < 0, 0, 0> }          //kamera irány
plane {y, 300                          //egy égbolt
       pigment {SkyBlue}               // kék
       finish {ambient 1 diffuse 0}
       }              //
sphere { <3, 0, 0>,1
         pigment {White} // Négy gömb
        }                          // különböző színnel
sphere { <-3,0,0>,1
         pigment {NewTan}
        }
sphere { <0, 3,0>,1
         pigment {Blue}
        }
sphere { <0, 0, 3>,1
         pigment {Yellow}
        }
plane { y, -3                              &nb sp;    // még egy sík
       pigment {checker Green, Pink} // „kockázva"
       }
```

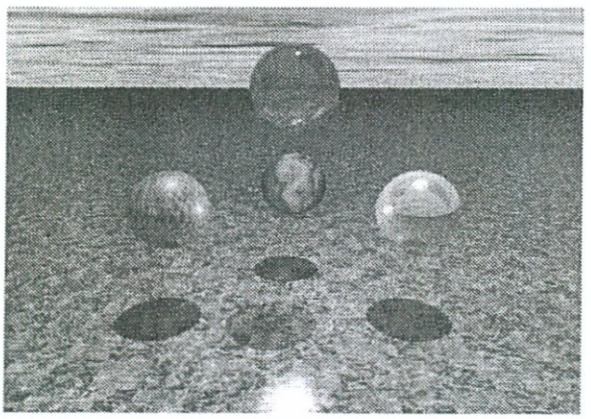
297. sz. ábra
Gömbök definiálása a POVRAY leírónyelvén

Ez a fájl POVRAY-vel renderelve a következő képet adja:



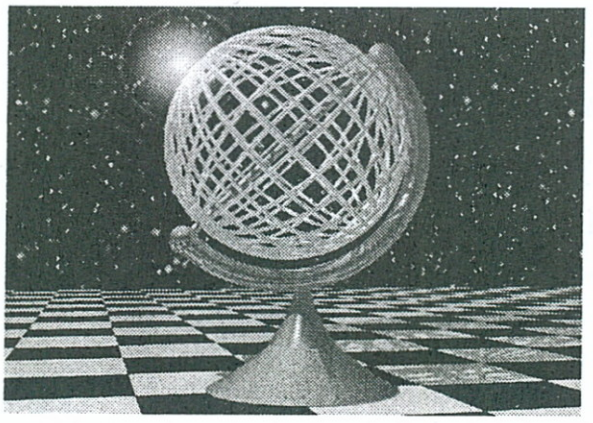
298. sz. ábra
Renderelt kép az előzőekben definiált gömbökről

Ha az előző definícióban textúrákat is alkalmazunk, az „égboltot" és a „talajt" effektekkel (pl. turbulence) feljavítjuk, a renderelt kép a következőképpen fog kinézni:

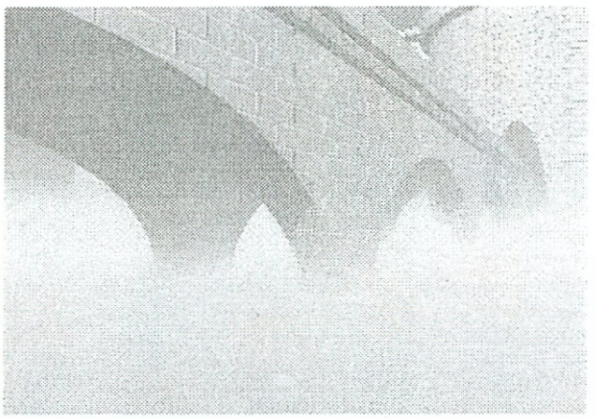


299. sz. ábra
Gömbök renderelt képe a POVRAY standard textúráival

A programcsomag további lehetőségeit néhány speciális effekttel készített képpel érzékeltetjük.



300. sz. ábra
Csillagsugárzás ábrázolása POVRAY-el



301. sz. ábra
Híd ködeffektussal

8.6.2. Modellezők a POVRAY-hez

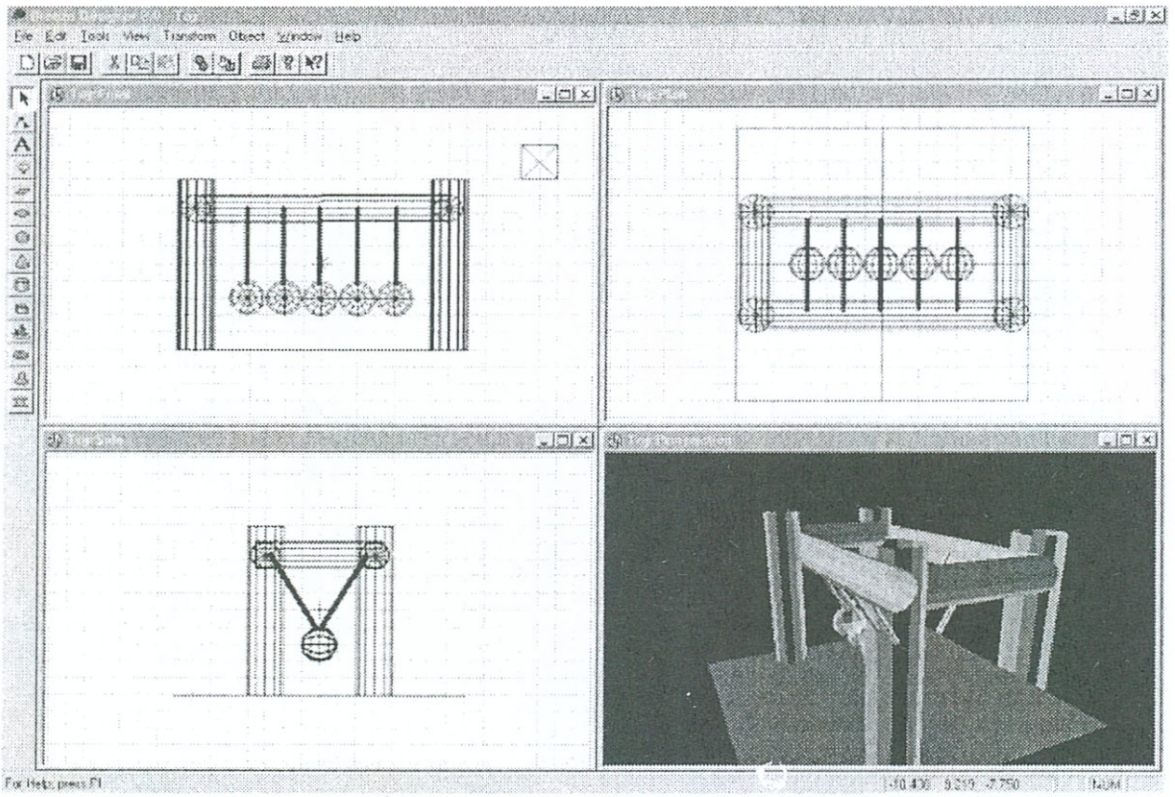
A Povray-nek nem létezik grafikus interaktív felhasználói felülete, azaz a rendelni kívánt objektumokat a *.pov fájlok formális nyelvén először definiálnunk kell és csak ezt követően futtathatjuk a Povray-t és nézhetjük meg képi formában a definíció eredményét.

