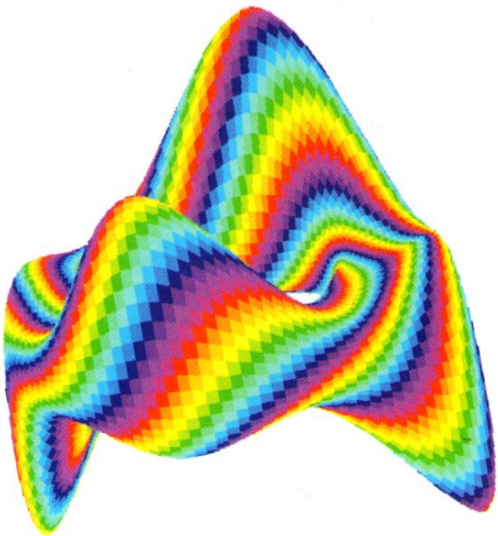
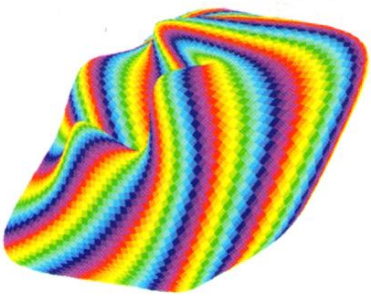
Interaktív grafika

FÜZI JÁNOS

GÖRBESZERKESZTÉS
FELÜLETSZERKESZTÉS
ANIMÁCIÓ
VIRTUÁLIS SZERKEZETEK





ComputerBooks

FÜZI JÁNOS

Interaktív grafika

GÖRBESZERKESZTÉS
FELÜLETSZERKESZTÉS
ANIMÁCIÓ
VIRTUÁLIS SZERKEZETEK

LEKTOR

DR. VERMES IMRE
BENKŐ LÁSZLÓ

COMPUTERBOOKS
BUDAPEST, 1997

©Kiadó: ComputerBooks Kiadói, Szolgáltató és Kereskedő Kft.
1126 Bp., Tartsay Vilmos u. 12.
Telefon: 175-15-64; Tel/Fax: 175-35-91
e-mail: info@computerbooks.hu
Felelős kiadó: ComputerBooks Kft ügyvezetője
ISBN: 963 618 149 7

Borítóterv: Székely Edith

Tartalomjegyzék

Előszó

A különböző geometriai alakzatok, objektumok megjelenítésével és mozgatásával foglalkozó, *3D grafika és animáció PC-n* című könyv mintegy természetes folytatásaként, jelen munka az illető objektumok interaktív szerkesztését mutatja be.

Az interaktív grafika kiváltképpen vonzó tulajdonsága, hogy a számítógép gyorsaságát és pontosságát ötvözi a felhasználó találékonyságával. Az alkalmazott kutatás, műszaki tervezés, ipari formatervezés, építészet egyre elterjedtebb és eredményesebb eszköze. Oktatási segédeszközként, valamint a reklám- és szórakoztatóiparban ugyancsak sikerrel alkalmazható.

A könyvben bemutatott anyag különböző szinten hasznosítható. A programozásban jártas olvasó a leírt módszerek alapján adott követelményeknek megfelelő saját alkalmazásokat készíthet. A szerkesztési és megjelenítési eljárások programozási vonatkozásait számos mintaprogram szemlélteti. Aki az interpolációs és approximációs eljárások tulajdonságait kívánja megismerni, a lemezmellékleten futtatásra kész programrendszert talál.

A sík- és térgörbék, nyílt illetve zárt felületek szerkesztéséhez szükséges kontrollpont-konfiguráció (karakterisztikus keret) kialakítását felhasználóbarát fejlesztési rendszer könnyíti meg. A számítógéppel való kommunikáció eszközeit (egérkezelés grafikus módban, grafikus kurzorok, vizuális billentyűk) értelmező és kezelő eljárások, a különböző interpolációs illetve approximációs módszereket megvalósító programok, valamint néhány összetettebb, a virtuális valóság alapelemeit szemléltető alkalmazás Turbo Pascal nyelven készült, a bemutatott elvek és algoritmusok alapján viszont tetszés szerinti programozási nyelven megírhatók.

1. Az interaktív számítógépkezelés célkitűzései és eszközei

A számítógép és felhasználó közötti interaktív kapcsolat lényegét úgy fogalmazhatjuk meg, hogy a gép a felhasználót állandóan tájékoztatja a folyamatban levő műveletek állapotáról, (rész)eredményeiről és a beavatkozási lehetőségekről, illetve lehetővé teszi a beavatkozást, azaz parancsokat fogad el és teljesít.

A számítógépes grafika területén a legfontosabb célkitűzések közé tartozik az interaktív szerkesztés és megjelenítés. Az előbbi azt jelenti, hogy a grafikai objektumok (görbék, felületek, testek) alakját, színét, méreteit a felhasználó változtatni tudja, anélkül, hogy a forráskódot módosítania kellene. Megfelelő felhasználói program nem is követel meg programozási ismereteket, hanem az objektumok létrehozásához és meghatározásához szükséges adatbevitel a program futása közben történik. Az szerkesztés eredményének interaktív megjelenítése lehetővé teszi a kapott objektumok különböző irányból való megfigyelését illetve a (virtuális) térben való mozgatását.

A Turbo-Pascal fejlesztési környezet (IDE) és fordító (Compiler) már önmagukban igen erős interaktív programozási eszközök, amennyiben lehetővé teszik a készülő program azonnali fordítását, ellenőrzését és futtatását, az esetleges hibák kiküszöbölését. A paraméterek változtatásával vizsgálható ezek hatása az eredményekre. Bonyolultabb feladatok esetében a sokszori újraindítás zavaró lehet és szükségessé válik egy interaktív program megírása, amely lehetővé teszi a paraméterek futás közbeni változtatását.

A számítógép és felhasználó közötti kommunikáció legelterjedtebb eszközei a billentyűzet, az "egér" és a képernyő. Az első kettő a felhasználótól a gép felé továbbítja a parancsokat, az utóbbi a gép tevékenysége eredményeinek a megjelenítését teszi lehetővé. Ugyanakkor a felhasználó a képernyőn virtuális eszközöket hozhat létre, a billentyűzettel vagy egérrel történt cselekménynek a virtuális eszköz bizonyos rendszer szerinti kezelését feleltetve meg. Az úgynevezett "joystick", amint neve is mutatja, a számítógépes játékokkal kapcsolatban jelent meg és a folyamatos helyzetközlés eszköze.

Léteznek bonyolultabb eszközök is, például a digitális rajztábla (kész műszaki rajzok számítógépre viteléhez), a "scanner" (bonyolult ábrák, fényképek rasz-

teres átalakítása és gépre vitele), különböző mozgásérzékelők (a felhasználó fejére vagy kezére szerelt eszközök, amelyek mozgásához hozzárendelik a virtuális térben való mozgást), adatgyűjtő rendszerek. Az eredmények megjelenítését szolgálják a különböző vektoros rajzolók (plotterek) vagy a számítógép által vezetett gépek (robotok). Mindezen eszközök használatát specifikus, rendszerint a gyártó által rendelkezésre bocsátott programcsomagok (szoftver) teszik lehetővé.

1.1. A grafikus meghajtó

A legelterjedtebb számítógép-konfigurációknak megfelelő grafikus meghajtó DOS operációs rendszerben a VGA, amelynek VGAHI üzemmódja 640x480 pontos képernyőfelbontással dolgozik, VGAMED üzemmódja pedig 640x350 pontos felbontással. Az utóbbi lehetővé teszi a két (egy aktív és egy látható) képernyős rajzolást, amely a folytonosság látszatát keltő animációk elkészítéséhez szükséges.
A grafikus meghajtó elindítására szolgál az *InitGraph* utasítás, amelyben azonban meg kell adni az EGAVGA.BGI file-t tartalmazó alkönyvtárhoz vezető útvonalat. Előnyösebb olyan programegységet (unit) írni, amely beépíti a grafikus meghajtó Object formátumú eljárását (és az esetleg szükséges karakterkészletet), ugyanis az ezt használó programok végrehajtható (*.exe) változatai szállíthatók, azaz olyan gépen is futnak, amelyiken más elérési útvonallal találhatók meg, vagy egyáltalán nem találhatók .BGI file-ok.

A lemezmellékleten található GRAFIND.PAS unit forráskódja:

```
unit grafind; {grafikus meghajtó elindítása}

interface

  uses graph;
  procedure triplexfontproc;
  procedure egavgadriverproc;
  procedure grindhi;
  procedure grindmed;

implementation

  {$L trip.obj }
  procedure triplexfontproc; EXTERNAL;
                            {Triplex karakterkészlet befoglalása)
```

```
  {$L egavga.obj }
  procedure egavgadriverproc; EXTERNAL;
                            {grafikus meghajtó befoglalása}

  var gdr, gmd: integer;

  procedure incl; {befoglalás}
  begin
    if RegisterBGldriver(@egavgadriverproc) < 0 then begin
      writeln('Could not include EGAVGA.OBJ'); halt(1); end;
    if RegisterBGlfont(@triplexfontproc) < 0 then begin
      writeln('Could not include TRIP.OBJ'); halt(2); end;
    gdr:= vga;
  end;

  procedure indit; (indítás}
  begin
    initgraph(gdr, gmd, '');
    if GraphResult <> grOK then begin
      writeln('Could not initiate graphic mode'); halt(3); end;
  end;

  procedure grindhi;
  begin
    incl;
    gmd:=vgahi; {640x480 pontos képernyőfelbontás}
    indit;
  end;

  procedure grindmed;
  begin
    incl;
    gmd:=vgamed; {640x350 pontos képernyőfelbontás}
    indit;
  end;

end.
```

1.2. Grafikus kurzorok

A grafikus kurzorok olyan jelek, amelyek a képernyőn egérrel vagy bizonyos (például nyíl) billentyűkkel mozgathatók és segítségükkel képernyőpontokat vagy a képernyőn megjelenített objektumokat lehet kiválasztani. Fontos, hogy a kurzor a háttérrajzolattól függetlenül látható legyen és a kurzor mozgatása ne változtassa a háttért. Ez Turbo-Pascal-ban legkényelmesebben a *getimage* (téglalapalakú képernyőtartomány tárolása rasztermátrix (bitmap) alakban) - *putimage* (bitmap megjelenítése) utasításokkal valósítható meg.

A *putimage* parancs többféle módon végrehajtható, a megjelenítendő bitmap és a képernyő megfelelő pontjainak színei között értelmezett művelettől függően. Megjegyzendő, hogy a műveletek az illető szín kódját (0-tól 15-ig terjedő természetes szám) veszik figyelembe és nem a színt magát, tehát a paletta újraértelmezése a műveleti táblát nem befolyásolja.

Az értelmezett műveletek:

- *normalput* - a megadott helyen a bitmap-ben tárolt alakzat jelenik meg, a háttértől függetlenül

- *notput* - a megadott helyen a bitmap-ben tárolt alakzat inverze jelenik meg, a háttértől függetlenül. A paletta 16 (0-tól 15-ig) színének inverzét a 15—c képlet adja meg, ahol *c* az adott szín kódja.

- *orput* - a bitmap pontja és az azonos helyen levő képernyőpont színei között "vagy" műveletet végez. Műveleti táblája:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	3	3	5	5	7	7	9	9	11	11	13	13	15	15
2	2	3	2	3	6	7	6	7	10	11	10	11	14	15	14	15
3	3	3	3	3	7	7	7	7	11	11	11	11	15	15	15	15
4	4	5	6	7	4	5	6	7	12	13	14	15	12	13	14	15
5	5	5	7	7	5	5	7	7	13	13	15	15	13	13	15	15
6	6	7	6	7	6	7	6	7	14	15	14	15	14	15	14	15
7	7	7	7	7	7	7	7	7	15	15	15	15	15	15	15	15
8	8	9	10	11	12	13	14	15	8	9	10	11	12	13	14	15
9	9	9	11	11	13	13	15	15	9	9	11	11	13	13	15	15
10	10	11	10	11	14	15	14	15	10	11	10	11	14	15	14	15
11	11	11	11	11	15	15	15	15	11	11	11	11	15	15	15	15
12	12	13	14	15	12	13	14	15	12	13	14	15	12	13	14	15
13	13	13	15	15	13	13	15	15	13	13	15	15	13	13	15	15
14	14	15	14	15	14	15	14	15	14	15	14	15	14	15	14	15
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15

- *xorput* - a bitmap pontja és az azonos helyen levő képernyőpont színei között "kizáró vagy" műveletet végez. Jellegzetessége, hogy megőrzi a háttérpontok színére vonatkozó információt, a művelet ugyanis, amint az alábbi műveleti táblán ellenőrizhető, zérusosztómentes (mindegyik sorban illetve oszlopban mindegyik szám egyszer és csakis egyszer fordul elő). Kétszer egymás után végrehajtva visszanyerhető az eredeti háttérrajzolat, vagyis a művelet önmagának inverze. Műveleti táblája:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
2	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
3	3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
4	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
5	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
6	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
7	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
8	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
10	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
11	11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
12	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
13	13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
14	14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- *andput* - a bitmap pontja és az azonos helyen levő képernyőpont színei között "és" műveletet végez. Műveleti táblája:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
2	0	0	2	2	0	0	2	2	0	0	2	2	0	0	2	2
3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
4	0	0	0	0	4	4	4	4	0	0	0	0	4	4	4	4
5	0	1	0	1	4	5	4	5	0	1	0	1	4	5	5	5
6	0	0	2	2	4	4	6	6	0	0	2	2	4	4	6	6

7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
8	0	0	0	0	0	0	0	0	8	8	8	8	8	8	8	8
9	0	1	0	1	0	1	0	1	8	9	8	9	8	9	8	9
10	0	0	2	2	0	0	2	2	8	8	10	10	8	8	10	10
11	0	1	2	3	0	1	2	3	8	9	10	11	8	9	10	11
12	0	0	0	0	4	4	4	4	8	8	8	8	12	12	12	12
13.	0	1	0	1	4	5	4	5	8	9	8	9	12	13	12	13
14	0	0	2	2	4	4	6	6	8	8	10	10	12	12	14	14
15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

A bemutatott műveletek segítségével több lehetőség van kurzor megjelenítésére és mozgatására. Ahhoz, hogy a nem feltétlenül téglalap alakú kurzor körül változatlan maradjon a háttér, olyan műveletet használó megjelenítésre van szükség, amely bizonyos mértékben megőrzi a háttérrajzolatra vonatkozó információt.

Az egyik lehetőség az, hogy fekete alapon fehér színű kurzort rajzolunk, bitmap-ben tároljuk *(getimage)* és *xorput* módon jelenítjük meg (putimage). Így a kurzor körvonalán kívül eső pontok nem változnak. Mozgatáskor előbb megismételjük a megjelenítést a régi helyen - ezáltal a kurzor eltűnik és helyén megjelenik az eredeti háttérrajzolat, azután az új helyen jelenítjük meg a kurzort. Így a mozgás folytonos lesz, az eljárás viszonylag kis kiterjedésű (célszerű a 16x16 pixeles méret) kurzor esetén meglehetősen gyors. A háttérrajzolat azonban, inverz színkóddal, a kurzoron látható.

Az alábbi, GR_KZ_X.PAS nevű program a bemutatott módszerrel körlap alakú kurzort definiál. A felhasznált, Turbo-Pascal rendszerben nem található unit-ok a továbbiakban kerülnek bemutatásra.

```
{$N+}

uses dos, crt, graph, grafind, kozos, eger_kez;

const
  a=8;
  fel = #72; le = #80; bal = #75; jobb = #77;

var
  k, 1, x, y: integer;
  q: pointer; {bitmap heap-beli tárolási címére mutat}
  mm: word;
  kr: char;
```

```
begin
  grindhi;
  eger_k; {egér inicializálása)
  eger_t(12,12,631,472); {egér mozgástere}
  eger_h(320,240); {egér kezdeti helye}
  eger n; (az implicit egérkurzor eltűnik)
  setfillstyle(1,15); fillellipse(a,a,a,a); (kurzor alakja}
  mm:=imagesize(0,0,2*a,2*a); {bitmap memóriaszükséglete}
  getmem(q,mm); {helyfoglalás a bitmap-nek a heap-ben}
  getimage(0,0,2*a,2*a,q^); {kurzor tárolása)
  for 1:=0 to 120 do {háttérmintázat}
    for k:=0 to 160 do begin
      setfillstyle(1,k*1 mod 163 mod 16); bar(k*4,1*4,k*4+4,1*4+4);
    end;
  x:=320-a; y:=240-a;
  putimage(x,y,q^,xorput);
  repeat
    repeat
      with reg do begin
        ax:=3; intr($33,reg);
        if (cx<>x+a) or (dx<>y+a) then begin (mozgatás egérrel)
          putimage(x,y,q^,xorput); {kurzor eltűnik)
          x:=cx-a; y:=dx-a;
          putimage(x,y,qA,xorput); (kurzor megjelenik az új helyen)
        end;
      end;
    until keypressed;
    kr:=readkey; {mozgatás nyíl billentyűkkel}
    putimage(x,y,q^,xorput); {kurzor eltűnik}
    case kr of
    fel: y:=y-1;
    le: y:=y+1;
    jobb: x:=x+1;
    bal: x:=x-1;
    end;
    putimage(x,y,q^,xorput); {kurzor megjelenik az új helyen}
    eger_h(x+a,y+a);
  until kr=char(27);
end.
```

Ahhoz, hogy a kurzor háttértől függetlenül látszódjék, és ugyanakkor a kurzor körül látszódjék a háttér, a következő módon járhatunk el: — kurzor tárolása

- fekete alapon fehér kurzort rajzolunk és bitmap-ben tároljuk
- fekete alapon fehérrel megrajzoljuk a kurzor körvonalát és ezt is tároljuk

— kurzor megjelenítése
  - bitmap-ben tároljuk azt a képernyőtartományt, ahová a kurzor kerül
  - *orput* módon megjelenítjük a kurzort - így a kurzor a háttértől függetlenül fehér marad
  - *xorput* módon megjelenítjük a kurzor körvonalát - így a kurzor fekete körvonallal jelenik meg, ezért fehér mezőben is látszani fog

— kurzor mozgatása
  - megjelenítjük a régi helyen a tárolt háttértartományt - ezáltal a kurzor eltűnik és helyén megjelenik az eredeti háttérrajzolat
  - tároljuk az új helyen levő háttértartományt
  - megjelenítjük a kurzort *(orput)* és a körvonalát *(xorput).*

Az alábbi, GR_KZ_O.PAS nevű program fekete szegélyű fehér kereszt alakú kurzort mozgat tarka háttér előtt.

```
(*N+)

uses dos, crt, graph, grafind, kozos, eger_kez;

const
  a=8;
  fel = #72; le = #80; bal = #75; jobb = #77;

var
  k, 1, x, y: integer;
  q1, q2, e: pointer; {bitmap-ek heap-beli tárolási címére mutat)
  mm: word;
  kr: char;

procedure megj;
begin
  getimage(x,y,x+2*a,y+2*a,e^); {képernyőfolt tárolása}
  putimage(x,y,q1^,orput);      {kurzor megjelenítése}
  putimage(x,y,q2^,xorput);     {kurzorkeret megjelenítése}
end;

begin
  grindhi;
  mm:=imagesize(0,0,2*a,2*a);       {bitmap memóriaszükséglete)
  getmem(ql,mm) ; getmem(q1,mm) ; getmem(e,mm) ;
                                    (helyfoglalás a heap-ben)
  eger_k; (egér inicializálása}
  eger_t(12,12,631,472);            {egér mozgástere}
  eger_h(320,240);                  (egér kezdeti helye}
  eger_n;                           (az implicit egérkurzor eltűnik}
  setcolor(15); setlinestyle(0,0,3);
  line(0,a,2*a,a); line(a,0,a,2*a); {kurzor alakja}
```

```
  getimage(0,0,2*a,2*a,ql");      (kurzor tárolása)
  setcolor(0); setlinestyle(0,0,1);
  line(0,a,2*a,a); line(a,0,a,2*a); (kurzorkeret alakja)
  getimage(0,0,2*a,2*a,q2^);      {kurzorkeret tárolása}
  for 1:=0 to 48 do {háttérmintázat}
    for k:=0 to 64 do begin
            setfillstyle(l,k*l mod 71 mod 16);
            bar(k*10,1*10,k*10+10,1*10+10);
    end;
  x:=320-a; y:=240-a; megj;
  repeat
    repeat {kurzor mozgatása egérrel}
     with reg do begin
       ax:=3; intr($33,reg);
       if (cx<>x+a) or (dx<>y+a) then begin
          putimage(x,y,e",normalput); {kurzor eltűnik}
          x:=cx-a; y:=dx-a; megj;
       end;
     end;
    until keypressed;
    kr:=readkey; {kurzor mozgatása nyíl billentyűkkel}
    putimage (x, y, e", normalput) ; {kurzor eltűnik}
    case kr of
    fel: y:=y-1;
    le: y:=y+1;
    jobb: x:=x+1;
    bal: x:=x-1;
    end;
    megj; eger_h(x+a,y+a);
  until kr=char(27);
end.
```

1.3. Az egér programozása

Az egér a géppel való kommunikáció igen jól használható eszköze, gondoljunk csak arra, hogy mennyivel könnyebb és érthetőbb egy fényképen rámutatni az utolsó előtti sorban balról a negyedik személyre, mint ugyanezt végigmondani, illetve a mondottak szerint megkeresni.