! *Ezért több olyan programot készítettek, mely a felhasználó számára interaktív módon lehetővé teszi az objektumok megtervezését, a megfelelő képek megtekintését és esetleges változtatását. Ezeket a programokat nevezzük 3D-s modellező programoknak.*

Az elkövetkezőkben a POVRAY-hoz illeszkedő, legismertebb modellezők jellemzőit tekintjük át:

Breeze Designer (freeware)

A Breeze Designer a POVRAY-hez az egyik legjobb interaktív tervezőfelületet biztosítja. Előnye, hogy freeware program, korlátlanul felhasználható.



302. sz. ábra
A Breeze Designer kezelőfelülete

A program kezeli az összes primitívet, grafikus objektumot: például a True Type fontokat, a megvilágítást, a kamera elhelyezést, a spline modellezést, textúrákat és az egyszerűbb animációs funkciókat. Saját plugin formátuma van, amivel a programfunkciók gyakorlatilag korlátlanul bővíthetők.

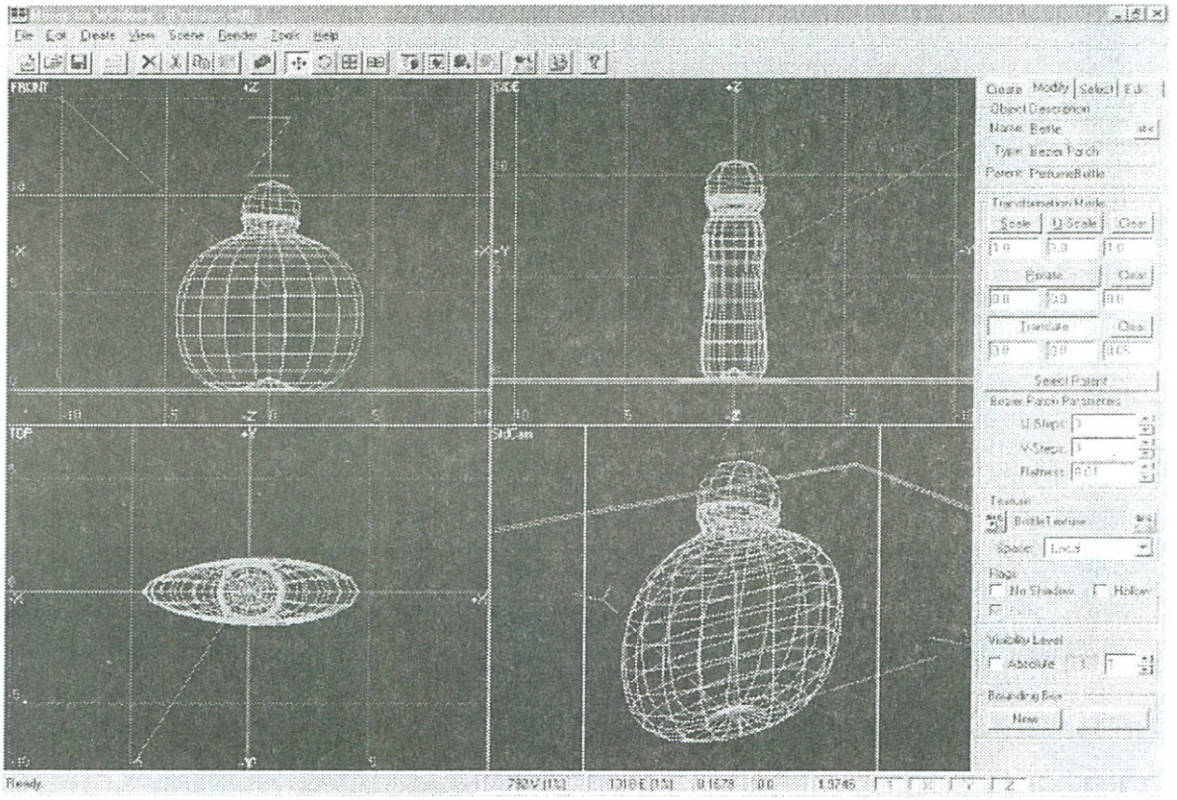
Képes importálni az AutoCad (.dxf), a 3D Studio (*.3ds) fájlformátumait. Hátránya, hogy nem fogadja a PovRay (*.pov) fájlokat.*

A fájlokat POVRAY (.pov), AutoCad (*.dxf), Renderman (*.rib) és VRML (*.wrl) formátumokba lehet exportálni.*

Hátrányai:

- Nem képes a kamera nézetet megjeleníteni,
- A CSG műveletek eredményét nem képes grafikusan megjeleníteni,
- Az objektumokat nem lehet a cursorral forgatni, csak a forgási szögek megadásával.

Moray for Windows (shareware)

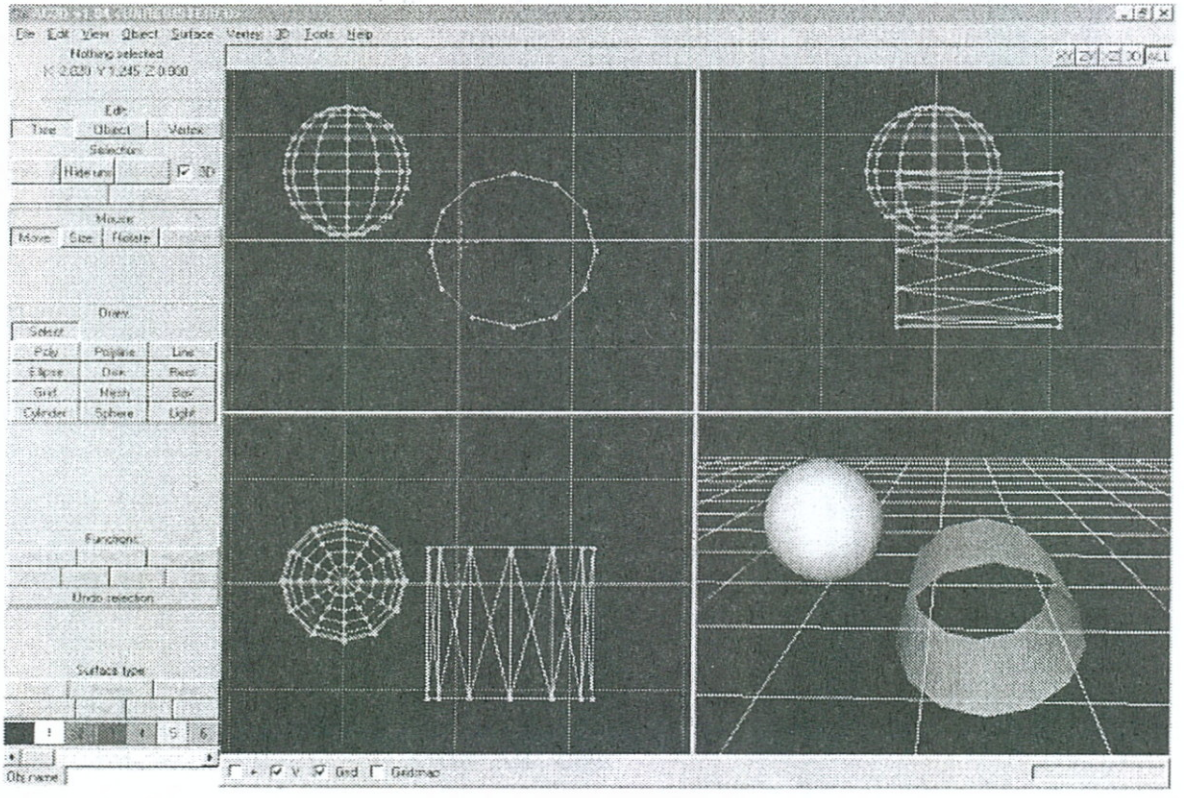


303. sz. ábra
A Moray for Windows kezelőfelülete

Felhasználóbarát tervezőprogram a POVRAY-hoz, mellyel a megjelenítés nagyon gyors, de csak huzalvázas. Nem kezeli az animációt. Erőssége:

- minden POVRAY objektumot támogat, a CSG műveleteket grafikusan is bemutatja,
- skálázás, forgatás, mozgatás egérrel is lehetséges.

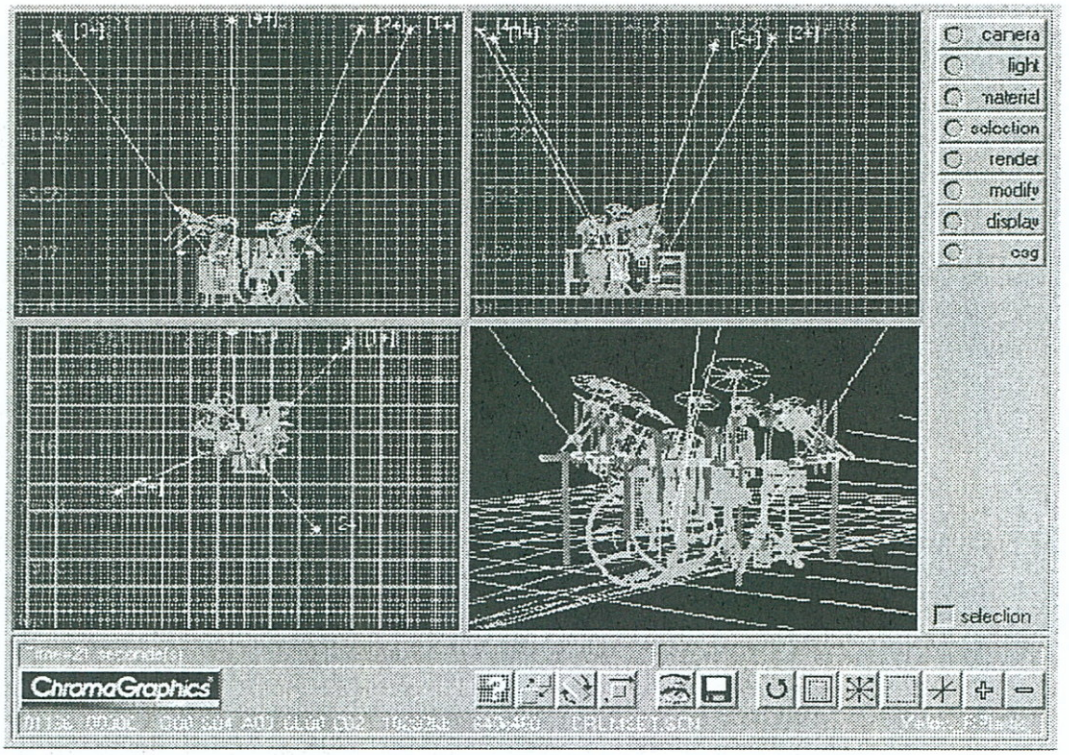
Az AC3D (shareware)



304. sz. ábra
Az AC3D kezelőfelülete

Ez a modellező a DOS, WINDOWS mellett a Linux-on is futóképes. Erőssége a spline kezelés. Széleskörű export-import lehetőséget biztosít pl. Lightwave fájlokat is képes importálni. Nem kezeli az animációkat, a True Type betűformák helyett saját speciális Fontformátumokat használ. Hátránya, hogy Win95 alatt rendkívül lassú.

PovLab (shareware)



305. sz. ábra
A POVLAB modellező kezelőfelülete

A PovLab eredetileg egy DOS-os program volt, de ma már problémamentesen futtatható WIN95 alatt is. A kezelőfelületre nagyon hasonlít a 3D Stúdióra. Menüvezérelt, a 3D-s generálásban nagyon gyors, de valósidejű renderelt képet nem képes bemutatni. Az összes 3D-s objektum primitív definiálható, ezek 3D-s transzformációt kezeli. Animáció előállító funkciója nincs.