Az egér kezelését a $33 - as megszakítás látja el. A funkciók és paraméterek közlése program és megszakítás-software között a regiszterek segítségével történik. Néhány, a grafikus üzemmódban használatos funkció esetén a regiszterek be- és kimeneti értékei a következőek [1]:

A regiszterek értékei			A funkció leírása
funkció	bemenet	kimenet	
AX = 0		AX = 0 AX = $FFFF BX	inicializálás - nem sikerült - megtörtént egér gombjainak száma
AX = 1			egér láthatóvá tétele
AX = 2			egér láthatatlanná tétele
AX = 3		BX bit0,1,2 CX, DX	egér koordináták lekérdezése gombok állapota: bit0-bal, bit 1-jobb, bit2-középső 1 - lenyomva, 0 - nem létezik, vagy nincs lenyomva pld.: 011 - a bal és a jobb gomb lenyomva (BX = 3) vízszintes ill. függőleges koordináta pixelben
AX = 5	BX	 AX BX CX, DX	0-bal, 1-jobb, 2-középső gomb lenyomásainak száma bit0..2 - melyik gomb volt lenyomva a kérdezett gomb lenyomásainak száma vízszintes ill. függőleges koordináta pixelben
AX = 6	BX	 AX BX CX, DX	0-bal, 1-jobb, 2-középső gomb felengedései száma bit0..2 - melyik gomb volt lenyomva a kérdezett gomb felengedéseinek száma vízszintes ill. függőleges koordináta pixelben
AX = 4	CX, DX		adott koordinátájú pontra való mozgatás
AX = 7	CX, DX		vízszintes mozgásterület bal- illetve jobb határa
AX = 8	CX, DX		függőleges mozgásterület felső- illetve alsó határa
AX = 9	BX, CX		grafikus kurzor meghatározása aktív pont helye a kurzort megjelenítő bitmap-ben kurzoralak
AX = 29	BX		képernyőkiválasztás több képernyő-lap használatakor
AX = 30		BX	képernyő számának lekérdezése
AX = 36		 BH,BL CH CL	egér meghajtó lekérdezése verziószám: BH.BL típus (1-busz, 2-soros, 3-InPort, 4-PS-2, 5-HP) IRQ-szám

Ezenkívül állítható a $33 - as megszakításhoz való másodpercenkénti hozzáfordulások száma (28-as funkció), a lépésköz Mickey/pixel-ben (15-ös funkció), az egér érzékenysége és más paraméterek. A Mickey hosszmértékegység, az egér egységnyi fizikai elmozdulását jelenti (1/200 inch).

Az EGER KEZ.PAS unit néhány, a bemutatott funkciókat felhasználó eljárást tartalmaz, amelyek megkönnyítik az egér kezelését. A hívott KOZOS.PAS unit értelmezi a regiszter típusú "reg" változót és a "lap" procedúrát. A vizuális billentyűre, ablakra vagy korongrakattintást kezelő eljárások állandói a további programegységekben értelmezett vizuális objektumok méreteinek felelnek meg.

```
{$N+}
unit eger_kez;

interface

uses
  graph, dos, crt, kozos;

var
  ev: byte;

procedure eger_k;
procedure eger_1; {egér megjelenik}
procedure eger_n; {egér eltűnik)
procedure eger_t(xl,y1,x2,y2: integer); {egér mozgástere}
procedure eger_h(x,y: integer); {egérhelye}
procedure katt(xl,yl,x2,y2: integer; var ny: byte);
procedure kattint(x,y: integer; vi,v2: char; var vl: char);
procedure gb_katt(x,y: integer; var ny: byte);
procedure kp katt(x,y: integer; var ny: byte);

implementation

procedure eger_k;
begin
  lap(208,5); setcolor(0);
  with reg do begin
    ax:=0; intr($33,reg);
    if ax=0 then begin
      outtextxy(250,24,'Egér nem müködik'); ev:=0;
    end else begin
      outtextxy(250,24,'Egér müködik'); ev:=1;
      ax:=7; cx:=2; dx:=638; intr($33,reg);
      ax:=8; cx:=2; dx:=478; intr($33,reg);
      ax:=4; cx:=2; dx:=2; intr($33,reg);
    end;
```

```
    delay(1000); lap(208,5); ax:=1; intr($33,reg)
   end;
end;

procedure eger_1;
begin
   reg.ax:=1; intr($33,reg);
end;

procedure eger_n;
begin
   reg.ax:=2; intr($33,reg);
end;

procedure eger_t(xl,y1,x2,y2: integer); {egér mozgástere}
begin
   with reg do begin
    ax:=7; cx:=x1; dx:=x2; intr($33,reg);
    ax:=8; cx:=yi; dx:=y2; intr($33,reg);
   end;
end;

procedure eger_h(x,y: integer); {egér helye}
begin
   with reg do begin
    ax:=4; cx:=x; dx:=y; intr($33,reg);
   end;
end;

procedure katt(xl,yi,x2,y2: integer; var ny: byte);
                              {téglalapra való kattintás ellenőrzése}
begin
   ny:=0;
   with reg do
    if (cx>xl) and (cx<x2) and (dx>yi) and (dx<y2) then ny:=1;
end;
procedure kattint(x,y: integer; vl,v2: char; var vl: char);
            {kételemű lajstromból való választás ablakrakattintással}
var n: byte;
begin
   with reg do begin
    ax:=3; intr($33,reg);
    if bx=1 then begin
       katt(x+40,y+21,x+100,y+34,n); if n=1 then v1:=v1;
       katt(x+150,y+21,x+210,y+34,n); if n=1 then vl:=v2;
       bx:=0; eger_h(x+120,y+35);
    end;
   end;
   if keypressed then v1:=upcase(readkey);
end;
```

```
procedure gb_katt(x,y: integer; var ny: byte); {gombnyomás egérrel}
begin
  with reg do begin
     ax:=3; intr($33,reg);
     if bx=1 then katt (x, y, x+70, y+12, ny) ;
  end;
end;

procedure kp_katt(x,y: integer; var ny: byte);
                                  {korongrakattintás egérrel}
begin
  with reg do begin
     ax:=3; intr($33,reg);
     if bx=1 then begin
       if sqr(cx-x)+sqr(dx-y)<10 then ny:=1 else ny:=0;
     end;
  end;
end;

end.
```

1.4. Egérkurzor alakjának meghatározása grafikus módban

Erre a 33-as megszakítás 9-es funkciója szolgál. A kurzor maga két, egyenként 16x16 pontból álló bitmap, az úgynevezett képernyőmaszk és kurzormaszk, és az illető helyen levő háttérpontok színe közötti műveletek folytán jelenik meg, a következő művelettábla szerint:

Képernyőmaszk (XOR-maszk)	Kurzormaszk (AND -	Eredmény (képernyőpont színe)
0	0	0 (fekete)
0	1	15 (fehér)
1	0	c (a háttérpont színe)
1	1	15-c (a háttérpont kiegészítő színe)

Ezt a műveleti táblát szemlélteti az EG_KZ_K.PAS nevű program első része (az első billentyűleütésig).

A maszkok egyenként 16 *word* típusú állandóban (a 16-os számrendszerben felírt négyjegyű szám) tárolhatók. Egy-egy állandó a megfelelő bitmap egy-egy sorának pontjait tárolja. Egy 16-os számrendszerbeli szám minden számjegyének 4 darab 2-es számrendszerbeli - bináris - számjegy felel meg, ez összesen 16 bináris számjegy (0 vagy 1).

Ugyanabban a programban több kurzoralakkal is dolgozhatunk, amint azt az említett program második fele is szemlélteti, amelyben egy mosolygó arc "kacsint" a felhasználóra, azzal a szemével, amelyik egérgombot az utóbbi éppen lenyomja.

```
{$N+}

uses dos, crt, graph, grindhi, kozos, eger_kez;

type
  gr_kr = record
     ernyo, egerm: array[0..15] of word;
       {képernyőmaszk és egérmaszk (kurzormaszk)}
     kpx, kpy: integer; {aktív ("forró") pont helye a bitmap-ben}
  end;

const
  kurzor: gr_kr = {a maszkok szerepét szemléltető, "sakktábla" kurzor}
      (ernyo:($0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000,
         $FFFF, $FFFF, $FFFF, $FFFF, $FFFF, $FFFF, $FFFF, $FFFF) ;
         egerm:($00FF,$00FF,$00FF,$00FF,$00FF,$00FF,$00FF,$00FF,
         $00FF,$00FF,$00FF,$00FF,$00FF,$00FF,$00FF,$00FF); kpx: 8;
         kpy: 8);

  kurzor0: gr kr =          {mosolygó arc két nyitott szemmel}
      (ernyo:($FC3F,$FOOF,$E007,$C003,$8001,$8001,$0000,$0000,
            $0000,$0000,$8001,$8001,$C003,$E007,$FOOF,$FC3F);
      egerm:($0000,$03CO3$OFFO,$1FF8,$33CC,$2DB4,$6DB6,$73CE,
            $7FFE,$7FFE,$37EC,$3BDC,$1C38,$OFFO,$03CO3$0000);
      kpx: 8; kpy: 8);

  kurzorb: gr kr =          {mosolygó arc lehunyt bal szemmel}
      (ernyo:($FC3F,$FOOF,$E007,$C003,$8001,$8001,$0000,$0000,
            $0000,$0000,$8001,$8001,$C003,$E007,$FOOF,$FC3F);
      egerm:($0000,$03CO3$OFFO,$1FF8,$3FCC,$3FB4,$6DB6,$73CE,
            $7FFE,$7FFE,$37EC,$3BDC,$1C38,$OFFO,$03CO3$0000);
      kpx: 8; kpy: 8);

  kurzorj: gr_kr =          {mosolygó arc lehunyt jobb szemmel}
      (ernyo:($FC3F,$FOOF,$E007,$C003,$8001,$8001,$0000,$0000,
            $0000,$0000,$8001,$8001,$C003,$E007,$FOOF,$FC3F);
      egerm:($0000,$03CO3$OFFO,$1FF8,$33FC,$2DFC,$6DB6,$73CE,
            $7FFE,$7FFE,$37EC,$3BDC,$1C38,$OFFO,$03CO3$0000);
      kpx: 8; kpy: 8);
```

```
   kurzor2: grkr =          {mosolygó arc két lehunyt szemmel}
     (ernyo:($FC3F,$FOOF,$E007,$C003,$8001,$8001,$0000,$0000,
             $0000,$0000,$8001,$8001,$C003,$E007,$FOOF,$FC3F);
     egerm:($0000,$03CO3$OFFO,$1FF8,$3FFC,$3FFC,$6DB6,$73CE,
             $7FFE,$7FFE,$37EC,$3BDC,$1C38,$OFFO,$03CO3$0000);
     kpx: 8; kpy: 8);

var
   k, 1, b0: byte;
   rega: registers;

procedure kurz_alakkurzor: gr_kr); (kurzor alakjának váltortatása)
begin
   with kurzor do
     with rega do begin
       ax: =9;
       bx:=kpx;
       cx:=kpy;
       dx:=ofs(ernyo);
       es:=seg(ernyo);
     end;
   intr($33,rega);
end;

begin
   grafind; eger_k; eger_n;
   for 1:=0 to 12 do {háttérmintázat}
     for k:=0 to 16 do begin
       setfillstyle(1,k*1 mod 17 mod 16);
       bar(k*40-40,1*40-40,k*40,1*40);
     end;
   kurz_alak(kurzor); eger-h(320,240); eger_1;
   readln; eger_n;
   for 1:=0 to 120 do (háttérmintázat}
     for k:=0 to 160 do begin
           setfillstyle(1,k*1 mod 163 mod 16); bar(k*4,1*4,k*4+4,1*4+4);
end;
   kurz_alak(kurzor0); eger_h(320,240); eger_1;
   repeat
     with reg do begin
       ax:=3; intr($33,reg);
       if bx<>b0 then
         case bx of
         0:kurz_alak(kurzor0);
         1:kurz_alak(kurzorb);
         2: kurzalak(kurzorj);
         3:kurz_alak(kurzor2);
         end;
       b0 : =bx;
     end;
   until keypressed;
   closegraph;
end.
```

1.5. Interaktív kommunikáció

A továbbiakban az interaktív kommunikációnak a számítógépes grafikában használatos elemeit és ezek grafikus módban - billentyűzet vagy egér segítségével - való megvalósítását mutatjuk be. Lényeges jellemzőjük az egyértelműség, azaz bármely, a billentyűzeten illetve egérrel végbement eseménynek az adott helyzetben egyértelműen meg kell feleltetni egy parancsot. Ugyanakkor az egyértelműség általában nem kölcsönös, előfordulhat, hogy ugyanazt a parancsot különböző módon adhatjuk meg (például billentyű lenyomásával vagy egérrel is).

1.5.1. Képernyőpont meghatározása

Történhet billentyűzetről - grafikus kurzor segítségével - úgy, hogy a grafikus kurzort mozgatjuk a nyíl billentyűk segítségével, míg a kívánt helyre kerül, akkor leütjük az "Enter"-t. Ennél gyorsabb a képernyőkoordináták számszerű megadása. Legkényelmesebb azonban az egérrel rákattintani a kívánt pontra. Viszonylag jó felbontású képernyő esetén, amikor a képernyőpontok "aprók", azaz két különböző egymásmelletti pont között a távolság kicsi, szükség lehet egy visszajelzésre, vagyis az egérkurzor helyének kiírására, különösen akkor, ha a környező képernyőpontok ugyanolyan színűek, mint a keresett pont (az ábra részletei nem teszik lehetővé a kívánt pont azonosítását).

Ide tartozik még a grafikus kurzorok vagy más grafikus objektumok képernyőn való mozgatása, abban az értelemben, hogy ezek mindenkori új helyzetét a nyíl billentyűk sorozatos leütésével (relatív elmozdulás) vagy az egér megfelelő mozgótásával adjuk meg. Ezeket a lehetőségeket mutatják be a már ismertetett GR_KZ X.PAS és GR_KZ_O.PAS programok.

1.5.2. Adatbevitel

Történhet billentyűzet segítségével úgy, hogy egy számokból, esetenként előjelből és tizedespontból is álló karaktersort (string) ütünk le *(readkey* utasítás), majd átalakítjuk egész vagy valós számmá *(val* utasítás). Meg kell valósítani a leütött karakterek kiírását és lehetővé kell tenni a beírt érték változtatását is. Következésképpen szükség van egy billentyűre, amely letörli az utolsó karaktert (pld. a "Backspace") és egyre, amellyel véglegesítjük a beírt értéket (pld. az

"Enter"). Az alábbi, KOZOS.PAS nevű unit tartalmazza az *iras* procedúrát, amely a karaktersor bevitelét valósítja meg a felsorolt szempontok szerint.

```
{$N+}
unit kozos;

interface   •

uses graph, dos, crt;

type
  kars = set of char;

const
  betu: kars = ['A'...'Z','0'..'9','-','.'];
  b1:real=1; b2:real=0;
                                    {görbék β-spline közelítéséhez}
  blu:real=1; b2u:real=0;           (felületek β-spline közelítéséhez}
  blv:real=1; b2v:real=0;
  ku:real=0.5; kv:real=0.5;         {felületek helyi ellenőrzésű spline
                                     közelítéséhez}

type
   pont = object (síkbeli pont objektum}
     xp, yp: integer;
     procedure uj(ux, uy: integer);
   end;

var
   v, qe, qf, qr, qm, qk, qu, qa, qj, qd, elj, obj: char;
   sr: searchrec;
   x, y, z, hk, sz, szx, szy, k, 1: integer;
   fe, ko: byte;
   kn: string[12];
   reg: registers; (egérkezeléshez)
   k0: real;
procedure kkeret(xa, ya, xb, yb, ka, kb: integer) ;
procedure bkeret(xa, ya, xb, yb, ka, kb: integer) ;
procedure lap(x,y: integer);
procedure ablak(xl,yl,x2,y2: integer);
procedure iras(lm, xi, yi: integer);
procedure gomb(x,y: integer; kbet: char; szov: string; nv: byte);
procedure keret(x0, y0: integer);
procedure eltl;
procedure fesz;
procedure param;
implementation
```

```
procedure pont.uj(ux, uy: integer);
begin
   xp:=ux;
   yp :
=uy; end;

procedure kkeret(xa, ya, xb, yb, ka, kb: integer);
   {térhatású külső keret}
var
   kr: array [ 1 . . 4, 1. . 4 ] of pont;
begin
   kr[1,1].uj(xa,ya); kr[1,2].uj(xa-ka,ya-kb); kr[1,3].uj(xa-ka,yb+kb);
   kr[1,4].uj(xa,yb); kr[2,1].uj(xa,ya); kr[2,2].uj(xb,ya);
   kr[2,3] .uj (xb+ka,ya-kb); kr[2,4] .uj (xa-ka,ya-kb); kr[3,1] .uj (xa,yb);
   kr[3,2].uj(xb,yb); kr[3,3].uj(xb+ka,yb+kb); kr[3,4].uj(xa-ka,yb+kb);
   kr[4,1] .uj (xb,ya); kr[4,2] .uj (xb+ka,ya-kb); kr[4,3] .uj (xb+ka,yb+kb);
   kr[4,4].uj(xb,yb); setcolor(0);
   setfillstyle(1,3); fillpoly(4,kr[4]);
   setfillstyle(1,15); fillpoly(4,kr[1]);
   setfillstyle(1,15); fillpoly(4,kr[2]);
   setfillstyle(1,3); fillpoly(4,kr[3]);
end;

procedure bkeret(xa, ya, xb, yb, ka, kb: integer);
   {térhatású belső keret)
var
   kr: array[1..4,1..4] of pont;
begin
   kr[1,1].uj(xa,ya);    kr[1,2].uj(xa+ka,ya+kb);    kr[1,3].uj(xa+ka,yb-kb);
   kr[1,4].uj(xa,yb);  kr[2,1].uj(xa,ya);  kr[2,2].uj(xb,ya);  kr[2,3].uj(xb-
   ka,ya+kb); kr[2,4].uj(xa+ka,ya+kb); kr[3,1].uj(xa,yb); kr[3,2].uj(xb,yb);
   kr[3,3].uj(xb-ka,yb-kb);  kr[3,4].uj(xa+ka,yb-kb);  kr[4,1]  .uj  (xb,ya);
   kr[4,2] .uj (xb-ka,ya+kb); kr[4,3] .uj (xb-ka,yb-kb);
   kr[4,4].uj(xb,yb); setcolor(0);
   setfillstyle(1,15); fillpoly(4,kr[4]);
   setfillstyle(1,3); fillpoly(4,kr[1]);
   setfillstyle(1,3); fillpoly(4,kr[2]);
   setfillstyle(1,15); fillpoly(4,kr[3]);
end;

procedure lap(x,y: integer); {kommunikációs lap}
begin
   setfillstyle(1,11); bar(x,y,x+240,y+40);
   kkeret (x, y, x+240, y+41, 3, 3) ;
end;

procedure ablak(xl,yl,x2,y2: integer); {kommunikációs ablak}
begin
   setfillstyle(1,0); bar(xl,yl,x2,y2); bkeret(xl,yl,x2,y2+1,2,2);
end;
```

```
procedure iras(lm, xi, yi: integer);
     {karaktersor beírása koordinátameghatározás céljából}
begin
   setfillstyle(1,0); setcolor(15);
   repeat
     qr:=upcase(readkey);
     if qr<>chr(13) then
      if (qr=chr(8)) and (1>1) then begin {utolsó karakter törlése}
         bar(xi+(1-1)*8,yi,xi+1*8,yi+8); delete(kn,1-1,1); 1:=1-1;
      end else if 1<lm then if qr in betu then begin {karakterbeírás}
         kn:=kn+qr; outtextxy(xi+1*8,yi+1,qr); 1:=1+1;
      end;
   until (qr=chr(13)) or (qr=chr(27)); {kilépés "Enter" vagy "ESC"
                                        leütésével}

end;

procedure gomb(x,y: integer; kbet: char; szov: string; nv: byte);
     {vizuális billentyű (gomb) megjelenítése}
begin
   case nv of
   0:kkeret(x,y,x+70,y+12,1,1); {elengedve}
   1:bkeret(x-1,y-1,x+71,y+13,1,1); {lenyomva}
   end;
   setcolor(12); outtextxy(x+4,y+2,kbet); {aktív kezdőbetű}
   setcolor(0); outtextxy(x+12,y+2,szov); {név többi betűje}
end;

procedure keret(x0, y0: integer); {aktív szerkesztési mező}
begin
   setfillstyle(1,11); bar(0,0,640,480);
  bkeret(x0-6,y0-406,x0+606, y0+6,4,4);
  end;

procedure eltl; {új eltolásérték megadása)
begin
   repeat
     lap(208,5); setcolor(0); outtextxy(232,26,'Eltolás:');
     ablak(330,19,375,32); kn:= "; 1:=1; iras(5,330,21); val(kn,bl,hk);
   until hk=0;
end;

procedure fesz; {új feszültségérték megadása)
begin
   repeat
     lap(208,5); setcolor(0); outtextxy(232,26,'Feszültség:');
     ablak(330,19,375,32); kn:= "; 1:=1; iras(5,330,21); val(kn,b2,hk);
   until hk=0;
end;
```

```
procedure param;
   (érintőhossz-együttható megadása helyi ellenőrzésű spline
   interpolációhoz)
begin
   repeat
    lap(208,5); setcolor(0); outtextxy(232,26,'Együttható:');
    ablak (330, 19, 375, 32) ; kn:=" ; 1:=1; iras (5, 330, 21) ; val (kn, k0, hk) ;
   until hk=0;
   if k0>1 then k0:=1;
   if k0<0 then k0:=0;
end;

end.
```

Egy másik lehetőség egy virtuális eszköz, a potenciométer értelmezése és használata. Ez arra szolgál, hogy adott változó értékét növelni vagy csökkenteni tudjuk kurzorának mozgatásával. A kurzormozgatás történhet billentyűzetről vagy egérrel (ha a képernyőn látható és létezik a megfelelő kezelési eljárás).

A VJS.PAS program a "joystick" nevű eszközt modellezi. A képernyő jobb alsó sarkában levő négyzet alakú mezőben megjelenik a "fogantyú", amelyet az egérrel megfoghatunk és az egér bal gombját lenyomva mozgathatunk. Ha az egérgombot felengedjük, a fogantyú visszaugrik a négyzet közepére. A program egyébként mozgástani modell, a fogantyúval egy mozgó pont (körlap) gyorsulásának vízszintes és függőleges összetevőit szabályozzuk. A pont egy négyzet alakú pályán mozog, a mindenkori bemenő gyorsulásnak megfelelően módosul a sebessége és helyzete, a pálya szélével pedig rugalmasan ütközik. Egyszerű véges differenciás közelítést használva, minden időlépés után:

$$\begin{cases} v_x = v_{x0} + a_x\tau \\ v_y = v_{y0} + a_y\tau \end{cases} \quad \begin{cases} x = x_0 + \left(v_{x0} + v_x\right)\tau \ / \ 2 \\ y = y_0 + \left(v_{y0} + v_y\right)\tau \ / \ 2 \end{cases} \tag{1.1}$$

ahol T az időlépés, a 0 index pedig a hely és sebesség időlépés kezdetén való értékeit jelenti. A pálya falaival való ütközés a sebességnek a falra merőleges összetevőjének előjelét változtatja meg.

```
($N+)

uses
   crt, dos, graph, grafind, eger kez, kozos;

const
   a1=50; a2=470; a3=260; b1=20; b2=440; b3=230; tau=5e-5; r=10;
```

```
p e
  kor = object(pont)
     xr, yr: integer;
     fg, sz: byte;
     q: pointer; {bitmap heap-bell tárolási címére mutat}
     procedure koord(x0, y0: integer);
     procedure megj;
     procedure eltn;
  end;


     b: kor; {a - gyorsulást módosító "fogantyú", b - mozgó pont}
      yl, x2, y2, x3, y3: integer;
  ze: longint;
  ks: string;
     vx, vy, vx0, vy0, ux, uy: real;
     . word;

procedure kor.koord(x0, y0: integer); {középpont képernyőkoordinátái}
begin
  xr:=x0+xp;
  .r:=y0-yp;
end;

procedure kor.megj; {körlap megjelenik}
begin
  =etimage (xr-r, yr-r, xr+r, yr+r, q^ ) ;
  setfillstyle (1, sz) ; setcolor (0) ; fillellipse (xr, yr, r, r) ;
end;

procedure kor.eltn; {körlap eltűnik}
begin
  putimage(xr-r,yr-r,q^,normalput);
end;

procedure eger_f; {megfogás)
begin
  with reg do begin
     ax:=3; intr($33,reg);
     if (bx=1) and (getpixel(cx,dx)=a.sz) and (a.fg=0) then a.fg:=1;
  end;
end;

procedure eger_e; {elengedés}
begin
  with a do
     with reg do begin
      ax:=3; intr($33,reg);
      if (bx=0) and (fg=1) then begin
         fg : =0;
         eger_n; eltn; uj(0,0); koord(x3,y3); megj; eger_l; end;
           {az elengedett "fogantyú" visszaugrik nyugalmi helyzetébe}
  end;
end;
```

```
procedure eger_v; (egér viszi a körlapot)
begin
  with a do
    with reg do begin
      ax:=3; intr($33,reg);
      if (fg=1) and (sqr(cx-xr)+sqr(dx-yr)>2) then begin
            eger_n; eltn; uj(cx-x3,y3-dx); koord(x3,y3); megj; eger 1;
end;
    end;
end;

begin
  grindhi; mm:=imagesize(0,0,2*r,2*r); getmem(a.q,mm); getmem(b.q,mm);
  ux:=0; uy:=0;
  eger k; eger_n; cleardevice;
  setfillstyle(1,11); bar(0,0,getmaxx,getmaxy);
  x1:=getmaxx-110; y1:=getmaxy-110;
  x2:=getmaxx-10; y2:=getmaxy-10;
  x3:=getmaxx-60; y3:=getmaxy-60;
  ablak(a1-2,b1-2,a2+2,b2+2); ablak(x1-2,y1-2,x2+2,y2+2);
  setcolor(7);
  for k:=-4 to 4 do begin
    line(x1,y3+10*k,x2,y3+10*k); line(x3+10*k,y1,x3+10*k,y2);
    line(a1,b3+50*k,a2,b3+50*k); line(a3+50*k,b1,a3+50*k,b2);
  end;
  getimage(x3-r,y3-r,x3+r,y3+r,a.q^);
  getimage(a3-r,b3-r,a3+r,b3+r,b.q^);
  setcolor(0);
  for k:=-4 to 4 do begin
    line(x1-10,y3+10*k,x1,y3+10*k); line(x3+10*k,y1-10,x3+10*k,y1);
    line(a1-10,b3+50*k,a1,b3+50*k); line(a3+50*k,b2,a3+50*k,b2+10);
    str(k*10:3,ks);
    if not odd(k) then outtextxy(x1-35,y3+10*k-3,ks);
    if k mod 4 = 0 then outtextxy(x3+11*k-18,y1-20,ks);
    str(k*50:4,ks);
    outtextxy(a1-50,b3+50*k-3,ks);
    outtextxy(a3+50*k-20,b2+15,ks);
  end;
  a.uj(0,0); a.sz:=12; a.koord(x3,y3); a.megj; a.fg:=0;
  b.uj(0,0); b.sz:=10; b.koord(a3,b3); b.megj; b.fg:=0;
  eger_t(x1+r,y1+r,x2-r,y2-r); eger_h(x3,y3); eger_1;
  outtextxy(500,12,'Idö:'); outtextxy(625,12,'s');
  ablak(560,5,620,25);
  outtextxy(500,45,'Hely:'); ablak(500,80,550,100);
  ablak(570,80,620,100);
  outtextxy(507,65,'x       y'); outtextxy(520,110, 'pixel');
  outtextxy(500,135,'Sebesség:');
  ablak(500,170,550,190);     ablak(570,170,620,190);
  outtextxy(500,150,'v     v'); outtextxy(507,155,'x   y');
  outtextxy(520,200, 'pixel / s');
  outtextxy(500,225,'Gyorsulás:'); ablak(500,260,550,280);
  ablak(570,260,620,280);
```

```
  outtextxy(500,240,'a        a'); outtextxy(507,245,'x        y');
  outtextxy(520,290, 'pixel / s»);
  t:=0; te:=0; vx0:=0; vy0:=0;
  repeat
    t:=t+tau;
    eger_f; eger_v; eger_e; {"fogantyú" kezelése}
    vx:=vx0+a.xp*tau; vy:=vy0+a.yp*tau; {sebesség kiszámítása}
    if trunc(6*(t+tau))>trunc(6*t) then begin {10 másodpercenkénti
                                               kiírás}
      if trunc(t)>te then te:=te+1;
      setfillstyle(1,0);
      bar(563,8,617,22);
      bar(503,83,547,97); bar(573,83,617,97);
      bar(503,173,547,187); bar(573,173,617,187);
      bar(503,263,547,277); bar(573,263,617,277);
      setcolor(15); str(te:7,ks); outtextxy(562,12,ks);
      str(b.xp:5,ks); outtextxy(504,87,ks); {hely kiírása}
      str(b.yp:5,ks); outtextxy(574,87,ks);
      str(vx:5:0,ks); outtextxy(504,177,ks); {sebesség kiírása}
      str(vy:5:0,ks); outtextxy(574,177,ks);
      str(a.xp:5,ks); outtextxy(504,267,ks); {gyorsulás kiírása}
      str(a.yp:5,ks); outtextxy(574,267,ks);
    end;
    ux:=ux+(vx0+vx)*tau; uy:=uy+(vy0+vy)*tau; {elmozdulás kiszámítása}
    if (ux>200) or (ux<-200) then vx:=-vx; {ütközések kezelése}
    if (uy>200) or (uy<-200) then vy:=-vy;
    if sqr(ux-b.xp)+sqr(uy-b.yp)>2 then begin {mozgó pont
                                               megjelenítése}
      b.uj(round(ux),round(uy)); b.eltn; b.koord(a3,b3); b.megj; end;
    vx0:=vx; vy0:=vy;
  until keypressed;
end.
```

1.5.3. Lajstromelem kiválasztása

A parancsközlés fontos eszköze. Természetesen léteznie kell a lajstromnak, amely utasítások, eljárások vagy vizuális objektumok rendezett halmaza, és a lajstromelemek valamilyen azonosítójának, amelynek alapján ki lehet választani. Az azonosító lehet sorszám, "forró" betű, vizuális billentyű vagy ikon. A kiválasztás az azonosításnak megfelelően több módon történhet:

- sorszám beütésével
- a megfelelő "forró" betű leütésével
- egérrel történő billentyűre vagy ikonra kattintással

- "Enter" leütésével, amennyiben létezik megfelelően jelzett prezumtív (előre kijelölt) kiválasztás, amely bizonyos (például "Tab") billentyű leütésével változtatható

Amennyiben a kiválasztás eredménye később jelentkezik, vagy nem egyértelmű, a kiválasztás tényét a kiválasztott elem megjelenítésének megváltoztatásával jelezni kell.

A GOMB KEZ.PAS nevű unit az interaktív görbe- illetve felületszerkesztési programok számára teszi lehetővé vizuális billentyűk kezelését, egérrel való rákattintással vagy (a gomb nevében más színnel írt) forró betű leütésével. A KOZOS.PAS unit *gomb* eljárása térhatású vizuális billentyűt jelenít meg, esetenként lenyomott vagy felengedett állapotban.

```
{$N+}
unit gomb kez; {vizuális billentyűk kezelése}

interface
uses graph, crt, kozos, eger_kez;

const
  gb: array [ 1..10 ] of byte = (0,0,0,0,0,0,0,0,0,0);

procedure parancs(pr: byte);
procedure par_katt(pr: byte; var vl: char);
procedure eljaras;
procedure elj_katt(var vl: char);
function gombny(valasz: kars): boolean;
procedure bvalt;

implementation

procedure parancs(pr: byte);
                {vizuális billentyűk meghatározása parancsközléshez}
begin
  gomb(460,4,'V','álaszt',gb[1]); gomb(550,38,'Q','-
  kilép',gb[4]); gomb(460,38,'E','redmény',gb[2]);

  case pr of
  1,3,4:begin
      gomb(460,21,'B','eszúr',gb[6]); gomb(550,21,'T','öröl',gb[5]);
      gomb(550,4,'G','örbe',gb[3]);
    end;
  2:begin
      gomb(460,21,'F','elület',gb[3]);
      if obj='F' then begin {β-spline felületekhez}
         setfillstyle(1,11);
```

```
      bar(80,465,170,479); bar(220,465,310,479);
      bar.(360, 465, 470, 479) ; bar (520, 465, 640, 479) ;
      setcolor(0);     .
      str(blu:3:2,kn); outtextxy(2,470,'Eltolás_u = '+kn);
      str(b2u:2:1,kn); outtextxy(292,470,'Feszültség_u = '+kn);
      str(biv:3:2,kn); outtextxy(145,470,'Eltolás_v = '+kn);
      str(b2v:2:i,kn); outtextxy(471,470,'Feszültség_v = '+kn);
    end;
    if obj='H' then begin {helyi helyi ellenőrzésű spline
                           felületekhez)
      setfillstyle(1,11); bar(100,465,570,479); setcolor(0);
      str(ku:3:2,kn); outtextxy(120,470,'k0_u = '+kn);
      str(kv:3:2,kn); outtextxy(340,470,'k0_v = '+kn); end;
  end;
end;
case pr of
1:begin
    if gb[3]=1 then setcolor(13) else setcolor(0);
    outtextxy(28,5,'Interpolációs eljárás');
  end;
3:begin
    setcolor(0);
    outtextxy(20,20,'COONS INTERPOLACIO');
    outtextxy(134,10,', ,');
  end;
4:begin
    setfillstyle(1,11); bar(75,2,155,17); bar(110,37,200,52);
    setcolor(0);
    str(b1:3:2,kn); outtextxy(20,5,'Eltolás = '+kn);
    str(b2:3:2,kn); outtextxy(35,40,'Feszültség = '+kn);
    end;
end;
procedure par_katt(pr: byte; var vl: char); {parancs kiválasztása}
var fn: byte;

gb_katt(460,4,fn); if fn=1 then vl:='V'; {egérrel}
gb katt(460,38,fn); if fn=1 then v1:='E'; gb
gb_katt(550,38,fn); if fn=1 then vl:='Q';

case pr of
1:begin
    gb_katt(460,21,fn); if fn=1 then vl:='B';
    gb_katt(550,4,fn); if fn=1 then vl:='G';
    gb_katt(550,21,fn); if fn=1 then vl:='T';
    gb_katt(30,21,fn); if fn=1 then v1:='A'; { β-spline görbékhez}
    gb katt(120,21,fn); if fn=1 then v1:='C';
  end;
2:begin
    gb katt(460,21,fn); if fn=1 then vl:='F';
```

```
        gb_katt(550,4,fn); if fn=1 then v1:='A'; {β-spline felületekhez}
       gb_katt(550,21,fn);if fn=1 then v1:='C'; {β-spline és helyi
                                    ellenőrzésű spline felületekhez)
     end;
   end;
   if keypressed then v1:=upcase(readkey); {billentyűzetről}
end;

procedure eljaras;
     (vizuális billentyűk meghatározása interpolációs
     eljárás kiválasztásához)
begin
   gomb(30,21,'1',' Coons',gb[7]); gomb(30,38,'2',' Bézier',gb[8]);
  gomb(120,21,'3',' Spline',gb[9]); gomb(120,38,'4',' B-spl.',gb[10]);
  end;

procedure elj_katt(var v1: char); {interpolációs eljárás kiválasztása}
var fn: byte;
begin
   gb_katt(30,21,fn); if fn=1 then v1:='1'; {egérrel}
   gb katt(30,38,fn); if fn=1 then v1:='2';
   gb katt(120,21,fn); if fn=1 then v1:='3';
   gb katt(120,38,fn); if fn=1 then v1:='4';
   if keypressed then v1:=upcase(readkey); {billentyűzetről}
end;

function gombny(valasz: kars): boolean; {gomblenyomás detektálása}
begin
   if reg.bx=1 then par_katt(1,gr);
   if qr in valasz then gombny:=true else gombny:=false;
end;

procedure bvalt;
     (vizuális billentyűk meghatározása (3-spline paraméter
     kiválasztásához)
begin
   case obj of
   'G':begin {görbe}
       gomb(30,21,'A','- elt.',gb[7]); gomb(120,21,'C','- fesz.',gb[8]);
end;
   'F':begin {felület}
       gomb(550,4,'A','- elt.',gb[7]); gomb(550,21,'C','- fesz.',gb[8]);
end;
   'H':begin (felület)
     {vizuális billentyű helyi ellenőrzésű spline paraméter
     kiválasztásához}
     gomb(550,21,'C','- k0',gb[8]); end;
   end;
  end;
end;

end.
```

1.5.4. Tárolás, beolvasás

Bonyolult szerkesztési feladatokat megoldó vagy nagy mennyiségű számítást végző programok esetében, amikor a konfiguráció betáplálása és módosítása sok figyelmet és időt igényel, illetve a program futása hosszadalmas, fontos lehetővé tenni a konfiguráció illetve az eredmények tárolását más alkalommal való felhasználás vagy más programmal végzett eredmény-feldolgozás céljából. A Turbo-Pascal *findfirst* utasítása lehetővé teszi megadott nevű file létezésének ellenőrzését, így elkerülhető a futási hiba, ha nem létező file-t próbálunk olvasni, illetve az ugyancsak bosszantó incidens, hogy már létező file-ra, amelyet nem szándékozunk letörölni, ugyanazon néven ráírjunk egy újat.