8.6.3. POVRAY Plug-in-ok

Az INTERNETEN a POVRAY-be beépíthető plugin-ekből számtalan megtalálható. Néhány freeware ezek közül, melyet Chris Colefax készített:

- Lens Effects – Több mint 20 speciális effektus, ezek kívánságra lobognak, villannak
- Galaxy – Véletlenszámmal generált világűrjelenetek, ködök, csillagok
- Liquid Spray – Folyékony „spray" pl.: vízesés
- Objekt Exploder – 3D-s objektumok felrobbanthatók

- Clock Modifier – Egyszerűsíti a POVRAY Clock funkciójának használatát
- Object Bender – Tetszőleges objektumot segítségével meg lehet nyújtani.
- Spline Generator – Görbe ívdarabokat generál

8.6.4. További grafikus freeware és shareware programok

Az elkövetkezendőkben néhány grafikus freeware és shareware programot mutatunk be felsorolásszerűen:

- Behemot Graphics Editor V.0.8.1. (freeware): grafikus 3D-s objektumgenerátor (b-rep, NURBS) és renderelő. Ismeri a CSG műveleteket, DXF és VRML és MPEG formában exportálhatók a fájlok
- M3D (feeware): függvény egyenlet alapján 3D-s felületet kirajzoló program
- Metamorpher 98 (freeware): kiinduló és befejező kép közötti morphing-ot készítő program
- SGRAM (Custom Stereograms) (freeware): sztereogram készítő program
- Alice (freeware): szövegvezérelt, interaktív 3D-s modellező
- WcvT2Pov (freeware): 3D-s fájlformátum konverter (DXF, 3DS és POV)
- Street Graphics V.1.0.6 (shareware): vektoros rajzoló program
- Winfractint (freeware): fraktálgenerátor (Windows, Win95)
- B_Spline (freeware): B_Spline generátor
- Irfan View 32 (freeware): képnéző és konvertáló

- AnmanieSmp V2.0 (freeware): grafikus eszköztár képrészletek nagyítására, összevonására, egérrel való vontatására
- Ico2Bmp V.1.0. (freeware): a *.ico file-okat *.bmp-be konvertálja
- GuckMal 2000 V0.1 (shareware): a program közel 40 képformátumot dolgoz fel
- Carricature V1.8.1 (freeware): képfeldolgozó program, mellyel karikatúrákat lehet készíteni
- Microangelo 98 V4.72 (shareware): ikonkezelő, készítő program
- Microangelo Gif animator V.1.0 (shareware): Gif animáció készítő
- PCD-Viewer (shareware): Photo-CD néző és konvertáló, dia előállító funkció, hang és szövegbeágyzás
- Kamera V1.1 (shareware): eszközkészlet, mellyel képernyő fotókat lehet készíteni és BMP formátumba letárolni
- Fractindos (freeware): DOS-os fraktálgenerátor

? ELLENŐRZŐ KÉRDÉSEK

497. *Milyen lépései vannak a Viewing-Pipeline-nek, és ezekben milyen koordináta-rendszert használunk fel?*
498. *Milyen hardver elemek játszanak szerepet a Viewing-Pipeline 1,2,3,(4) lépésében, és ezek közül melyik teljesítménye a meghatározó a vektorgrafikában?*
499. *Milyen hardver eszköz teljesítménye a meghatározó a Viewing-Pipeline (4),5,6 lépésében?*
500. *Miért fejlesztették ki az MMX processzorokat?*
501. Jellemezze az MMX processzorokat!
502. *Gyorsítják-e az MMX processzorok a vektorgrafikus és rasztergrafikus programcsomagokat?*
503. *Milyen új hardver lehetőségeket biztosít a Pentium III processzor a számítógépes grafikában?*
504. *Mikor lehet kihasználni a PIII grafikus képességeit?*
505. Milyen grafikus API-t optimalizáltak a PIII utasításkészletére?
506. Mi tette szükségessé az új típusú buszok és csatolók kifejlesztését?
507. *Mit jelent az USB?*
508. Hogyan csatlakoztathatók a perifériák az USB-hez?
509. Sorolja fel az USB legfontosabb jellemzőit!
510. *Milyen perifériákhoz fejlesztették ki a FIREWIRE-t?*
511. Miért jött létre az AGP.
512. *Milyen rendszerelemeket köt össze az AGP, és mennyi az átviteli teljesítménye?*
513. *Mennyi az AGP memóriaigénye?*
514. *Mennyi a grafikus szoftverek jellemző memóriaigénye?*
515. Milyen RAM memóriákat célszerű használni a számítógépes grafikában?
516. *Milyen speciális felépítése van a videomamóriának?*
517. Mi az SGRAM?
518. Mi a WRAM?
519. Mi jellemzi a 3DRAM-ot?
520. Hogyan függ a monitorvezérlőkártyák memóriaszükséglete a monitor felbontásától és a színkódolástól?
521. *Hogyan számítjuk ki a frame-buffer memóriaigényét?*
522. *Milyen esetben elegendő egy vektorgrafikus rendszer számára az átlagos PC merevlemez kapacitása?*
523. *Mikor szükséges nagy közvetlen elérésű háttértár kapacitás egy grafikus rendszer számára?*

524. Mikor célszerű egy grafikus rendszerben DVD tárolót alkalmazni.
525. *Sorolja fel egy professzionális grafikus felhasználásra megfelelő monitor jellemző műszaki adatait!*
526. *A grafikus kártya és a monitor milyen műszaki paramétereinek kell összhangban lennie?*
527. Miért alakultak ki a 2D-s, 3D-s gyorsító chipek és kártyák?
528. *Miben mérjük a monitorvezérlő kártyák és a 3D-s gyorsítók teljesítményét?*
529. *Milyen részegységekből épül fel a monitorvezérlő kártya?*
530. *Mi a különbség a gyorsító chippel vagy anélkül működő monitorvezérlőkártya között?*
531. *Sorolja fel azokat a legfontosabb tulajdonságokat, melyek egy korszerű grafikus kártyát jellemeznek!*
532. *Mi különbség van az abszolút és relatív koordinátás input eszközök között?*
533. *Sorolja fel a relatív koordinátás eszközöket!*
534. *Sorolja fel az abszolút koordinátás eszközöket!*
535. *Mit nevezünk lokátor eszköznek?*
536. *Sorolja fel a képbeviteli eszközöket!*
537. *Mi a dpi?*
538. *Mi különbség van a szkenner fizikai és logikai felbontása között?*
539. Jellemezze a kisteljesítményű szkennereket!
540. Jellemezze a nagyteljesítményű szkennereket!
541. *Hogyan működnek a digitalizáló táblák (tablett)?*
542. Milyen paramétereket célszerű értékelni egy tablett vásárlásakor?
543. *Mire használható a digitális kamera és fényképezőgép a számítógép grafikában?*
544. Milyen szoftverek befolyásolják a raszteres beviteli eszközök alkalmazásának hatékonyságát?
545. *Sorolja fel a számítógépes grafikában alkalmazható output eszközöket!*
546. *Jellemezze a professzionális nyomtatókat!*
547. *Milyen plottertípusokat ismer?*
548. Adja meg egy korszerű plotter néhány jellemzőjét!
549. *Sorolja fel a számítógépes tervezés néhány részterületét, és adjon meg néhány professzionális tervezőprogramot!*
550. *Milyen célokra állíthatunk elő grafikákat például a COREL DRAW-al?*
551. *Milyen programcsomagokat használnak fel a DTP területén?*
552. *Nevezzen meg legalább három programcsomagot, mellyel a filmipar számára készítenek animációkat!*

553. Milyen haásai vannak a grafikus programcsomagok piacán élesedő versenynek?
554. *Mire használható leginkább a 3D Studio MAX?*
555. Hogyan kezeli a 3D Studio MAX a térgörbéket és felületeket ezek definiálása és renderelése során?
556. Mi a RadioRay?
557. *Milyen API-kat támogat a 3D Studio MAX?*
558. *Melyik tervezőprogram a legelterjedtebb a PC alapú rendszerek körében, és melyek a fő felhasználási területei?*
559. *Milyen AUTOCAD alapú integrált gépészeti és építészeti szoftvert ismer?*
560. Ismertesse az Architectural Desktop néhány funkcióját!
561. Ismertesse a Mechanical Desktop néhány funkcióját!
562. Mire használható a 3D Studio VÍZ?
563. *Milyen fontosabb részekből áll a Corel 9 verziójú rajzolóprogramja?*
564. Milyen célokra használták kezdetben a Corel DRAW-t?
565. *Ismertesse a Corel DRAW néhány újabb funkcióját!*
566. Milyen a Corel DRAW 8 és 9 verziójának memóriaigénye?
567. *Mit tartalmaznak a Corel objektumkönyvtárak, és miért előnyös a használatuk?*
568. *Mutassa be az ADOBE PHOTOSHOP lehetőségeit legalább három példán keresztül!*
569. *Milyen célra készítünk színkivonatot (color plate) a PHOTOSHOP-al?*
570. *Mondjon legalább két példát a PHOTOSHOP filtereire!*
571. *Mit értünk virtuális valóság alatt, és mi a VRML?*
572. *Mi jellemzi a VRML 1 verzióját?*
573. *Mi a VRML fájlok kiterjesztése?*
574. Hogyan definiáljuk a színeket VRML-ben?
575. Mit értünk séta (walk) alatt a virtuális térben?
576. *Mit jelent a VRML fájlok lejátszása, hogyan változtathatjuk helyzetünket a virtuális térben?*
577. Mivel lehet lejátszani VRML 1 fájlokat?
578. *Ismertesse a VRML 2 új lehetőségeit a VRML 1-hez képest!*
579. *Mire használható a COSMO PLAYER?*
580. Mire használható jelenleg a VRML, és a jövőben mi várható?
581. *Hogyan definiáljuk a modelltér elemeit a POVRAY-ben?*
582. *Mi az előnye és a hátránya a POVRAY-nek?*
583. *Mit nevezünk 3D-s modellezőknek?*
584. *Jellemezze a Breeze Designer modellezőt!*
585. Jellemezze Moray modellezőt!
586. Jellemezze az AC3D modellezőt!

587. Jellemezze a Povlab modellezőt!
588. Milyen programmal lehet morphing-ot megvalósítani?
589. Milyen programmal lehet sztereogramot készíteni?
590. Milyen programmal lehet B-spline-okat, illetve fraktálokat generálni?

IRODALOMJEGYZÉK

SZAKKÖNYVEK, TANKÖNYVEK, JEGYZETEK, TANULMÁNYOK

Bartha Gábor: Corel Ventura Publisher. LSI 1996.

Bartha Gábor: Nagy Corel könyv I–II. LSI

Bartha István – Bartha Gábor: Corel DRAW! 3.0... a csapattagok. LSI 1996

Berke József, Hegedűs Gy. Csaba, Kelemen Dezső, Szabó József: Digitális képfeldolgozás és alkalmazásai. Keszthelyi Akadémia Alapítvány 1998

Geier János: Egy komputációs modell a sztereo párosítási probléma megoldására (Kandidátusi értekezés, Budapest 1994)

Hajós György: Bevezetés a geometriába. Tankönyvkiadó, Budapest, 1987

Horváth Imre – Juhász Imre: Számítógéppel segített gépészeti tervezés. Műszaki könyvkiadó, 1996

Jakab Zsolt, Juhász György, Vészi József: Adobe Photoshop. ComputerBooks, Budapest 1996

László József: A VGA kártya programozása. Computerbooks 1997

Nagy Sándor – Perjés László: A számítógépes grafika. LSI, 1996

Pétery Kristóf: Autocad 14. LSI, Budapest 1999

Pétery Kristóf: CorelDRAW 6. LSI, Budapest 1997.

Pétery Kristóf: Rajzolás és szerkesztés számítógéppel. Kossuth Könyvkiadó 1996

Pintér Miklós: Autovision. ComputerBooks, Budapest 1997

Siklósi Attila: Mindennapi tipográfiánk. Műszaki Könyvkiadó 1997

Székely Vladimir, Poppe András: A számítógépes grafika alapjai IBM PC-n. Computerbooks 1997

Tóth Dezső: Multimédia mikroszámítógépes környezetben. LSI, 1996

Kris Jamsa, Phil Schmander, Nelson Yee: VRML programok könyvtára I–II. kötet. Kossuth Kiadó 1998

Ch. Spanik – H. Rügheimer: A multimédia alapjai. Kossuth Kiadó 1997

Színtan, Fénytan, Az emberi látás, Refraktometria
http://www.physic,Julis.kee.hu/3_ideig.html

Bansley M. F. – Hurd P. L.: Fractal image compression. AK Peters Ltd, Wellesley 1993.