Ezeknek a követelményeknek tesz eleget a FILE_KEZ.PAS unit, amely az olvasási és tárolási eljárásokat készíti elő.

```
  {$N+}
unit file kez;

interface

uses dos, crt, graph, kozos, kerdesek;

procedure olvas(ext: string);
procedure ment(ext: string);

implementation

procedure ment(ext: string);
begin
  v:='I';
  repeat
    lap(200,200); setcolor(0); outtextxy(262,208,'File neve ?');
    ablak(270,219,350,232);
    setfillstyle(1,0); outtextxy(350,222,'.'+ext); setcolor(15);
    kn:=''; 1:=1; iras(9,270,221);
  until 1>1;
  kn:=kn+'.'+ext;
  findfirst(kn,0,sr); (file létezésének ellenőrzése)
  if doserror=0 then rair; (ráírási jóváhagyás lekérdezése)
end;

procedure olvas(ext: string);
begin
  lap(200,200); setcolor(0); outtextxy(262,208,'File neve
  ?'); ablak(270,219,360,232);
  setcolor(0); outtextxy(364,222,'.'+ext); kn:= ''; k:=1;
  setcolor(15); setfillstyle(1,0);
```

```
  kn:=''; 1:=1; iras(9,270,222);
  kn:=kn+'.'+ext;
  findfirst(kn,0,sr); {file létezésének ellenőrzése}
  if doserror<>0 then begin
    lap(200,200); setcolor(0);
         outtextxy(260,220,'Nincs ilyen file'); ko:=0; delay(2000);
lap (200, 200) ;
  end else ko:=1;
end;

end.
```

A KERDESEK.PAS unit tartalmazza a *rair* procedúrát néhány, az interpolációs programok által hívott, kommunikációs eljárás mellett. A KOZOS.PAS unit *kattint* eljárása teszi lehetővé a válaszadást egérrel (a választ tartalmazó ablakra kattintva) vagy billentyűzetről (a válasz forró betűjét leütve).

```
{$N+)
unit kerdesek;

interface

uses crt, graph, kozos, eger kez;

procedure kerdez(x,y: integer; szov, kdl, kd2: string; kbl, kb2:
char) ;
procedure vege;
procedure rair;
procedure ujkonf;
procedure alak;
procedure megj;
procedure kpl;
procedure b_spline;
procedure vlt;
procedure kp_er;
procedure evlt;
procedure ujra;
procedure irany;
procedure egyutth;

implementation

procedure kerdez(x,y: integer; szov, kdl, kd2: string; kbl, kb2: char)
;
  {kételemű lajstromból való választás előkészítése}
begin
  lap(x,y); setcolor(0); outtextxy(x+7,y+8,szov);
  ablak(x+40,y+21,x+104,y+34); ablak(x+150,y+21,x+21.0,y+34);
  setcolor(12); outtextxy(x+45,y+24,kb1); outtextxy(x+155,y+24,kb2);
```

```
  setcolor(15); outtextxy(x+53,y+24,kd1); outtextxy(x+163,y+24,kd2);
  eger_t(x,y,x+240,y+40); eger_l;
end;

procedure vege; {kilépés eldöntése)
begin
  qk:=' ';
  kerdez(200,200,'      Vége a munkának ?','gen' 'em' 'I' 'N');
  repeat kattint (200, 200, ' I' , 'N' , qk) ; until (qk='I') or
  (qk='N') ; eger_n;
end;

procedure rair; {létező file-ra való ráírás eldöntése}
begin
  v:='
  kerdez(200,200,kn+' létezik. Ráírni?','gen','em','I','N');
  repeat kattint(200,200,'I','N',v); until (v='I') or (v='N');
  eger_n;
end;

procedure ujkonf; {konfigurációváltoztatás eldöntése}
begin
  qu:=    ;
  kerdez(200,200,'      Uj konfiguráció ?','gen','em','I','N');
  repeat kattint(200,200,'I','N',qu); until (qu='I') or (qu='N')
  eger_n;
end;

procedure alak; (nyitott/zárt görbealak megadása)
begin
  qa:='
  kerdez(208,5,'      Görbe alakja ?','yitott','árt','N','Z');
  repeat kattint(208,5,'N','Z',qa); until (qa='N') or (qa='Z');
  eger_n; lap(208,5);
end;

procedure megj; {megjelenítési módszer
megadása} begin
  qj •=' ';
  kerdez(208,5,'      Megjelenitès ?','rótváz','ele','D','T');
  repeat kattint(208,5,'D','T',qj); until (qj='D') or (qj='T');
  eger_n; lap(208,5);
end;

procedure kpl;
begin
  qm:= ;
  kerdez(200,200,' Kontrollpontok látszanak ?','gen','em','I','N');
  egert(200,200,440,240); eger              ' '1; qm:=;
  repeat kattint(200,200,'','N7,qm); until (qm='I') or (qm='N');
  eger_n;
end;
```

```
procedure b_spline; {polinomok fokának megadása B-spline közelítésben)
begin
  qr:=' ';
  kerdez(200,200,'     Polinomok foka ?','','','2','3');
  repeat kattint(200,200,'2','3',qr); until (qr='2') or (qr='3');
  if qr='3' then elj:='5';
  eger_n;
end;

procedure vlt; {sorszám/hely változtatási szándék jelzése}
begin
  qr:=' ';
  kerdez(208,5,'          Mi  változik  ?','orszám','ely','S','H');
  repeat  kattint(208,5,'S','H',qr);  until  (qr='S')  or  (qr='H');
  eger_n;
end;

procedure kp_er; {kontrollpont hely/érintő változtatási szándék
                  jelzése Coons-Hermite interpoláció esetén}
begin
  qr:='
  kerdez(208,5,'         Mi változik ?','ont','rintö','P','E');
  repeat kattint(208,5,'P','E',qr); until (qr='P') or (qr='E');
  eger_n;
end;

procedure evlt; {érintővektor hossz/irányítás változtatási szándé
                 jelzése}
begin
  qe:=' ';
  k e r d e z ( 2 0 8 , 5 , '    M i   v á l t o z i k
     repeat kattint(208,5,'H','S',qe); until (qe='H') or (qe='S');
eger_n;
end;


procedure ujra; {önműködő érintőújraszámítás eldöntése)
begin

  kerdez(208,5,' Önmüködö érintöszámítás ?','gen','em','I','N');
  repeat kattint(208,5,'I','N',qm); until (qm='I') or (qm='N');
  eger_n; lap(208,5);
end;

procedure irany; {paraméterirány kiválasztása β-spline
                  alakegyütthatómódosításhoz}
begin
  qd:=' ';
  kerdez(208,5,'         Irány
  repeat kattint(208,5,'U','V',qd);
  until (qd='U') or (qd='V'); eger_n;
end;
```

```
procedure egyutth; (érintőhossz kiszámításához Coons-Hermite
                    interpolációban)
begin
   repeat
     lap(208,5); setcolor(0); outtextxy(222,23,'együttható = ');
     ablak(330,19,375,32); kn:= ''; 1:=1; iras(5,330,21); val(kn,k0,hk);
   until hk=0;
   if k0<-10 than k0:=-10; if kO>10 then k0:=10;
end;

end.
```

Nyelvtanulás

számítógéppel!

Nyelvoktató és oktató CD-k

széles választékban

	PICDIC Angol, Német, Francia lemezenként	6400,-
	Angol Kiejtésiskola	5400,-
	ClipDIC English 1, 2 lemezenként	6400,-
	ClipDIC Deutsch 1	6400,-
	Üzleti Angol 1	6400,-
ÚJ	ClipDIC Deutsch 2	6400,-
ÚJ	Üzleti Angol 2	6400,-

* Áraink az ÁFA-t nem tartalmazzák.

ÉRDEKLŐDÉSÜKET ÉS MEGRENDELÉSEIKET AZ ALÁBBI CÍMEKEN VÁRJUK:

ComputerBooks
1126 Budapest, Tartsay V. u. 12.
Tel: 1/175-1564, 175-3591
info@computerbooks.hu

Profi-Média Kft.

Profi-MédiaKft.
6500Baja, DériFrigyesstny4.
Tel./fax: 79/325-467
pmed@mail.matav.hu

2. Sima görbék interaktív előállítása

Számos műszaki terület egyik feladatköre bizonyos sorrendben elhelyezett pontokra görbéket vagy felületeket illeszteni. Ezen területek némelyike a művészetekkel határos (ipari formatervezés, építészet), és a görbék, felületek alakja esztétikai, ergonómiai szempontoknak is eleget kell tegyen. Más esetben a geometriai jellemzőknek műszaki szerepe van, például, ha egy mérési eredményhalmazt további feldolgozás során differenciálni kell, akkor ez a művelet tetemesen megnöveli a mérési "zajokat". Ez a zavaró körülmény elkerülhető, ha az adott ponthalmazra megfelelő eljárással közelítő sím görbét vagy felületet illesztünk.

2.1. Görbék illesztésének kérdései

Egy paraméteres előállítású síkgörbét:

$$\begin{cases} x = x(t) \\ y = y(t) \end{cases}; \qquad t \in I \subset \Re \quad , \tag{2.1}$$

simának nevezünk az I intervallumon, ha minden pontjában, azaz a t paraméternek minden, az intervallum belsejéhez tartozó értékére létezik folytonosan változó érintője. Ez azt jelenti, hogy az előállítási függvényeknek léteznek deriváltjai és ezek folytonosak, továbbá a két deriváltfüggvény egyidejűleg nem lehet O.

Használatos a görbék következő osztályozása is: azt mondjuk, hogy egy görbe k -ad rendű folytonos az I intervallumon, ha az intervallum minden pontjában létezik és folytonos az előállítási függvények első k deriváltja.

Az illesztés folytonosságának kérdése akkor merül fel, amikor egy görbe szakaszosan analitikus előállítású, vagyis előállítási függvényei több, páronként diszjunkt belsejű intervallumon különbözőek:

$$\begin{cases} x = x_k(t) \\ y = y_k(t) \end{cases}; \qquad t \in I_k \subset \Re\,;\ \bigcup_{k=1}^{n} I_k = I\,;\ \operatorname{card}\left\{ I_k \bigcap_{k \neq l} I_l \right\} \leq 1 \quad , \tag{2.2}$$

ahol *card* a halmaz számosságát jelenti, azaz csak a szomszédos intervallumoknak van közös pontjuk, éspedig a közös határpont.

Az illesztés folytonossága *k*-ad rendű az I_1 és I_2 intervallumokat elválasztó t_0 pontban, ha:

$$\begin{cases} x_1(t_0) = x_2(t_0) \\ y_1(t_0) = y_2(t_0) \end{cases} \quad \begin{cases} x_1^{(k)}(t_0) = x_2^{(k)}(t_0) \\ y_1^{(k)}(t_0) = y_2^{(k)}(t_0) \end{cases} \quad \begin{cases} x_1^{(k+1)}(t_0) \neq x_2^{(k+1)}(t_0) \\ y_1^{(k+1)}(t_0) \neq y_2^{(k+1)}(t_0) \end{cases} . \quad (2.3$$

Ha például egy görbe előállítási függvényei *n* -ed rendű polinomok, akkor az illeszkedési pontokban *n —1* -ed rendű folytonosság érhető el. Minél magasabb rendű az illesztések folytonossága, annál simább a kapott görbe. Vizuális szempontból a folytonosság rendje kettőig viszonylag jól megkülönböztethető (a 0-d rendű folytonos illesztési pontban szögpont található, az elsőrendűben a görbület hirtelen változik), ezért a harmadfokú polinomokkal való előállítás megkülönböztetett fontosságnak örvend, ugyanis a jó simaság mellett a viszonylag egyszerű előállítás előnyét is hordozza.

Térbeli görbék esetében a fenti képletek annyiban módosulnak, hogy még egy *(t)* koordinátának megfelelő előállítási függvényre van szükség:

$$\begin{cases} x = x(t) \\ y = y(t) ; \\ z = z(t) \end{cases} \qquad t \in I \subset \Re \quad . \qquad (2.4$$

Tömörebb a vektoros felírás, amelyet a következőkben használni fogunk:

$$\mathbf{r(t)} = r(x(t), y(t)) \qquad (2.5)$$

síkgörbék, illetve:

$$\mathbf{r(t)} = r(x(t), y(t), z(t)) \qquad (2.6)$$

térgörbék esetén.

2.2. Interaktív paraméteres előállítás

Analítikus paraméteres előállítást használva, az előállítási függvények együtthatóinak módosításával változtatható a görbe alakja. Két kérdés vetődik fel: bármely alak elérhető-e így, és ha igen, hogy lehet meghatározni az együtthatókat, hogy a kívánt alakú görbéhez jussunk. Az első kérdésre van igenlő válasz, ugyanis léteznek függvénycsaládok, amelyeknek tagjai megfelelő együtthatókkal összegezve, bármely függvényalakot bármilyen jól megközelítenek (például a trigonometrikus polinomok). A második kérdés kissé bonyolultabb, legalábbis a megvalósítása. Ha a görbe ismert, abban az értelemben, hogy elég sok pontja adott, akkor az előállítási függvényei meghatározhatóak, például Fourier-sorbafejtéssel.

Egy Fourier-sor általános alakja:

$$f(t) = a_0 + \sum_{k=1}^{\infty} a_k \sin(kt + \varphi_k);\ a_0 \in \Re,\ a_k \in [0,\infty),\ \varphi_k \in [0.2\pi]\ .\quad (2.7$$

Természetesen nem lehet végtelen számú taggal dolgozni. Ezért a tagok számát felülről korlátozni kell. Ezenkívül gondot okozhat az együtthatók kiszámítá- sához szükséges pontok meghatározása is.

A GB_P_2D.PAS nevű program interaktív alakmódosítást tesz lehetővé úgy, hogy az

$$\mathbf{r}(t) = \left(a_0 + \sum_{k=1}^{20} a_k \sin(kt + \pi p_k), b_0 + \sum_{k=1}^{20} b_k \sin(kt + \pi q_k) \right);$$
$$a_0, b_0 \in \Re,\ a_k, b_k \in [0,\infty),\ p_k, q_k \in [0.2],\ t \in [0,2\pi] \qquad (2.8$$

előállítású görbe együtthatóit külön-külön változtatni lehet. A módosítandó együtthatók a megfelelő ablakra egérrel való rákattintással választhatók ki, vagy úgy, hogy előbb lenyomjuk a "Választ" vizuális billentyűt (egérrel vagy a V billentyű leütésével), majd a megjelenő kommunikációs lapon megadjuk az ezvüttható nevét és sorszámát. Az új értéket az illető együttható értékét kijelző ablakba írjuk be, amelyből előbb eltűnik a régi érték. Minden együtthatómódosítást követően megjelenik az új értékeknek megfelelő görbealak.

A megjelenítés úgy történik, hogy a paraméter intervallumát felosztjuk és a görbének az intervallumosztópontoknak megfelelő pontjait összekötjük, ezzel a görbét a rajzolt húrokkal közelítve meg. Ha a felosztás elég finom (az osztópontoknak megfelelő görbepontok elég közel vannak egymáshoz), akkor a megfelelő húrok és görbeívek közötti különbség nem észlelhető.

```
($N+)

program gb p 2d; {síkgörbék interaktív paraméteres előállítása}

uses crt, dos, graph, grafind, kozos, kerdesek, eger_kez,

file_kez;

const
  nf=20;
  kx0=320; ky0=270;
  ehv: array[1..4] of char = ('a','p','b','q');
  xl: array[1..4,1..2] of word
      =((19,60), (66,115), (540,581), (587,636));
  yl1=70; y12=470;
  gb: array[1..2) of byte = (0,0);

type
  mt = array[1..4,0..nf) of real;

var
  ff: file of mt;
  i, j, lx:
  integer; a: mt;
  t, tl, t2:
  real;
  nv: char;

function x(t: real): real;
var w: real;
begin
  w:=a[1,0];
  for k:=1 to nf do
  w:=w+a[1,k]*sin(k*t+a[2,k]*pi);
  x:=w;
end;

function y(t: real): real;
var w: real;
begin
  w:=a[3,0];
  for k:=1 to nf do
    w:=w+a [3, k] *sin (k*t+a [4, k] *pi) ;
  y:=w;
end;
```

```
procedure megj; {görbe megjelenítése}
begin
  setfillstyle(1,0); bar(124,70,514,470);
  setlinestyle(0,0,3); setcolor(15);
  moveto(kx0+round(x(tl)),ky0-round(y(tl)));
  for 1:=1 to lx do begin
    t:=t1+1/lx*(t2-tl);
    lineto(kx0+rbund(x(t)),ky0-round(y(t))); end;
end;

procedure keret; (aktív szerkesztési mező)
var ks: string;
begin
  setfillstyle(1,11); bar(0,0,640,480); bkeret(120,66,519,475,4,4);
  for k:=0 to of do begin
    str(k,ks);
    if k<10 then begin
      outtextxy(8,94+k*18,ks); outtextxy(529,94+k*18,ks) end
            else begin outtextxy(1,94+k*18,ks); outtextxy(522,94+k*18,ks);
end;
  end;
end;

procedure fuggveny; {paraméteres előállítási képlet}
coast xk=10;
begin
  setcolor(0);
  outtextxy(xk+64,0,'n'); outtextxy(xk+48,20,'0 1 k        k');
  outtextxy(xk,10,'x(t)=a +Ea sin(kt+p 1)');
  outtextxy(xk+64,35,'n'); outtextxy(xk+48,55,'0 1 k         k');
outtextxy(xk,45,'y(t)=b +Eb sin(kt+q i)'); end;

procedure konfig;
begin
  repeat
    qf:=' ';
    kerdez(200,200,'        Konfiguráció ?','j','árolt','U','T');
    repeat kattint(200,200,'U','T',gf); until (qf='U') or (qf='T');
    if qf='T' then begin
      eger_n; olvas('GBP'); {tárolt együtthatók betöltése}
      if ko=1 then begin assign(ff,kn); reset(ff); read(ff,a);
                         close(ff);
                   end;
      eger_l;
    end else ko:=1;
  until ko=1;
end;
```