Dieter W. Fellner: Computergrafik (2. Átdolgozott és bővített kiadás)
http://medoc.htwk-leipzig.de/interdoc/cd1/proben/fellner.htm

ENCARNACAO, J. L.–PEITGEN, H. O.–SAKAS, G.–ENGLERT, G.: Fractal Geometry and Computer Graphics, Springer-Verlag, Berlin Heidelberg, 1992

PEITGEN, H. O.–SAUPE, D.: The Science of Fractal Images, Springer-Verlag, Berlin Heidelberg, 1988

Az XWINDOWS felhasználói felület
http://www.itb.hu/ajanlasok/a7/html/a7_4_3.htm

Bevezetés a számítógépi grafikába – jegyzetlapok
http://valerie.inf.elte.hu/~krammer/eltettk/grafika/jegyzet/index.html

OpenGL grafikus rendszer alkalmazása Visual C++rendszerben (PhD diplomamunka)
http://www.iit.uni-miskolc.hu/segedletek/nehez.phd/open.htm

Computer-Graphik spielend lernen
http://www.gris.uni-tuebingen.de/gris/grdev/java/doc/html/MainPage.html

Übung zur Vorlesung Graphische Datenverarbeitung
http://www.zenger.informatik.tu-muenchen.de/lehre/vorlesungen/graphik/

Prof. Dr. Schiedermeier Blockunterricht WS1995/96 Computergraphik
http://f7alpha1.informatik.fh-muenchen.de/~schieder/blockunterricht-ss96/index.html

Computergrafik, Einführung
http://medoc.rz.uni-leipzig.de:1302/Books/encarnao/html/Inhaltsz.html

Jed Hartman, Josie Wernecke: The VRML 2.0 Handbook: Building Moving Worlds on the Web. Addison-Wesley Publishing Co. 1996

Rikk Carey, Gavin Bell: The Annotated Vrml 2.0 Reference Manual. Addison-Wesley Publishing Co. 1997. A teljes könyv megtalálható még:
http://www.best.com/~rikk/Book/

Michael Stein, Eric Bowman, Gregory Pierce: Direct3d: Professional Reference. New Riders Publishing, 1997

Mason Woo, Jackie Neider, Tom Davis: Opengl Programming Guide: The Official Guide to Learning Opengl, Version 1.1 Addison-Wesley Publishing Co. 1997

OpenGL Reference Manual: The Official Reference Document to Opengl, Version 1.1

A. L. Yarbus: Eye Movements and Vision. Plenum, New York, 1967

Aurum – Boca: 3D Studio MAX. Aurum DTP Studio, Budapest, 1997

M. A. Arnold: Non Uniform Rational B-Splines zur Darstellung von Kurven und exakten Repräsentationen von Kegelschnitten. Diplomarbeit, Tübingen, 1988.

B. A. Barsky: Computer Graphics and Geometric Modelling Using Beta-splines. Springer Verlag 1988.

James D. Foley, Andries Van Dam, Steven K. Feiner, John Hughes: Computer Graphics: Principles and Practice. Addison-Wesley Publishing Company, 1990

I. Herman: Projective Geometry and Computer Graphics. In M. Grave und M. Roch, Hrsg., Advances in Computer Graphics IV. Springer Verlag, Berlin-Heidelberg-New York-Tokyo, 1989.

Josef Hoschek, Dieter Lasser: Grundlagen der geometrischen Datenverarbeitung. Teubner, Stuttgart, 1992.

C. M. Hoffmann: Geometric and Solid Modeling: An Introduction. Morgan Kaufmann Publishers, Inc., 1989

Michael F. Cohen, John R. Wallace: Radiosity and Realistic Image Synthesis. Academic Press, 1. Auflage, 1993.

SZAKCIKKEK

Komenczi Bertalan: Az információs társadalom és az oktatás. Új Pedagógiai Szemle 1997 szeptember

Kabdebo György: A látszólagos valóság. Természet Világa 1995. szeptember

Nemzeti Informatikai Stratégia Kezdeményezés. I. Globális trendek

Térbeli sebesség, 3D gyorsítóchipekre épülő monitorvezérlők tesztje. CHIP. 1998. április

Kenzler Mihály: Folyékony képek. PC WORLD 1996. november

Kenzler Mihály: Fractal Design Expression, Digitális (ön)kifejezés. PC WORLD 1997. június

Tihanyi László: Jönnek a digitális fényképezőgépek. PC WORLD 1996. december

Lucz Géza: A hálózat használata a számítógépes grafika területén. NiiF Információs Füzetek, 1995

Itt a CorelDRAW 8. Telecomputer 3 évfolyam 5. szám 1998 március 11, 1998

Lévay István: Az AutoCAD mérnöki tervezőprogram rövid leírása
http://www.iit.uni-miskolc.hu/segedletek/autocad/autocad1.htm

M. K. Hassan: Fraktálok, multifraktálok és a bonyolultság tudománya. (Magyar fordítás a Virtual Physics 7. Számából)
http://www.kfki.hu/~cheminfo/hun/olvaso/fraktal/frakmult.html

Dr. Dörte Haftendom: Chaos und Fraktale
http://rzserv2.fh-lueneburg.de/n1/gym03/homepage/feacher/mathe/chaos/chaos.htm

Prof. Dr. T. Rauber: Graphik und Animation
http://infwww.informatik.uni-halle.de/~schmidt/grapx2.htm

Mandelbrot and Julia sets for other functions
http://www.math.rochester.edu:8080/courses/current/MTH215/lecture12.html

Erdős Márton – Nagy Imre: Boszorkánykonyha (13 új 3D-s kártyatesztje). CHIP magazin 1999 február

Kép a gépben (szkennerek tesztje). CHIP magazin 1996 május

Monitorok - összehangolva a géppel. Telecomputer 2. évfolyam 15. Szám 1997 augusztus

Hörcsik Imre: Látótávolságban az AUTOCAD Architectural Desktop. CAD világ 1998 2. évf. 4. szám

Hörcsik Imre: A látványon alapuló tervezés. Megjelent a 3D StudioVIZ. CAD világ 1998. 2. évf. 4. szám

Gyenge Balázs: DINAMIX. Dinamikai modellezés a 3D studioMAX-ban. CAD világ 1998. 2. évf. 4. szám

Sebők Róbert: Leselkedő: AutoCAD Mechanical Desktop 3.0. CAD világ 1998. 2. évf. 5. szám

Cservenák Róbert: HP DesignJET. CAD világ 1998. 2. évf. 5. szám

Siklósi Attila: Generálozzák a Vályogtéglát. Adobe Photoshop és PageMaker. CHIP magazin 1997. április

Hardver és szoftver raszterizálók (RIP) http://www.digit.hu/rip.htm

Hatékonyabb a munka az új Photoshoppal. PC WORLD 1998. július

Makk Attila: A legjobb tíz: Lapolvasók. PC WORLD 1998. március

GKS, D3-Library http://www.tu-bs.de/rz/software/gks/d3lib/d3lib.html

Berke József – Kocsis Péter – Kovács József: Veszteséges képtömörítő eljárások pszihovizuális összehasonlítása http://tutor.hok.bme.hu/mmo/1/kpfeldol.htm

Sztereogram http://www.c3.hu/butterfly/Szucs/project/3a.html

Grafikával foglalkozó cikkek magyar nyelven http://www.lezlisoft.com/grafika.htm

Claudio Andre, Till Clemeus: POV-Ray Proseminar. Technische Universität München 1997 július
http://www.informatik.tu-muenchen.de/~clemens/POVray/pov3.html

K. Weiler, K. Atherton: Hidden surface removal using polygon area sorting. Computer Graphics (SIGGRAPH '77 Proceedings), 1977. Juli

CERN. (December 1996) Discrete Cosine Transform.