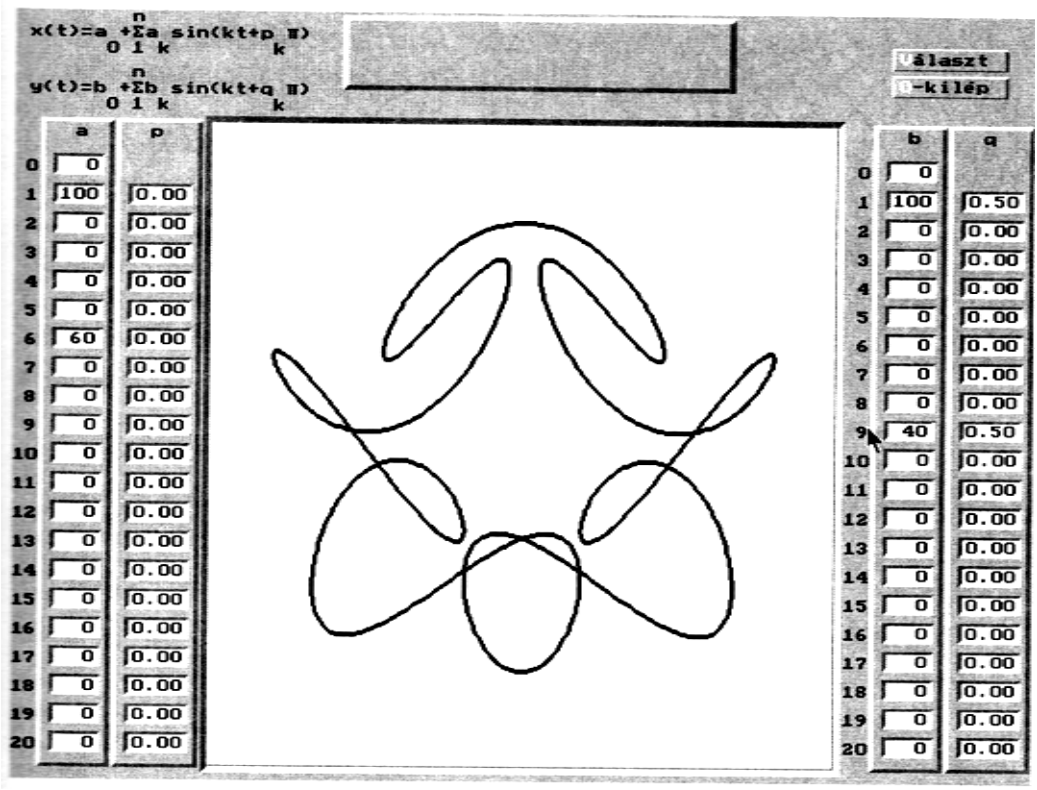
```
procedure kilep;
begin
      qm:=' '; kerdez(200,200,'                Menteni a konfigurációt
?', 'gen', 'em', ' I','N') ;
  repeat kattint(200,200,'I','N',qm); until (qm='I') or (qm='N');
  if qm='I' then begin
     eger n; ment('GBP'); {együtthatók tárolása}
     if v='I' then begin assign(ff,kn); rewrite(ff); write(ff,a);
     close(ff); end;
     eger_1;
  end;
end;

procedure parancs; (vizuális billentyűk meghatározása parancsközlés-
hez)
begin
      gomb(550,21,'V','álaszt',gb[i]); gomb(550,38,'Q','-kilép',gb[2]);
end;

procedure par_katt(var vl: char); {parancs kiválasztása}
var fn: byte;
begin
  gb_katt(550,21,fn); if fn=1 then v1:='V'; {egérrel}
  gb_katt(550,38,fn); if fn=1 then v1:='Q';
  if keypressed then v1:=upcase(readkey); {billentyűzetről}
end;

procedure e_lap(xi,y1,x2,y2: integer; nev: char);
                                  {együtthatók kijelzésére szolgáló lap}
var kl, h: byte; ks: string;
begin
  kkeret(xl,yi,x2,y2,2,2); outtextxy(x1+20,y1+2,nev);
  if (nev='a') or (nev='b') then begin h:=31; k1:=0 end
                            else begin h:=39; k1:=1; end;
  for k:=k1 to of do begin
     abiak(xi+6,yl+k*18+20,x1+6+h,yi+k*18+32); setcolor(15);
     case nev of
     'a':str(a[i,k]:3:0,ks);
     'p':str(a[2,k]:3:2,ks);
     'b':str(a[3,k]:3:0,ks);
     'q':str(a[4,k]:3:2,ks);
     end;
     outtextxy(x1+11,y1+k*18+23,ks);
  end;
end;

procedure egyutth; {együtthatók kiírása}
begin
  for j:=1 to 4 do e lap(xl[j,i],y11,xl[j,2],y12,ehv[j]); end;
```

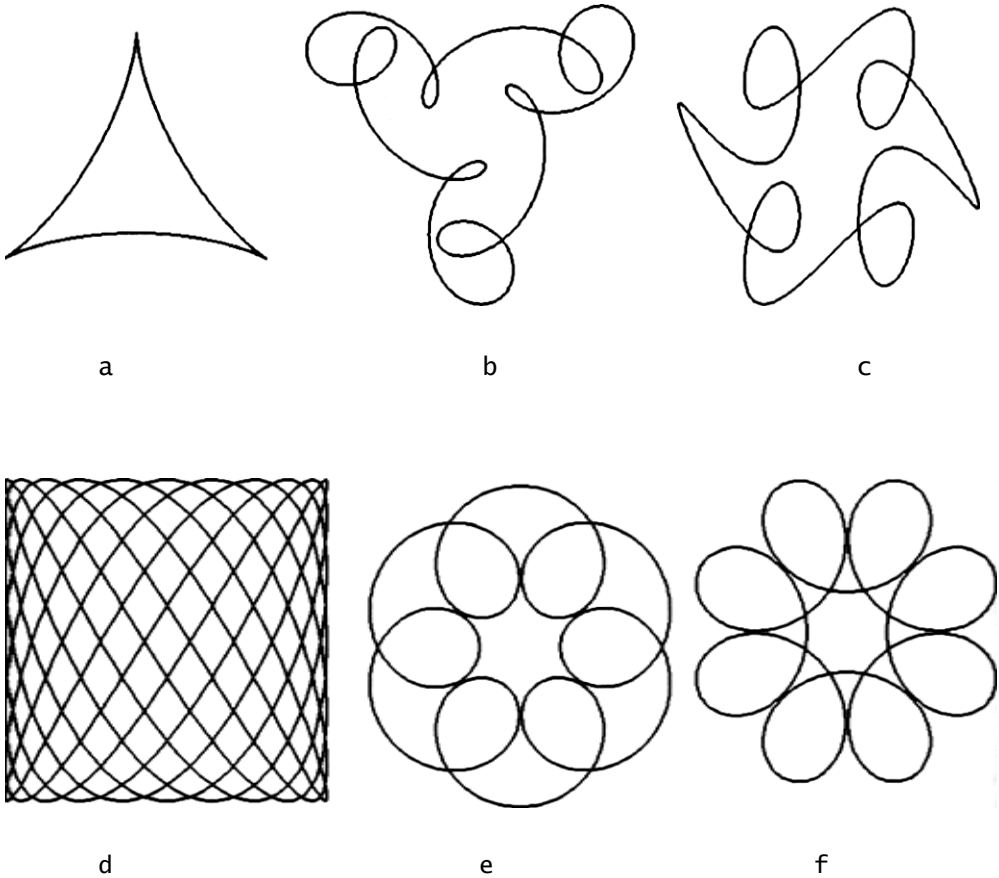
```
procedure beir(i, sz: byte); {együttható értékének beírása}
var
   xk, h, kz: integer;
   ks: string;
begin
   if odd(i) then begin h:=31; kz:=4; end else begin h:=39; kz:=5; end;
   repeat
     ablak(xl[j,1]+ó,y11+sz*18+20,x1[1,2]-5,y11+sz*18+32);
     kn:=''; 1:=1; iras(kz,xl[j,1]+3,yll+sz*18+22);
     val(kn,a[j,sz],hk);
   until hk=0;
   ablak(xl[j,1]+6,y11+sz*18+20,x1[j,2]-5,y11+sz*18+32);
   setcolor(15);
   if odd(j) then str(a[j,sz]:3:0,ks) else str(a(j,sz]:3:2,ks);
   outtextxy(x1[1,1]+10,y11+sz*18+23,ks);
end;

procedure e_valaszt; (együttható kiválasztása és módosítása)
var ev, sv, k1, h: byte;
begin
   j:=0; sz:=100;
   with reg do
     repeat
       ax:=3; intr($33,reg);
       if bx=1 then
         for i:=1 to 4 do
           if j=0 then begin
             katt(xl[i,1],yll,xl[i,2],y12,ev);
             if ev=1 then begin
               j =i;
               if odd(j) then begin kl:=0; h:=31 end
                         else begin k1:=1; h:=39; end;
               for k:=k1 to 20 do
                 if sz=100 then begin
                                katt (xl [ j , 1 ] +6, yl 1+k* 18+20, xl [ j , 2 ] -
                                5,y11+k*18+32,sv);
                   if sv=1 then sz:=k; end;
             end;
           end;
     until bx=1;
   eger_n; if sz<100 then beir(j,sz); eger_1;
end;
procedure nev_szam; {módosítandó paraméter nevének és sorszámának
                     lekérdezése}
var sa: byte;
begin
   lap(208,5); setcolor(0); outtextxy(230,26,'Együttható:');
            setcolor(12);       outtextxy(320,12,'név');      setcolor(0);
outtextxy(320,30,'szám');
   ablak(360,28,383,41);
   repeat
     j:=0;
```

```
    ablak(360,10,383,23); nv:=upcase(readkey); nv:=char(ord(nv)+32);
    setcolor(15); outtextxy(368,13,nv); delay(500);
    for i:=1 to 4 do if nv=ehv[i] then j:=i;
  until j>0;
  if (nv='a') or (nv='b') then sa:=0 else sa:=1;
           setcolor(0);     outtextxy(320,12,'név');     setcolor(12);
outtextxy(320,30,'szám');
  repeat
    ablak (360, 28, 383, 41) ; kn:=" ; 1:=1; iras (3, 356, 30) ; val (kn, sz,
  hk) ; until (hk=0) and (sz>=sa) and (sz<=nf);
end;

begin
  lx:=1000; t1:=0; t2:=2*pi;
  grindhi; keret; fuggveny; eger_k;
  repeat
    konfig; qk:='N'; qu:='N';
    if qf='U' then begin
      for k:=0 to of do for j:=1 to 4 do a[j,k]:=0;
      a[1,1]:=100; a[3,1]:=100; a[4,1]:=0.5;
    end;
    eger_n; egyutth; parancs; megj; eger_l;
    repeat
      eger_t (10, 10, 620, 470) ; qr:=' '; k:=0;
      e_valaszt; {együtthatómódosítás egérrel}
      if sz=100 then repeat par_katt(qr); until (qr='V') or (qr='Q'
                eger_n;
      case qr of
      'V':begin { }
            gb[1]:=1; parancs;
            nev_szam; beir(j,sz);
            gb[1]:=0; lap(208,5);
          end;
      'Q':begin {kilépés}
            gb[2]:=1; parancs;
            repeat kilep until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf; gb[2]:=0;
          end;
      end;
      if (qu='N') and (qk='N') then megj; parancs; eger_1;
    until (qk='I') or (qu='I');
  until qu='N';
  closegraph;
end.
```

E program segítségével készült a *2.1. ábrán* (a szerkesztési környezettel együtt) látható görbe, valamint a *2.2. ábrán* látható görbék, amelyeknél a 0-tól különböző együtthatók értékei rendre:

a) $a_1=b_1 = 100,\ q_1=q_2=0.5, a_2=b_2=50, p_2= 1$
b) $a_1=b_1 = 100, q_1=0.5, a_2=b_2=50, q_2=q_7= 1.5, a_7=b_7=40$
c) $a_1=b_1 = 100,\ q_1 =0.5,\ a_5= 50,\ b_7 = 50$
d) $a_1=b_1 = 160,\ q_{11} =0.5$
e) $a_1=b_1 = 100,\ q_1=0.5,\ a_7 = b_7 = 60,\ q_7 =0.5$
f) $a_1=b_1 = 100,\ q_1 =0.5,\ a_7=b_7=60,\ q_7=1.5$





2.2. ábra

2.3. Sima görbék illesztése kontrollpontsorra

Tetszőleges alakú sima görbék legkényelmesebb szerkesztési módja. A felhasználó a kontrollpontok számának és helyzetének megfelelő változtatásával módosítani tudja a görbe alakját. A konfigurációváltoztatás hatása a görbealakra elég jól előre látható, ugyanis a szerkesztett görbe bizonyos mértékben követi a kontrollpontokat összekötő törtvonalat (karakterisztikus keret), ami a tervezési folyamatot nagy mértékben megkönnyíti.

Ami az illesztés módját illeti, a módszerek két nagy csoportba oszthatók:
- interpolációs eljárások - az illesztett görbe átmegy minden kontrollponton
- approximációs eljárások - az illesztett görbe a kontrollpontok között halad, általában nem megy át rajtuk, viszont a kapott görbe simább és előállításának együtthatói esetenként egyszerűbben meghatározhatók

2.3.1. A kontrollpontok mozgatása

Síkgörbék szerkesztéséhez előnyös egy unit-ban (hogy több program használhassa) kontrollpont-objektumot értelmezni, amely tartalmazza a kontrollpontok abszolút- és képernyőkoordinátáit, a megjelenítési és eltüntetési (mozgatáshoz) eljárást illetve az egérrel való mozgatáshoz szükséges eljárásokat.

Ez a unit a lemezmellékleten KONF 2D.PAS néven található.

```
{$N+}
unit konf_2d; (kontrollponthalmaz meghatározása a síkban)

interface

uses
  graph, dos, kozos;

coast
  nm=99; r=10; rr=r*r;
  kx0=41; ky0=450; xn=0; yn=0; xx=560; yx=380;

type
   kpont = object(pont) {kontrollpont objektum)
   q: pointer; {bitmap heap-beli tárolási címére mutat
                kontrollpontmozgatáshoz}
   fg: byte;
   xr, yr: integer;
```

```
    procedure koord;
    procedure megj;
    procedure eltn;'
    procedure fog;
    procedure eng;
    procedure kezd;
    function bent: boolean;
  end;

  mp = array[0..nm] of kpont;

var
  a: mp;
  mm: word; {bitmap memóriamérete}

implementation

procedure kpont.koord; {kontrollpont képernyőkoordinátái}
begin
  xr:=kx0+xp;
  yr:=ky0-yp;
end;

procedure kpont.megj; {kontrollpont megjelenik)
var sf: string;
begin
  setfillstyle(1,8); setcolor(15); fillellipse(xr,yr,r,r);
  str(k, sf); if k<=9 then outtextxy(xr-2,yr-3,sf)
                      else outtextxy(xr-7,yr-3,sf);
end;

procedure kpont.eltn; {kontrollpont eltűnik)
begin
  putimage(xr-r,yr-r,q^,normalput);
end;

function kpont.bent: boolean;
      {ellenőrzi, hogy a kontrollpont az aktív mezőben van-e}
begin
  if (xp<=xx) and (yp<=yx) then- bent:=true else bent:=false;
end;

procedure kpont.fog; {egér megfogja a kontrollpontot}
begin
  if sqr(reg.cx-xr)+sqr(reg.dx-yr)<=rr then fg:=1;
end;

procedure kpont.eng; (egér elengedi a kontrollpontot)
begin
  fg =0;
end;
```

```
procedure kpont.kezd; {kontrollpont példány inicializálása)
begin
  uj(577,360); koord;
  getmem(q,mm); {helyfoglalás a bitmap-nek a heap-ben}
  getimage(xr-r,yr-r,xr+r,yr+r,q^);
  fg:=0; megj;
end;
end.
```

A kontrollpontokkal a következő műveletek végezhetők:

- beszúrás és törlés
  - — az erre szolgáló vizuális billentyűvel
  - — egérrel való húzás a kontrollponttárból a szerkesztési mezőbe illetve ebből ki
- helyváltoztatás
  - — sorszámát vizuális billentyűvel kiválasztva, majd új koordinátákat megadva
  - — egérrel az új helyre mozgatva
- sorszámváltoztatás
  - — a kiválasztott pont új sorszámát beírva

Az egérrel való mozgatás úgy történik, hogy miután a megfelelő kontrollpontra vittük az egérkurzort, lenyomjuk az egér bal gombját, ezzel "megfogjuk" a kontrollpontot és ez az egérkurzorral együtt mozog míg a gombot fel nem engedjük.

Ezen műveleteket megvalósító eljárások az EGER_KP.PAS unit-ban találhatók.

```
  {$N+}
unit eger kp; {kontrollpontmozgatás egérrel}

interface

uses
  graph, dos, crt, kozos, eger_kez, konf_2d;

var
  ef: byte;

procedure eger_f; {megfogás}
procedure egere; {elengedés}
procedure eger_ v; {egér viszi a kontrollpontot}
```

```
implementation

procedure eger_f; {megfogás}
begin               .
  with reg do begin
    ax:=3; intr($33,reg);
    if (bx=1) and (getpixel(cx,dx)=8) and (ef=0) then begin
      k:=0; repeat k:=k+1; a[k].fog; until a[k].fg=1;
      ef:=1;
    end;
  end;
end;

procedure eger_e; (elengedés)
begin
  with reg do begin
    ax:=3; intr($33,reg);
    if (bx=0) and (ef=1) then begin
      k:=0; repeat k:=k+1; if a[k].fg=1 then begin a[k].eng; ef:=0;
                                                    end;
            until of=0;
    end;
  end;
end;

procedure eger_v; {egér viszi a kontrollpontot}
begin
  with a[k] do
    with reg do begin
      ax:=3; intr($33,reg);
      if (fg=1) and (sqr(cx-xr)+sqr(dx-yr)>2) then begin
        eger_n; eltn; uj(cx-kx0,ky0-dx); koord;
        getimage(xr-r,yr-r,xr+r,yr+r,q^); megj; eger_l;
      end;
    end;
end;

end.
```

Térbeli kontrollpontok helyének meghatározása a koordináták megadásával egyszerűbb, bár itt is elképzelhető egérrel való mozgatás úgy, hogy a kontrollpontnak két koordinátasíkra eső vetületét szerre mozgatjuk és a két mozgást összehangoljuk.

A sorszámlekérdezést és koordinátabeírást a KOZOS_GB.PAS unit eljárásai végzik.

{$N+}

```
unit kozos gb;

interface

uses
  graph, kozos, konf 2d, gomb kez;

procedure sorszam;
procedure koordin;
    edure racs2;
    edure ker 2;

implementation

procedure sorszam; {kiválasztandó kontrollpont sorszámának
                    lekérdezése)
begin
  repeat
    lap(208,5); setcolor(0); outtextxy(262,26,'Sorszám:');
    ablak(330,19,361,32); kn:= "; 1:=1; iras(3,330,21); val(kn,sz,hk);
    until hk=0;
end;

procedure koordin; {síkbeli koordináták megadása)
     begin
  lap(208,5);
    etcolor(0); outtextxy(255,23,'Hely:');
    etcolor(13); outtextxy(315,15,'x = '); outtextxy(315,30,'y = '); ;
  ablak(340,12,378,24); ablak(340,27,378,39);
  repeat kn:= "; 1:=1; iras(4,340,14); val(kn,x,hk);
    if hk<>0 then ablak(340,12,378,24); until hk=0;
  if x>xx then begin x:=xx; ablak(340,12,378,24);
    str(x,kn); setcolor(15); outtextxy(348,15,kn); end;
  if x<xn then begin x:=xn; ablak(340,12,378,24);
    str(x,kn); setcolor(15); outtextxy(348,15,kn); end;
  repeat kn:=" ; 1:=1; iras (4, 340, 29) ; val (kn, y, hk) ;
    if hk<>0 then ablak(340,27,378,39); until hk=0;
  if y>yx then y:=yx; if y<yn then y:=yn;
end;

procedure racs2;
begin
  setfillstyle(1,0); bar(kx0-12,ky0-yx-12,kx0+xx+32,ky0+12);
  setcolor(15); if gb[2]=0 then outtextxy(607,67,'Tár'); setcolor(7);
  for k:=0 to 56 do line (kx0+k*10,ky0,kx0+k*10,ky0-yx);
  for k:=0 to 38 do line (kx0,ky0-k*10,kx0+xx,ky0-k*10);
end;
```

```
procedure ker_2; (keret koordinátákkal)
var ks: string;

begin
   setfillstyle(1,11); bar(0,0,640,480);
   bkeret(kx0-16,ky0-yx-16,kx0+xx+36,ky0+16,4,4);

   for k:=0 to 5 do begin
     line(kx0+k*100,ky0+12,kx0+k*100,ky0+20);
     str(k*100,ks); if k>0 then outtextxy(kx0+k*100-25,ky0+22,ks)
                           else outtextxy(kx0+k*100-14,ky0+22,ks) end;

   for k:=0 to 3 do begin
     line(kx0-30,ky0-k*100,kx0-12,ky0-k*100);
     str(k*100,ks); if k>0 then outtextxy(kx0-40,ky0-k*100-10,ks)
                          else outtextxy(kx0-30,ky0-10,ks) end;

end;
end.
```

2.3.2. A görbeelőállítás elve

Ha adott *n +1* pontból álló kontrollpontsor, a rájuk (approximációs eljárások esetén "közéjük") illeszkedő görbe előállítása a következő:

$$\mathbf{r}(u) = \sum_{k=0}^{n} f_k(u)\mathbf{p}_k \quad , \tag{2.9}$$

ahol $\mathbf{p}_0$, $\mathbf{p}_1$, ... , $\mathbf{p}_n$ a kontrollpontok helyvektorai, f_k , k =0,1,...,n pedig egyváltozós függvények, úgynevezett blending (keverő) függvények. Súlyfüggvényeknek is nevezhetjük őket, ugyanis a 2.9 összefüggés rögzített *u* értékre a kontrollpontok helyvektorainak súlyozott közepe, $\sum_{k=0}^{n} f_k(u) = 1$.

A továbbiakban polinomiális interpolációval foglalkozunk, vagyis a súlyfüggvények algebrai polinomok.

2.3.3. Coons-Hermite interpoláció

A Coons-Hermite interpoláció az illesztési feladatot helyileg, a karakterisztikus keret szakaszaira külön-külön oldja meg. Szükség van viszont a kontrollpontokban az érintővektorok értékeire.

Két szomszédos kontrollpontot összekötő görbeív analitikus előállítása egy harmadfokú, egyváltozós vektorfüggvény:

$$r(u) = a_3u^3 + a_1u^2 + a_1u + a_0 ; \qquad u \in [0,1] . \tag{2.10}$$

Az ív két vége a két kontrollpontra (Po és pl) kell támaszkodjon, a végpontokban az érintő egyenlő kell legyen az adott értékekkel (po" és **p** I" **):**

$$\begin{cases} \mathbf{r}(0) = \mathbf{p}_0 \\ \dfrac{d\mathbf{r}}{du}(0) = \mathbf{p}_0'' \end{cases} \qquad \begin{cases} \mathbf{r}(1) = \mathbf{p}_1 \\ \dfrac{d\mathbf{r}}{du}(1) = \mathbf{p}_1'' \end{cases} \tag{2.11}$$

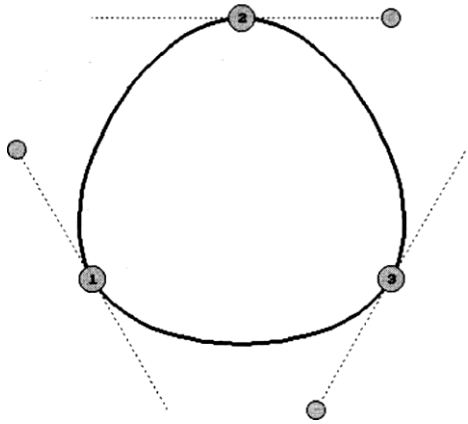
Következésképpen meghatározhatóak a harmadfokú súlyfüggvények, amelyek kielégítik a 2.11 feltételeket:

Az érintő folytonosságának feltétele, hogy a szomszédos íveknek a közös végpontjukban azonos legyen az érintő. A görbület az illesztési pontokban általában nem folytonos.

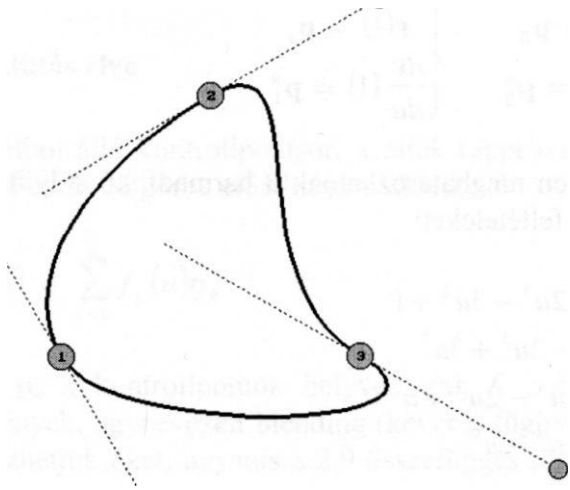
A görbeív előállítása

$$\begin{cases} f_1(u) = 2u^3 - 3u^2 + 1 \\ f_2(u) = -2u^3 + 3u^2 \\ f_3(u) = u^3 - 2u^2 + u \\ f_4(u) = u^3 - u^2 \end{cases} . \tag{2.12}$$

$$\mathbf{r}(u) = f_1(u)\mathbf{p}_0 + f_2(u)\mathbf{p}_1 + f_3(u)\mathbf{p}_0'' + f_4(u)\mathbf{p}_1'' \ . \tag{2.13}$$



a



b

2.3. ábra

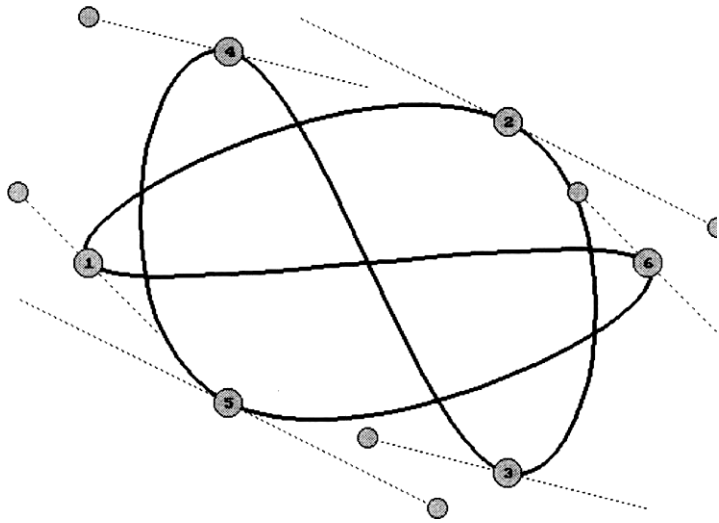
A görbe alakja természetesen a négy határfeltétel bármelyikének változtatásával módosítható, amint az a *2.3. ábrán* is látható. Az érintők meghatározása azonban a felhasználó számára kényelmetlen lehet. Ez a probléma megkerülhető például úgy, hogy a program minden belső kontrollponthoz tartozó érintőt a két szomszédos kontrollpontot összekötő vektorként számít ki, azaz a p_i és p_{i+1} pontokat összekötő görbeív

Nyílt görbe esetén a végpontokban az érintő az illető pontot a szomszédos kontrollponttal összekötő vektor. Így készült a *2.4. ábra.*

$$\mathbf{r}(u) = f_1(u)\mathbf{p}_i + f_2(u)\mathbf{p}_{i+1} + f_3(u)\mathbf{p}_i^u + f_4(u)\mathbf{p}_{i+1}^u \qquad (2.14$$

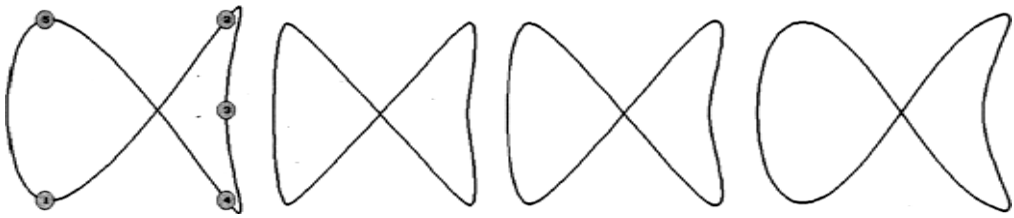
előállításában

$$\begin{aligned} \mathbf{p}_0^u &= \mathbf{p}_1 - \mathbf{p}_0 \\ \mathbf{p}_i^u &= \mathbf{p}_{i+1} - \mathbf{p}_{i-1} \quad i = 1,2,\ldots,n-1 \\ \mathbf{p}_n^u &= \mathbf{p}_n - \mathbf{p}_{n-1} \end{aligned} \qquad (2.15$$

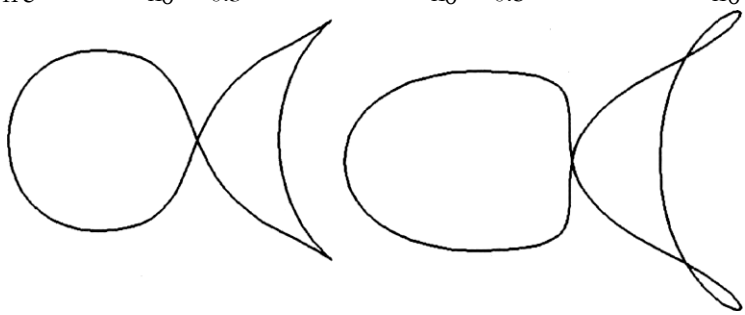


2.4. ábra

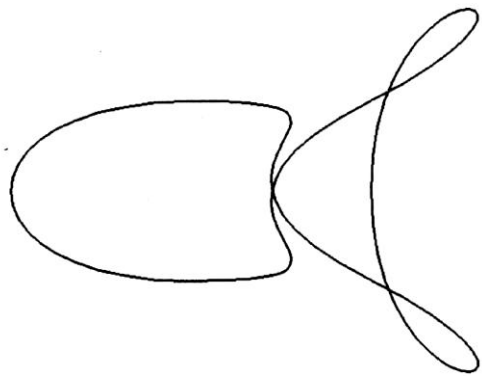
Ha a 2.15 összefüggések jobboldalát megszorozzuk egy tetszőleges ko állandóval, a görbe alakja módosul, az érintő viszont továbbra is folytonos marad. A *2.4. ábrán* $k_0 = 1$. Kisebb k_0 értékekre a görbe változásának mértéke csökken mintha egy rugalmas huzalt egyre jobban kifeszítenénk), nagyobb értékekre pedig megnövekedik, szögpontok vagy újabb hurkok is megjelenhetnek. A *2.5. ábrán* ugyanarra a karakterisztikus keretre illesztett, úgynevezett spline görbe (lásd *2.3.4.* alpont) és néhány, a bemutatott módszerrel szerkesztett interpolációs görbe látható. Megfigyelhető, hogy negatív k_0 értékekre a görbén hurkok jelennek meg, amint az várható is, hiszen az érintők iránya a kontrollpontokban ellenkező lesz a kontrollpontok helyéből és sorrendjéből származóval.



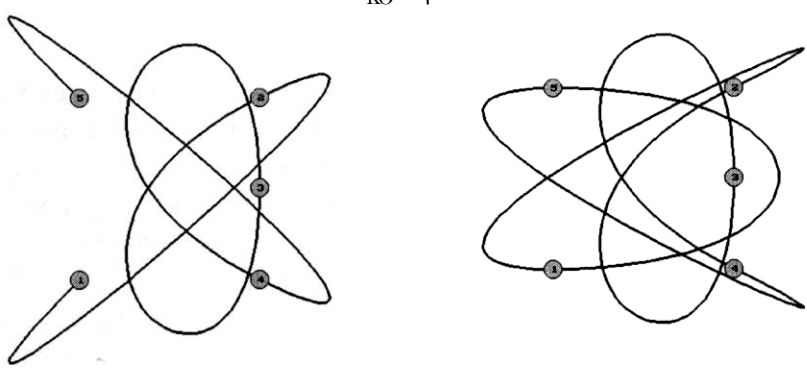
spline $k_0 = 0.3$ $k_0 = 0.5$ $k_0 = 1$



$k_0 = 2$ $k_0 = 3$



$k_0 = 4$



$k_0 = -5$

2.5. ábra

A $k_0 = 0.5$ értéknek megfelelő eljárás az úgynevezett Overhauser interpoláció, amelyet a következő módon is megkaphatunk:

tekintsük a három-három (p_{i-1} p_i , p_{i+1} illetve p_i, p_{i+1} , p_{i+2}) egymás utáni kontrollponton átmenő kúpszeletek

$$\mathbf{r}_1(u) = \frac{u(u-1)}{2}\mathbf{p}_{i-1} - (u+1)(u-1)\mathbf{p}_i + \frac{u(u+1)}{2}\mathbf{p}_{i+1}$$
$$\mathbf{r}_2(u) = \frac{(u-1)(u-2)}{2}\mathbf{p}_i - u(u-2)\mathbf{p}_{i+1} + \frac{u(u-1)}{2}\mathbf{p}_{i+2} \qquad (2.16$$

előállításait, amelyeknek az u E [0,1] paramétertartománynak megfelelő ívei a p_i és p_{i+1}, pontokat kötik össze.

- szerkesszük meg a fenti két másodfokú polinom lineáris interpolációját a [0,1] intervallumon:

$$r(u) = (1—u)\, r_1(u) + u r_2(u) \;;\; u \in [0,1] \;. \qquad (2.17)$$

Egy harmadfokú vektorpolinomot

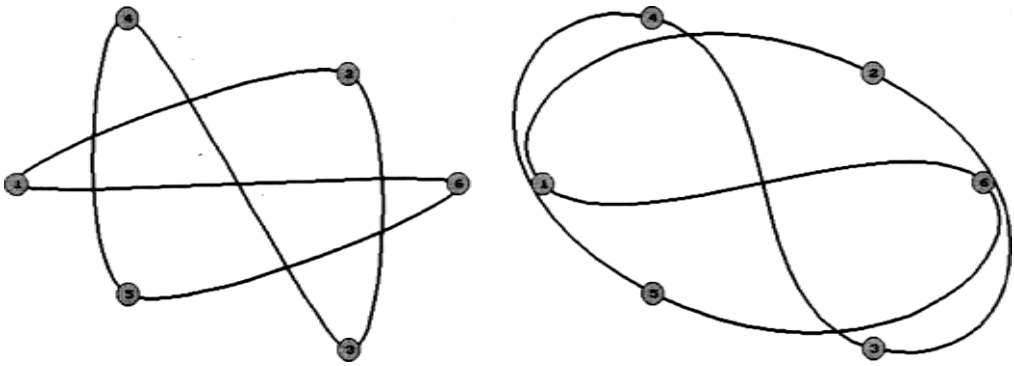
$$\mathbf{r}(u) = \frac{-u^3 + 2u^2 - u}{2}\mathbf{p}_{i-1} + \frac{3u^3 - 5u^2 + 2}{2}\mathbf{p}_i + \frac{-3u^3 + 4u^2 + u}{2}\mathbf{p}_{i+1} + \frac{u^3 - u^2}{2}\mathbf{p}_{i+2} \;;\quad u \in [0,1] \qquad (2.18$$

amellyel az i értékeit 0 és n-1 között változtatva, nyílt görbék végein a $\mathbf{p}_{-1}=\mathbf{p}_0$ **és** $P_{n+1} = P_n$ segédpontokkal, olyan görbét állíthatunk elő, amely a csatlakozási pontokban folytonos érintővektorokkal rendelkezik, amelyek a

$$\mathbf{p}_0^u = \frac{\mathbf{p}_1 - \mathbf{p}_0}{2}$$
$$\mathbf{p}_i^u = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2}; \quad i = 1,2,\ldots,n-1 \qquad (2.19$$
$$\mathbf{p}_n^u = \frac{\mathbf{p}_{n+1} - \mathbf{p}_n}{2}$$

képletekkel

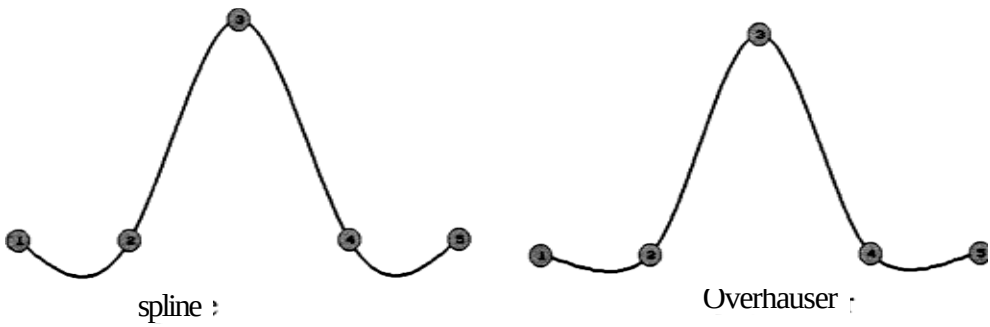
A *2.6. ábrán* a *2.4. ábra (*k_0 *= 1)* karakterisztikus keretére illesztett görbék láthatóak, $k_0 = 0.5$ (Overhauser) és $k_0 = 2$ esetre.



k_0=0.5 k_0 = 2

2.6. ábra

Az Overhauser interpoláció a spline interpolációnál lényegesen kisebb globális változású görbét eredményez *(2.7. ábra),* ami bizonyos alkalmazások esetében kedvező lehet.