http://www.cern.ch/Physics/DataAnalysis/BriefBook/AN1Opp/node58.html#57

Flippini, Luigi (1996): A Quick Tutorial on Generating a Huffman Tree.
http://www.crs4it/~luigi/MPEG/huffman_tutorial.html

Scott, John. (n.d.) Huffman Coding
http://www-students.doc.ic.ac.uk/~jcs2/huffman/

Wide Area Communications (1996): GIFs and JPEGs.
http://www.widearea.co.uk/designer/compress.html

Nemoda Balázs: Fraktál alapú képtömörítések (Magyar Elektronikus Könyvtár)
http://hal2000.elte.hu/doli/doli_htm

Nehéz Károly: A JPEG képtömöritési eljárás és a hozzá tartozó adatstruktúrák
http://www.iit.uni-miskolc.hu/segedletek/jpeg/JPEG.html

Őfelsége Pentium III. PC WORLD 1999. március

Itt lenne az idő DVD-re váltani? PC WORLD 1999. március

Gyenge Balázs: RadioRay, Még egy renderelő eljárás a 3D StudioMAX-hoz. CADvilág 1998. 1. Szám

Kulcsár Ferenc: Görbülő „MAX" világ. NURBS-alapú modellezés a 3D StudioMAX-ban. CADvilág 1998. 1. Szám

Horváth Attila: Virtuális valóság, avagy a számítógépes képalkotás és látványtervezés. CADvilág 1997. 2. Szám

Brian A. Barsky: Rational Beta-splines for Representing Curves and Surfaces. IEEE Computer Graphics and Applications, 1993. November

Brian B. Barsky, Tony D. DeRose: Geometric Continuity of Parametric Curves: Constructions of Geometrically Continuous Splines. IEEE Computer Graphics and Applications, 1990. January

H. Gouraud: Continuous Shading of Curved Surfaces. IEEE Transactions on Computers, 1971. Juni

Jack. E. Bresenham: Algorithm for Computer Control of a Digital Plotter. IBM System Journal 4 (1), IBM, 1965.

M. Cyrus, J. Beck: Generalized Two- and Three-Dimensional Clipping. Computers & Graphics, 3. 1978.

R. L. Cook, L. Carpenter, E. Catmull: The Reyes Image Rendering Architecture. ACM Computer Graphics, 1987. Juli

Andreas Schilling: A new simple and efficient anti-aliasing with subpixel masks. (SIGGRAPH '91 Proceedings), 1991. July

INTERNET PUBLIKÁCIÓK

Számítógépes grafika, multimédia, digitális képfeldolgozás szakirodalma
http://www.simsrv.cs.uni-magdeburg.de/lehre/cgl/frames/noden.html

Digitális kép és videotömörítő technikák
http://www..mek.iif.hu/porta/szint/muszaki/szamtech/multimed.picforms.hun

Az informatika jelenlegi és jövőbeli hatása
http://meh.hu.net:8080/egyeb/mis/2resz.htm

Robert Lindstrom: Az új média oktatása
http://www.mek.iif.hu/porta/szint/muszaki/szamtech/multimed/mediaokt.hun

Színkalibráció
http://www.lezlisoft.com/cikkek/szinkali.htm

Zbigwiew Koziol: Fraktálok
http://www.kfki.hu/~cheminfo/hun/olvaso/fraktal/hunt.html

Állománytípusok
http://www.mach.uni-miskolc.hu:8080/~szg/FILETYPE.HTML

Homogene Koordinaten
http://www.fh-jena.de/contrib/fb/et/personal/sauer/sauer/cg-vr/Grundlagen.html

Sichtbarkeits algorithmen unter Benutzung von Tiefen-Sortieren-Verfahren
http://www.fh-jena.de/contrib/fb/et/personal/sauer/sauer/cg-vr/Algorith.html

Lambert - Strahler
http://www.fh-jena.de/contrib/fb/et/personal/sauer/sauer/cg-vr/Beleucht.html

2-D und 3-D-Texturen
http://www.fh-jena.de/contrib/fb/et/personal/sauer/sauer/cg-vr/Textur.html

Raytracing
http://www.fh-jena.de/contrib/fb/et/personal/sauer/sauer/cg-vr/Raytrac.html

Das Radiosity – Verfahren
http://www.fh-jena.de/contrib/fb/et/personal/sauer/sauer/cg-vr/Radiosit.html

SRGP
http://www.iam.unibe.ch/software/SRGP/SRGP.html

Sun PHIGS
http://www.iam.unibe.ch/software/sunPHIGS/sunPHIGS.html

VR in Medizin
http://www.uni-weimar.de/architektur/infAR/lehre/Course01/medizin.html

Színek és képek az INTERNET-en
http://www.lezlisoft.com/cikkek/wwwcikk3.htm

AGP fórum
http://www.agpforum.org/

Fraktálok
http://www.cnam.fr/fractals/mandel.html

Svéd egyetemi hálózat: képtárak, fraktálok, sztereogrammok, ftp szervere
ftp://ftp.sunet.se/pub/pictures/

Geometry Center Graphics Archive (speciális effektusok) (University of Minnesota)
http://www.geom.umn.edu.80/graphics/

Grafikus archívumok címei
http://galaxy.einet.net/galaxy/Leisure-and-Recreaction/Pictures.html

Textúra szerver címek
http://www.meat.com/textures/textures/html

Autodesk honlapja
http://www.autodesk.com

Corel honlapja
http://www.corel.com

Fraktálgenerátor
http://spanky.trium.ca/www/fractint/fractint.html

3D Studio MAX honlap
http://www.3dmax.com

Virtuális Mars panoráma VRML-ben
http://mars.sgi.com/worlds/hth-planet/models/mp_latest_vlo.html

A Radiosity eljárás szervere
http://radsite.Ibl.gov/radiance/HOME.html

VRML címek, szabványok
http://www.sdsc.edu/vrml

GIS rendszer oktatóanyag
http://152.66.5.65/tutor_h/idrisi/tutindex.htm

3D Szimultációs szoftver (orvostudomány, robotika)
http://iregt1.iai.fzk.de/

ACM Transactions on Graphics folyóirat honlapja
http://www.acm.org/pubs/tog/

SGI Magyarországi honlapja
http://www.sgi.hu/

Vizuális művészet címgyűjtemény
http://www.yahoo.com/Arts/Visual_Arts/

Clipartok, animált GIF-ek stb.