spline Overhauser

2.7. ábra

A GB_2D_CH.PAS nevű program az érintők módosítását egérrel is lehetővé teszi. Az érintőket teljes hosszukban jeleníti meg, úgy, hogy az érintővektor középpontja kerül a kontrollpontba. Az érintővektor csúcsán aktív körlap található, amely egérrel megfogható és mozgatható.

```
($N+)
program gb_2d_ch; {síma síkgörbék előállítása Coons-Hermite
                   interpolációval}
uses
 graph, grafind, crt, dos, kozos, file_kez, konf_2d, eger_kez,
 kerdesek, gomb_kez, kozos_gb;
coast
 valasz: kars = ['B','E','G','Q','T','V'];

var
 ep: mp; {érintőpontok}
 ff: file of mp;
 ef: byte;
 e, f: array[l..nm] of real;
 nh, fv, bill: byte; er: word;
 st, i integer; p: kpont;

procedure eger_f; {kontrollpont vagy érintőcsúcs megfogása)
begin
 with reg do begin
   ax:=3; intr($33,reg);
   if (bx=1) and (getpixel(cx,dx)=8) and (ef=0) then begin
     k:=0;
               repeat k:=k+1; a[k].fog; ep[k].fog until (a[k].fg=1) or
             (ep[k].fg=1);
     ef:=1;
   end;
 end;
end;

procedure eger_e; {megfogott objektum elengedése)
begin
 with reg do begin
   ax:=3; intr($33,reg);
   if (bx=0) and (ef=1) then begin
     k:=0;
     repeat k:=k+1;
        if a[k].fg=1 then begin a[k].eng; ef:=0; end;
        if ep[k].fg=1 then begin ep[k].eng; ef:=0; end;
     until of=0;
   end;
 end;
end;

procedure ekoord(sz: integer); {érintővektor csúcsának
                                képernyőkoordinátái}
begin
 ep[sz].xr:=a[sz].xr+round(ep[sz].xp/2);
ep[sz].yr:=a[sz].yr-round(ep[sz].yp/2); end;
```

```
procedure emegj(s: integer); {érintővektor megjelenítése}
var
  x1, y1, x2, y2: integer;
  sf: string;
begin
  setfillstyle(1,8); setcolor(15);
  with ep[s] do begin
    getimage(xr-r,yr-r,xr+r,yr+r,q^);
    x1:=xr; y1:=yr; {csúcs}
    x2:=2*a[s]..xr-xr; y2:=2*a[s].yr-yr; {kezdőpont}
  end;
  setlinestyle(1,0,1); line(x1,y1,x2,y2); setlinestyle(0,0,1);
  fillellipse(x1,y1,7,7); {csúcson levő aktív korong, ami egérrel
                           mozgatható}
end;

procedure eger_v; {egér viszi a pontot}
var t: byte;
begin
  setfillstyle(1,8); setcolor(15);
  if a[k].fg=1 then t:=1 else t:=2; {1=kontrollpont; 2=érintő}
  if t=1 then p:=a[k] else p:=ep[k];
  with p do
    with reg do begin
      ax:=3; intr($33,reg);
      if (fg=1) and (sqr(cx-xr)+sqr(dx-yr)>2) then begin
        eger_n; eltn;
        case t of
        1:begin
            uj(cx-kx0,ky0-dx); koord;
          end;
        2:begin
            uj(2*(cx-a[k].xr),-2*(dx-a[k].yr));
            p.xr:=a[k].xr+round(p.xp/2);
            p.yr:=a[k].yr-round(p.yp/2);
          end;
        end;
        getimage(xr-r,yr-r,xr+r,yr+r,q^);
        if t=1 then megj else fillellipse(xr,yr,7,7); eger_l;
      end;
    end;
  if t=1 then begin a[k]:=p; ekoord(k); end else ep[k]:=p;
end;

procedure kilep;
begin

  kerdez(200,200,'      Menteni a konfigurációt
  repeat kattint(200,200,'I','N',qm);
  until (qm='I') or (qm='N');
  if qm='I' then begin
    eger n; ment('IC2'); (kontrollpontok tárolása)
```

```
    if v='I' then begin
                      a[0].uj(nh,0); assign(ff,kn); rewrite(ff);
                      write (ff, a) ;
                     close(ff);
                   end;
    delete(kn,length(kn)-2,3); kn:=kn+'EC2';
    if v='I' then begin (érintők tárolása}
      assign(ff,kn); rewrite(ff); write(ff,ep); close(ff); end;
    eger_l;
  end;
end;

procedure konfig; begin
  repeat
    qf:=' ';
    kerdez(200,200,'           Konfiguráció
    repeat kattint(200,200,'U','T',qf);
    until (qf='U') or (qf='T');
    if qf='T' then begin
      eger_n; olvas('IC2');
      if ko=1 then begin
                      assign(ff,kn); reset(ff); read(ff,a);
                      close(ff); {kontrollpontok}
                      delete(kn,length(kn)-2,3); kn:=kn+'EC2';
                      assign(ff,kn); reset(ff); read(ff,ep);
                      close(ff);
                     end; {érintők}
      eger_l;
    end else ko:=1;
until ko=1;
end;

procedure erintosz; {új érintőirányítás lekérdezése} begin
  lap(208,5); setcolor(0);
  outtextxy(285,10,'Érintö irányítása'); outtextxy(285,9,'              "');
  circle(245,26,8); outtextxy(256,22,'0'); outtextxy(212,22,'180');
  outtextxy(239,9,'90'); outtextxy(235,36,'270'); ablak(340,27,378,39);
  outtextxy(380,27,'r');
  repeat kn:=''; 1:=1; iras(4,340,29); val(kn,er,hk); if (hk<>0)
    or (er>359) then ablak(340,27,378,39);
  until (hk=0) and (er<360);
end;

procedure erintoh; {új érintőhossz lekérdezése} var x, y:
real;
begin
  lap(208,5); setcolor(0);
  outtextxy(285,10,'Érintö hossza'); outtextxy(285,9,'           "');
  x:=ep[sz].xp; y:=ep[sz].yp;
  str(round(sgrt(sgr(x)+sgr(y))),kn);
  outtextxy(212,30,'most: '+kn+'        új:');
```

```
   ablak-(340,27,378;39);
   repeat kn:=''; 1:=1; iras(4,340,29); val(kn,er,hk);
   if hk<>0 then ablak(340,27,378,39); until hk=0;
   end;

procedure erintsz(szog: word);
      {új irányításnak megfelelő érintőkoordináták kiszámítása}
var alf, x, y, re: real;
begin
   x:=ep[sz].xp; y:=ep[sz].yp;
   re:=sqrt(sqr(x)+sqr(y));
   alf:=szog*pi/180;
   ep[sz].uj(round(re*cos(alf)),round(re*sin(alf)));
   ekoord(sz);
end;

procedure erinth(h: word);
      {új hossznak megfelelő érintőkoordináták kiszámítása}
var x, y, re: real;
begin
   x:=ep[sz].xp; y:=ep[sz].yp;
   re:=sqrt(sqr(x)+sqr(y));
   if re<1 then ep[sz].uj(h,0)
   else ep[sz].uj(round(x*h/re),round(y*h/re));
   ekoord(sz);
end;

procedure kpontok; {kontrollpontok és érintők megjelenítése}
begin
   for k:=1 to nh do emegj(k);
   for k:=1 to nh do a[k].megj;
end;

procedure ersz; {érintők kiszámítása a szomszédos pontok függvényében}
begin
   alak;
   egyutth; lap(208,5); str(k0:3:2,kn);
   setfillstyle(1,11); bar(60,40,150,50); outtextxy(65,43,'k0 =
   '+kn); case qa of
   'N':begin {nyílt görbe esetére}
      ep[1].xp:=round((a[2].xp-a[1].xp)*k0);
      ep[1].yp:=round((a[2].yp-a[1].yp)*k0);
      ep[nh].xp:=round((a[nh].xp-a[nh-1].xp)*k0);
      ep[nh].yp:=round((a[nh].yp-a[nh-1].yp)*k0); end;
   'Z':begin {zárt görbe esetére)
      ep[1].xp:=round((a[2].xp-a[nh].xp)*k0);
      ep[1].yp:=round((a[2].yp-a[nh].yp)*k0);
      ep[nh].xp:=round((a[1].xp-a[nh-1].xp)*k0);
      ep[nh].yp:=round((a[1].yp-a[nh-1].yp)*k0); end;
      end;
   for k:=2 to nh-1 do begin
    ep[k].xp:=round((a[k+l].xp-a[k-1].xp)*k0);
    ep[k].yp:=round((a[k+1].yp-a[k-1].yp)*k0); end;
```

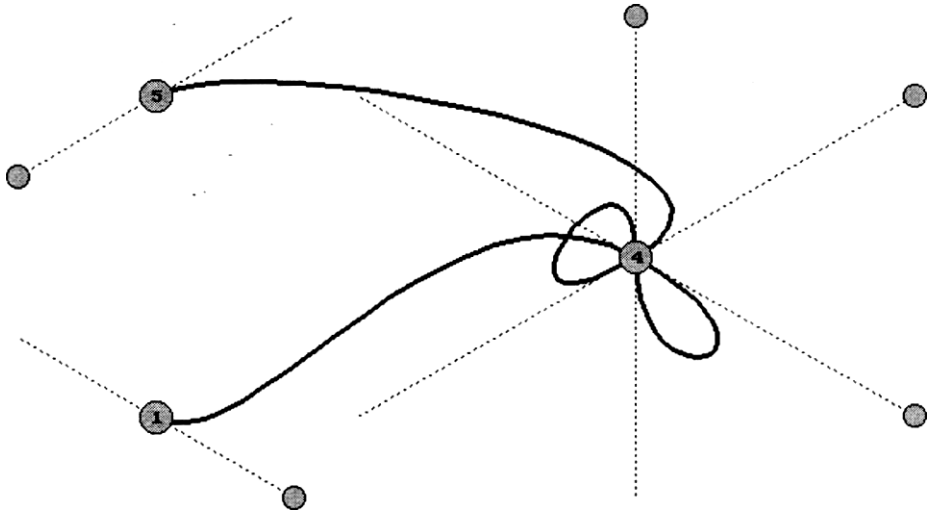
```
  for k:=1 to nh do
ekoord(k); racs2; kpontok;
end;

procedure ujrasz; {felhasználó által elrendelt érintőújraszámítás}
begin
  ujra; if qm='I' then ersz;
end;

procedure rajz; {görbe megjelenítése}
coast nr=20;

var
  j: integer;
  u, u2, u3, w, z: real;
function xr(i, j: integer): integer; (görbepont x koordinátája)
begin
  case qa of
  'N':begin
      u:=j/nr; u2:=u*u; u3:=u2*u;
      xr:=kx0+round((2*u3-3*u2+1)*a[i].xp+(-2*u3+3*u2)*a[i+l].xp+
                    (u3-2*u2+u)*ep[i].xp+(u3-u2)*ep[i+l].xp);
    end;
  'Z':begin
      u:=j/nr; u2:=u*u; u3:=u2*u;
      if i=nh then
         xr:=kx0+round((2*u3-3*u2+1)*a[nh].xp+(-2*u3+3*u2)*a[1].xp+
              (u3-2*u2+u)*ep[nh].xp+(u3-u2)*ep[1].xp)
      else
         xr:=kx0+round((2*u3-3*u2+1)*a[i].xp+(-2*u3+3*u2)*a[i+1).xp+
              (u3-2*u2+u)*ep[i].xp+(u3-u2)*ep[i+l].xp);
    end;
  end;
end;
function yr(i, j: integer): integer; (görbepont y koordinátája)
begin
  case qa of
  'N':begin
      u:=j/nr; u2:=u*u; u3:=u2*u;
      yr:=ky0-round((2*u3-3*u2+1)*a[i].yp+(-2*u3+3*u2)*a[i+l].yp+
                    (u3-2*u2+u)*ep[i].yp+(u3-u2)*ep[i+l].yp);
    end;
  'Z':begin
      u:=j/nr; u2:=u*u; u3:=u2*u;
      if i=nh then
         yr:=ky0-round((2*u3-3*u2+1)*a[nh].yp+(-2*u3+3*u2)*a[1].yp+
              (u3-2*u2+u)*ep[nh].yp+(u3-u2)*ep[1].yp)
      else
         yr:=ky0-round((2*u3-3*u2+1)*a[i].yp+(-2*u3+3*u2)*a[i+l].yp+
              (u3-2*u2+u)*ep[i]•yp+(u3-u2)*ep[i+l].yp);
    end;
  end;
end;
```

```
begin
  eger_n;
  racs2; setlinestyle(0,0,3); setcolor(15);
  if qa='N' then moveto(a[1].xr,a[1].yr);
  if qa='Z' then moveto(xr(1,0),yr(1,0));
    for is=1 to nh-1 do for j:=1 to nr do lineto(xr(i,j),yr(i,j));
    if qa='Z' then for j:=1 to nr do lineto(xr(nh,j),yr(nh,j));
  setlinestyle(0,0,1);
  eger_l;
end;

procedure tar; {kontrollponttár - ahonnan új pontot lehet egérrel
                elvenni}
begin
  k:=nh+l; a[k].kezd; ep[k].kezd; ep[k].uj(0,0); ekoord(k);
end;

begin
  grindhi; ker_2; eger_k; mm:=imagesize(0,0,2*r,2*r);
  ef:=0; getmem(p.q,mm) ;
  repeat
    qu:='N'; fe:=0;
    for k:=0 to nm do a(k].uj(0,0); qk:='N';
    eger_n; racs2; konfig; nh:=a[0].xp; parancs(3);
    eger_n; racs2; eger_l; setcolor(15);
    if nh>0 then for k:=1 to nh do begin {kezdőkonfiguráció
                                          megjelenítése}
      with ep[k] do begin
        getmem(q,mm); fg:=0; end;
      emegj (k) ;
      with a[k] do begin
        getmem (q, mm) ; getimage (xr-r, yr-r, xr+r, yr+r, q^) ; fg:=0; megj;
      end;
    end;
    tar;
    if nh>2 then ujrasz;
    repeat
      egér_t(kx0,10,620,ky0); eger_1; qr:=' '; k:=0;
      bill:=1; {billentyűzet aktív}
      repeat eger_f; until (ef=1) or (keypressed) or gombny(valasz);
      if of=1 then bill:=0; {billentyűzet az elengedésig nem aktív}
      repeat eger_v; eger_e; until ef=0;
      if k=nh+1 then begin if a[k].bent then nh:=nh+1; end
                else if not (a[k].bent) then begin
                          {aktív mezőről lehúzott kontrollpont törlése}
      eger_n;
      for sz:=k+1 to nh do begin a[sz-1]:=a[sz]; ep[sz-1]:=ep[sz];
                          end;
      nh:=nh-1; end;
      if bill=l then begin
        repeat par_katt(1,qr); until qr in valasz; eger_n;
```

```
case qr of
  'B':if nh<nm then begin (új kontrollpont beszúrása)
      gb[6]:=1; parancs(3);
      sorszam;
      if sz>0 then begin
        if sz>nh then sz:=nh+1;
        if sz<=nh then for k:=nh downto sz do begin
          a[k+1]:=a[k]; ep[k+1]:=ep[k]; end;
        koordin; a[sz].uj(x,y); a[sz].koord;
        ep[sz].uj(0,0); ekoord(sz);
        nh:=nh+1;
      end;
      gb[6]:=0; lap(208,5); fe:=0;
    end;
  'V':begin {váltortatandó kontrollpont kiválasztása}
      gb[1]:=1; parancs(3); sorszam;
      if (sz>0) and (sz<=nh) then begin
        kp_er;
        case qr of
        'E':begin (érintőváltortatás)
            evlt;
            case qe of
            'H':begin erintoh; erinth(er); end; (hossz)
            'S':begin erintosz; erintsz(er); end; (irányítás)
            end;
          end;
        'P':begin vlt; {pont sorszám/hely váltortatása}
            case qr of
            'S':begin
                  st:=sz; a[0]:=a[sz]; ep[0]:=ep[sz]; sorszam;
                if (sz>st) and (sz<=nh) then
                   for k:=st to sz-1 do begin
                     a[k]:=a[k+1]; ep[k]:=ep[k+1]; end;
                if (sz<st) and (sz>0) then
                  for k:=st downto sz+1 do begin
                    a[k]:=a[k-1]; ep[k]:=ep[k-1]; end;
                a[sz]:=a[0]; ep[sz]:=ep[0];
              end;
            'H':begin koordin;
                  a[sz].uj(x,y); a[sz] .koord; ekoord(sz); end;
            end;
          end;
        end;
      end;
      gb[1]:=0; lap(208,5); fe:=0;
    end;
 'T':if nh>0 then begin {kontrollpont törlése}
      gb[5]:=1; parancs(3); sorszam;
      if sz>0 then begin
        if sz<nh then for k:=sz+1 to nh do begin
          a[k-1]:=a[k]; ep[k-1]:=ep[k]; end;
        if sz<=nh then nh:=nh-1;
```

```
            end;
            gb(5]:=0; lap(208,5);
         end;
      'G':if nh>2 then begin {görbemegjelenítés}
            gb[3]:=1; parancs(3); alak;
            rajz;
            gb[3]:=2; fe:=1;
         end;
      'E':if fe=1 then begin {görbemegjelenítés keret nélkül}
            gb[2]:=1; parancs(3); kpl; cleardevice;
            if qm='I' then begin racs2; rajz; kpontok; end
                      else rajz;
            setcolor(7); outtextxy(5,5,'ESC - vissza');
            repeat qm:=readkey; until ord(qm)=27;
            ker_2;
            gb[2] :=0;
         end;
      'Q':begin (kilépés)
            gb[4]:=1; parancs(3);
            repeat kilep until not ((v='N') and (qm='I'));
            vege; if qk='N' then begin uj konf ; gb [ 4 ] : =0; end;
         end;
      end;
   end;
   if gb[4]=0 then begin
      if gb[3]=0 then racs2; kpontok; gb[3):=0; parancs(3); eger_1;
      if (nh>2) and (qr<>'Q') then ujrasz; tar;
   end;
 until (qk='I') or (qu='I');
  for k:=1 to nh do begin dispose(a[k].q); dispose(ep[k].q); end;
       (a bitmap-memóriák felszabadítása a heap-ben)
until qu='N';
 closegraph;
end.
```

Az érintők megfelelő kiválasztásával többszörös (több, ugyanazon helyen álló, a sorban szomszédos) kontrollpontokban is meghatározható a görbe alakja, amint az a *2.8. ábrán* is látható, amelyen a 4-es számmal jelölt kontrollpont háromszoros (a 2, 3 és 4 sorszámú pontok azonos helyen állnak), így a leírt önműködő érintőmeghatározás a 3-as sorszámú pontra nulla hosszúságú érintőt adna.



2.8. ábra

2.3.4. Spline interpoláció

A spline angol eredetű szó, egy, a műszaki rajzban használt eszköz neve. Ez egy hosszú, vékony, rugalmas vessző volt, amelyen hosszában több nehezék szabadon csúszhatott. A nehezékeket a rajzasztalon a kívánt pontokba (kontrollpontok) helyezve, a vessző egy, a pontokat összekötő sima görbe mentén helyezkedett el. A szilárdságtanban ismert elv, hogy rugalmas tárgyak külső erők hatására olyan alakot vesznek fel, amely az adott megszorítások mellett a helyzeti energia minimumát adja. Hajlított rugalmas vessző helyzeti energiája a görbület négyzetének integráljával arányos, az y = *f(x)* előállítású síkgörbe görbülete pedig:

$$\frac{1}{\rho} = \frac{y''}{\left(1 + y'^2\right)^{3/2}} \quad , \qquad (2.20$$

ahol p a görbületi sugár, ' az első-, " a második deriváltat jelenti. Nem túl nagy görbületi értékek esetén a nevező hatása elhanyagolható és a helyzeti energia az

$$\int_{x_0}^{x_n} \left(y''\right)^2 dx$$

integrállal arányos.
A funkcionál-analízis egyik fontos eredménye a következő: ha adott *n* +1 pont: (x_0,y_0), $(x_1 ,y_1),\ldots,(x_n,y_n)$ és $s \in C^2 [x_0, x_n]$ (az adott intervallumon első két deriváltjukkal együtt folytonos függvények halmaza), amely:

- minden $[x_k ,x_{k+1}]$, $k= 0, 1,..., n$ —1 intervallumon harmadfokú polinom
- $s(x_k) = y_k$, k = 0, 1,..., n
- x_0 és x_n pontokban a második deriváltja nulla

és g E C^2 $[x_0 ,x_n]$ egy másik függvény, amelyre $g(x_k) = y_k$, k= 0, 1,..., *n* , akkor

$$\int_{x_0}^{x_n} \left(g''(x)\right)^2 dx \geq \int_{x_0}^{x_n} \left(s''(x)\right)^2 dx$$

és az egyenlőség csakis akkor áll fenn, ha $g \equiv s$.