http://www.TheShockZone.com/

Institute of Computer Graphics kutatási publikációk (Wien)
http://www.cg.tuwien.ac.at/research/

USB leírás magyar nyelven
http://www.iit.uni-miskolc.hu/~jonas1/f3-3.html

A HP cég karikatúrái a számítógépes grafika alapfogalmairól
http://www.hp.com/mhm/CompGrfxFUNdamentals

VRML világok
http://www.virtpark.com

4D Vision honlap (Voxel Shading)
http://www.4dvision.com/

VRML 2-0 kézikönyv
http://cosmosoftware.com/developer/handbook

Cosmo Player
http://www.sgi.com/software/cosmo

VRML Consortium
http://www.vrml.org/

USB
http://www.usb.org/
http://www.usb.net/

3D animáció és renderelés (Silicon Graphics)
http://www.sgi.com/apps/3d_anim/

Virtuális valóságok (pl. kutatási programok)
http://www.sgi.com/virtual_reality/

Egy érdekes VRML világ sztereohanghatásokkal
http://fly.hiwaay.net/~cbullard/index.htm

VRML Architectural Group
http://vag.vrml.org

Grafikus demok
http://avalon.viewpoint.com

Shockwave honlap
http://www.macromedia.com/shockwave/

Computergrafik. Előadásanyag 1997 Universität Osnabrück
http://www-lehre.informatik.uni-osnabrueck.de/~eg/skript/skript.html

Metro VRML modellje
http://metro4.fph.hu/VRML/index.html

POVRAY honlapja
http://www.povray.org

Textúra könyvtárak
http://www.websharx.com/~ttbrown/tbtwfs.html

Grafikai programok linkgyűjteménye
http://www.extra.hu/linkmix/grafika.htm

Moray home page
http://www.stmnc.com/moray/

Breeze Designer
http://www.povray.org/ftp/pub/povray/utilities/modellers/breeze/BreezeDesigner.html

Textúra könyvtár a povray-ben
http://texlib.povray.org/color.html
http://texlib.povray.org/textures.html

D. P. M. = Deutsche Povray Magazin, cikkek, tippek
http://userpage.fu-berlin.de/~frexka/dpm/themen.htm

Metamorpher
http://www.gibb.ch/ressort-informatik/wieser/morph/metamorpher.html

Modellezők a POVray-ben, összehasonlítás
http://spike.forfree.at/~breuers/programs/ht

www.apm.tuwien.ac.at/docs/lva/mmgdv/k1_003.htm

Truespace 4 (Hybrid Radiosity, Volumetrikus renderelés)
http://www.caligari..de/extern/caligari/resource.nsf/common/attach/$file/ts4.html

DIRECTX letöltés
http://www.kvantum.hu/letoltes

NASA űrhajó VRML modellek
http://mars.jpl.nasa.gov/mgs/movpics/mgs_vrml/mgs_vrml.html

VRML háló
http://www.vrweb.net

INTERGRAPH SOLID EDGE
http://www.intergraph.com

VRML 2
http://vrml.sgi.com

The VRML Respository
http://www.sdsc.edu/vrml

OPEN GL
http://www.opengl.org
http://www.sgi.com/Technology/OpenGL/index.html

Fahrenheit
http://www.sgi.com/fahrenheit/

3D Studio MAX R2
http://www. construnet.hu/7DVision/max2new.html

WWW grafika a Photoshopban
http://www.weblapok.hu/rajz/cend001.htm

Digi_Art_3D
http://userpage.fu_berlin.de/~frexka/dpm/da_more.htm

M3D for Windows 95 and Windows NT
http://man.simplenet.com/graphics/3D/m3d/default.htm

WEB3D CONSORTIUM (VRML-NG) Specification
http://www.vrml.org

Creating Curver
http://www.csl.mtu.edu/cs390-2/www/LAB/curve/create.html

Creating surfaces
http://www.csl.mtu.edu/cs390-2/www/LAB/surface/create.html

3D gyorsítók: Kérdések és válaszok, 3dFAQ:
http://195.56.48.34/pcabc/hardver/3dfaq.htm

3Dfx Magyarország-Voodoo 2
http://www.aspect.hu/3Dfx/voodoo2.htm

SOFTIMAGE 3D
http://www.creative.hu/medic/termekek/softimage/news.html

INFAR
http://www.uni-weimar.de/architektur/inf/skripte/bildbearbeitung/vektgraf.htm

Digi_Art_3_ V. 5.03 modellező a Povray-hoz
http://www.digi-art.de

Fotorealisztikus renderelés
http://www.geocities.com/SiliconValley/Way/2419/3Doverview.html

Clipart
http://www.clipart.com

Open GL alapítvány
http://www.opengl.org

Terragen tájgenerátor
http://www.terragen.com

Minden cím, ami a VRML-el kapcsolatos, pl. newsgroup
http://www.pla-net.net/~vpyse/index.html

SZÓSZEDET

ISBN 963-577-243-2