Ez azt jelenti, hogy a harmadfokú spline függvények *(s)* az összes másodrendű folytonos függvények közül a legkisebb "rugalmas helyzeti energiával" interpolálják az adott kontrollponthalmazt.

A paraméteres előállítású görbék esetében (beleértve az y = *f(x)* előállítást is), adott kontrollpontsorra a megfelelő spline-függvény összetevői a különböző koordinátákra (derékszögű koordinátarendszerre: síkban x és y, térben *x,* y és *z)* egymástól függetlenül állíthatók elő a következőképen:

Legyen f_k , k = 0, 1,..., n a kontrollpontsor valamelyik koordinátájának sora, *g: [a.b]—>* ℜ a spline függvény megfelelő összetevője, t a görbe előállításának paramétere (a *g* függvény változója), $t_0 = a$, t_1,...,t_{n-1}, $t_n = b$ pedig az értelmezési tartomány egy felosztása (természetesen a görbe mindegyik előállítási függvényének kiszámításakor ugyanazt a felosztást kell használni).

A g függvény a következő feltételeknek kell, hogy eleget tegyen: (2.23)

- minden $[t_k, t_{k+1}]$ k = 0, 1,..., n —1 intervallumon harmadfokú polinom
- $g(t_k) = f_k$ k = 0, 1,..., n - annak feltétele, hogy a görbe átmenjen a kontrollpontokon
- $g'(t_k + 0) = g'(t_k - 0)$ - az érintő folytonosságának feltétele
- $g''(t_k + 0) = g''(t_k - 0)$ - a görbület folytonosságának feltétele

Ezenkívül szükség van még a szélső kontrollpontokban egy-egy határfeltételre. Ezek a feltételek a keresett görbealaktól függően esetenként lehetnek:

a) nyílt görbe, szabad végekkel (a szélső pontokban nulla a második derivált):

$$g''(t_0) = g''(t_n) = 0 \ . \qquad (2.24$$

b) nyílt görbe, a végeken adott

$$g'(t_0 + 0) = f_0' \ , \ g'(t_n - 0) = f_n' \qquad (2.25$$

Ez a feltétel használható gömb topológiájú zárt felületek paramétervonalainak előállításánál, ha ismertek az érintők a pólusokon.

c) zárt görbe - ez annyit tesz, hogy a spline függvény koordináták szerinti összetevői periodikus függvények, a periódus az értelmezési tartomány hossza. Eggyel több kontrollpontot tekintünk (és a felosztásban eggyel több intervallumot): $f_{n+1} = f_0$

$$g(t_{n+1}) = f_0\ ,\ g'(t_{n+1} - 0) = g'(t_0 + 0)\ ,\ g''(t_{n+1} - 0) = g''(t_0 + 0)\ . \qquad (2.2$$

Abból, hogy a függvény szakaszonként harmadfokú polinom és második deriváltja folytonos az következik, hogy a másodrendű derivált grafikus képe egy törtvonal (a kontrollpontokban levő másodrendű deriváltértékeket összekötő szakaszokból áll). Ez a 2.24 esetben:

$$g''(t) = 6a_k \frac{t_{k+1} - t}{d_k} + 6b_k \frac{t - t_k}{d_k}\ ; \quad d_k = t_{k+1} - t_k \qquad (2.27)$$
$$a_0 = a_n = 0\ , \quad b_k = a_{k+1}\ ; \quad k = 0,1,\ldots,n-1$$

Kétszer integrálva a fenti képletet, a g függvény következő szakaszonkénti előállítását kapjuk (figyelembe véve a kontrollpontokon való átmenés feltételét is):

$$g(t) = a_k \frac{(t_{k+1} - t)^3}{d_k} + a_{k+1} \frac{(t - t_k)^3}{d_k} + \left(f_k - a_k d_k^2\right) \frac{t_{k+1} - t}{d_k} + \qquad (2.28)$$
$$+ \left(f_{k+1} - a_{k+1} d_k^2\right) \frac{t - t_k}{d_k}\ ; \quad t_k \le t \le t_{k+1}$$

ami, figyelembe véve az érintő és a görbület folytonosságának feltételeit (2.23), a

$$d_{k-1} a_{k-1} + 2(d_{k-1} + d_k) a_k + d_k a_{k+1} = \frac{f_{k+1} - f_k}{d_k} - \frac{f_k - f_{k-1}}{d_{k-1}} \qquad (2.2$$
$$k = 1,..,n-1$$

egyenletrendszerhez vezet.

Az értelmezési tartományt és ennek felosztását úgy is megválaszthatjuk, hogy a d_k szakaszok (a felosztás intervallumai) hossza 1 legyen. Bevezetve még az $f_{k-1} - 2f_k + f_{0+1} = h_k$ jelölést, a 2.29 rendszer a következő egyszerűbb formát ölti:

$$\begin{cases} 4a_1 + a_2 = h_1 \\ a_{k-1} + 4a_k + a_{k+1} = h_k \; ; \qquad k = 2, \ldots, n-2 \\ a_{n-2} + 4a_{n-1} = h_{n-1} \end{cases} . \tag{2.30}$$

Ez a rendszer a kiküszöbölés módszerét kétszer alkalmazva a következő módon oldható meg leggyorsabban: először egy-egy ismeretlent küszöbölünk ki, kezdve az első két egyenlettel és rendre kapjuk:

$$\begin{aligned} & q_1 = \frac{1}{4}; \quad \overline{h_1} = q_1 h_1 \\ & q_k = \frac{1}{4 - q_{k-1}}; \quad \overline{h_k} = q_k \left(h_k - \overline{h_{k-1}} \right); \quad k = 2, \ldots, n-1 \\ & a_k + q_k a_{k+1} = \overline{h_k}; \quad k = 1, \ldots, n-2 \\ & a_{n-1} = \overline{h_{n-1}} \end{aligned} . \tag{2.31}$$

mivel az utolsó ismeretlent kiszámítottuk, visszahelyettesítéssel az utolsó előtti egyenlettől rendre az elsőig:

$$a_k = \overline{h_k} - q_k a_{k+1} \; ; \qquad k = n-2, n-3, \ldots, 1 . \tag{2.32}$$

A bemutatott algoritmusban az egyes változók fölötti vízszintes vonal az illető változónak a számítások során adott új értékét jelenti.

A kapott együtthatókkal a keresett görbe pontjainak koordinátáit a

(2.33)

$$g(t) = a_k (t_{k+1} - t)^3 + a_{k+1} (t - t_{k+1})^3 + (f_k - a_k)(t_{k+1} - t) + (f_{k+1} - a_{k+1})(t - t_{k+1})$$
$$t_k \le t \le t_{k+1} \; ; \qquad k = 0, \ldots, n-1$$

előállításban számíthatjuk és a görbét úgy jelenítjük meg, hogy meghúzzuk a kontrollpontok közé eső ívek

$$t_j = t_k + \frac{j}{n_r}; \qquad j = 0,1,\ldots,n_r \tag{2.34}$$

felosztásnak megfelelő elemi húrjait.

Írjuk fel a zárt görbék előállítására szolgáló harmadfokú spline függvény együtthatóinak kiszámítását, ezúttal a vektoros felírást használva. Ez annyi egyenletrendszert jelent ahány dimenziós a tér, amelyikben a kontrollpontsor értelmezett, de, akárcsak a fennebb tárgyalt esetben az egyenletrendszerek egymástól függetlenek és külön-külön megoldhatók.

Legyen tehát p_k , $k = 0, 1, \ldots, n$ helyvektorokkal jellemzett kontrollpontsor, $\mathbf{r}[0, n] \to \Re^m$ ($m >= 2$) a keresett spline függvény. Az értelmezési tartomány $t_o = 0, t_1 = 1, \ldots, t_k = k, \ldots, t_n = n,$ felosztását használjuk.

A kontrollpontokban való folytonossági (2.35

- $r(t_k) = p_k$, $k = 0, 1,\ldots, n$ - annak feltétele, hogy a görbe átmenjen a kontrollpontokon
- $r'(t_k + 0) = r'(t_k - 0)$ - az érintő folytonosságának feltétele
- $r''(t_k + 0) = r''(t_k - 0)$ - a görbület folytonosságának feltétele

Térgörbék esetén (2.6 vektoros

$$\mathbf{r}'(t) = \mathbf{r}(x'(t), y'(t), z'(t)), \mathbf{r}''(t) = \mathbf{r}(x''(t), y''(t), z''(t)) \quad . \tag{2.36}$$

A periodikussági

$$\mathbf{r}(t_{n+1}) = \mathbf{p}_0 \;,\; \mathbf{r}'(t_{n+1} - 0) = \mathbf{r}'(t_0 + 0) \;,\; \mathbf{r}''(t_{n+1} - 0) = \mathbf{r}''(t_0 + 0) \tag{2.37}$$

A következő előállítás

$$\mathbf{r}(t) = \begin{cases} \mathbf{a}_k(t_{k+1}-t)^3 + \mathbf{a}_{k+1}(t-t_k)^3 + (\mathbf{p}_k - \mathbf{a}_k)(t_{k+1}-t) + (\mathbf{p}_{k+1} - \mathbf{a}_{k+1})(t-t_k) \\ \qquad t_k \le t \le t_{k+1} \qquad k = 0,1,\ldots,n-1 \\ \mathbf{a}_n(t_{n+1}-t)^3 + \mathbf{a}_0(t-t_n)^3 + (\mathbf{p}_n - \mathbf{a}_n)(t_{n+1}-t) + (\mathbf{p}_0 - \mathbf{a}_0)(t-t_n) \\ \qquad t_n \le t \le t_{n+1} \end{cases} \tag{2.38}$$

együtthatóit kell meghatározni. A 2.35 feltételekkel az egyenletrendszert kapjuk.

$$\begin{cases} \mathbf{a}_{k-1} + 4\mathbf{a}_k + \mathbf{a}_{k+1} = \mathbf{p}_{k-1} - 2\mathbf{p}_k + \mathbf{p}_{k+1} = \mathbf{h}_{k-1}\,; \quad k = 1,\ldots,n-1 \\ \mathbf{a}_{n-1} + 4\mathbf{a}_n + \mathbf{a}_0 = \mathbf{p}_{n-1} - 2\mathbf{p}_n + \mathbf{p}_0 = \mathbf{h}_{n-1} \\ \mathbf{a}_n + 4\mathbf{a}_0 + \mathbf{a}_1 = \mathbf{p}_n - 2\mathbf{p}_0 + \mathbf{p}_1 = \mathbf{h}_n \end{cases} \tag{2.39}$$

Ez a rendszer is megoldható, a kiküszöbölés módszerét háromszor alkalmazva. A következő együtthatókat vezetjük be:

$$\begin{aligned} &q_0 = \frac{1}{4}\,; \quad q_k = \frac{1}{4 - q_{k-1}} \\ &s_0 = \frac{1}{4}\,; \quad s_k = -s_{k-1}q_k \quad ; \quad k = 1,\ldots,n-2 \quad . \\ &q_{n-1} = \frac{1 - s_{n-2}}{4 - q_{n-2}}\,; \quad q_n = \frac{1}{4 - q_{n-1}}\,; \end{aligned} \tag{2.40}$$

A 2.39 rendszer szabadtagjait a következő módon újraszámítva:

$$\begin{aligned}
&\overline{\mathbf{h}_0} = q_0\mathbf{h}_0 \\
&\overline{\mathbf{h}_k} = q_k\left(\mathbf{h}_k - \overline{\mathbf{h}_{k-1}}\right); \quad k=1,\ldots,n-2 \\
&\overline{\mathbf{h}_{n-1}} = \frac{\mathbf{h}_{n-1} - \overline{\mathbf{h}_{n-2}}}{4 - q_{n-2}} \\
&\overline{\mathbf{h}_n} = q_n\left(\mathbf{h}_n - \overline{\mathbf{h}_{n-1}}\right)
\end{aligned} \tag{2.4I}$$

az

$$\begin{cases}
\mathbf{a}_k + q_{k-1}\mathbf{a}_{k+1} + s_{k-1}\mathbf{a}_0 = \overline{\mathbf{h}_{k-1}}; \quad k = 1,\ldots,n-1 \\
\mathbf{a}_n + q_{n-1}\mathbf{a}_0 = \overline{\mathbf{h}_{n-1}} \\
\mathbf{a}_0 + q_n\mathbf{a}_1 = \overline{\mathbf{h}_n}
\end{cases} \tag{2.42}$$

alakjához jutunk.

Az együtthatókat a

$$\begin{aligned}
&\overline{s_n} = \frac{1}{q_n}; \qquad \overline{s_0} = \frac{s_0 - \overline{s_n}}{q_0} \\
&\overline{s_k} = \frac{s_k - \overline{s_{k-1}}}{q_k} \qquad ; \qquad k = 1,\ldots,n-2;
\end{aligned} \tag{2.43}$$

és a szabadtagokat a

$$\begin{aligned}
\overline{\overline{\mathbf{h}}}_n &= s_n \overline{\mathbf{h}}_n \\
\overline{\overline{\mathbf{h}}}_0 &= \frac{\overline{\mathbf{h}}_0 - \overline{\overline{\mathbf{h}}}_n}{q_0} \\
\overline{\overline{\mathbf{h}}}_k &= \frac{\overline{\mathbf{h}}_k - \overline{\overline{\mathbf{h}}}_{k-1}}{q_k} \quad k = 1, \ldots, n-2
\end{aligned} \tag{2.44}$$

képletekkel újraszámítva a következőt kapjuk:

$$\begin{cases}
\mathbf{a}_1 + \overline{s_n}\mathbf{a}_0 = \overline{\overline{\mathbf{h}}}_n \\
\mathbf{a}_k + \overline{s_{k-2}}\mathbf{a}_0 = \overline{\overline{\mathbf{h}}}_{k-2}\,; \qquad k = 2, \ldots, n \\
\mathbf{a}_0 = \dfrac{\overline{\mathbf{h}}_{n-1} - \overline{\overline{\mathbf{h}}}_{n-2}}{q_{n-1} - \overline{\overline{s}}_{n-2}}
\end{cases}, \tag{2.45}$$

majd

$$\begin{cases}
\mathbf{a}_1 = \overline{\overline{\mathbf{h}}}_n - \overline{s_n}\mathbf{a}_0 \\
\mathbf{a}_k = \overline{\overline{\mathbf{h}}}_{k-2} - \overline{s_{k-2}}\mathbf{a}_0; \qquad k = 2, \ldots, n
\end{cases} \tag{2.46}$$

A változók fölötti egy vagy két vízszintes vonal itt is az illető változónak a számítások során első illetve második ízben adott új értékét jelenti.

A bemutatott módszert valósítja meg a GB_2D_IP.PAS és a GB_3D_IP.PAS nevű programok síkbeli illetve térbeli spline interpolációt használó részei.

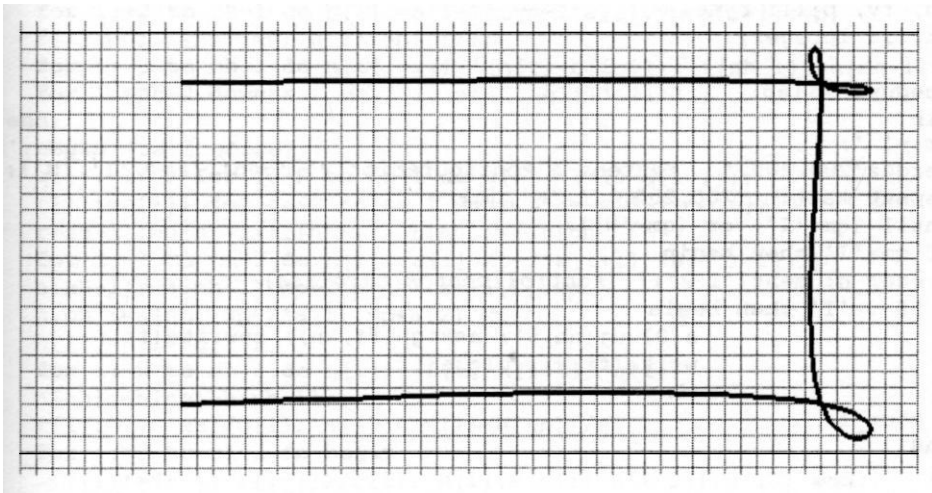
A spline interpoláció lényeges előnye a Coons-Hermite interpolációval szemben, hogy nem kell megadni az érintőket a kontrollpontokban, elég a kontrollpontok koordinátáit megadni. Bizonyos feladatok esetében hátrányos az a tulajdonsága, hogy egyetlen kontrollpont elmozdítása az egész görbe alakjára kihat, bár nagyobb számú kontrollpont esetén ez a hatás csak a szomszédos néhány intervallumon számottevő.

A *2.9 ábrán* egy spline interpolációval szerkesztett zárt síkgörbe látható, míg a *2.10. ábrán* egy zárt térgörbe, amely egy kocka csúcsain megy át.

Többszörös kontrollpontok esetén a spline görbéken hurkok keletkeznek, amint az a *2.11.ábrán* is látható, amelyen a jobb felső kontrollpont háromszoros, a jobb alsó pedig kétszeres (a kontrollpont-konfiguráció azonos a *2.14. ábrán* háthatóval).

2.11. ábra

A



GB_2D_IP.PAS nevű program sima síkgörbék különböző interpolációs illetve approximációs eljárással való szerkesztésére és megjelenítésére szolgál. **A** kontrollpont-konfigurációt billentyűzetről való koordinátabeírással vagy egérrel való mozgatással lehet meghatározni. A szerkesztési módszert változtatva ugyanarra a konfigurációra, a különböző módszereket össze lehet hasonlí**tani.** A megjelenítést lehetővé teszi a szerkesztési környezeten kívül is, a kontrollpontok tetszés szerinti megjelenítésével. A görbe nyílt vagy zárt voltát is ki lehet választani. A Coons-Hermite interpolációt a 2.3.3. alpontban leírt önműködő érintőszámítással végzi (a ko alakparaméter lekérdezésével együtt).

```
{$N+}

program gb_2d_ip; {sírna síkgörbeszerkesztés különböző approximációs
                   illetve interpolációs eljárással}

uses
  graph, grafind, crt, dos, kozos, file_kez, konf_2d, eger_kez, eger_kp,
  kerdesek, gomb_kez, kozos_gb;
```

```
const
   valasz: kars = ['B','E','G','Q','T','V'];
   modszer: kars =

var
   ff: file of mp;
   e, f: array[l..nm] of real;
   nh, fv, bill: byte;
   st, i: integer;

procedure kilep;
begin
   qm:='
   kerdez(200,200,'  Menteni a konfigurációt ?','gen','em','I','N');
   repeat kattint(200,200,'I','N',qm);
   until (qm='I') or (qm='N');
   if qm='I' then begin
    eger_n; ment (' IP2') ; {kontrollpontok tárolása}
    if v='I' then begin
                    a[0].uj(nh,0); assign(ff,kn); rewrite(ff);
                    write(ff,a); close(ff);
                  end;
    eger_l;
   end;
end;

procedure konfig;
begin
   repeat
    qf:=' ';
    k e r d e z ( 2 0 0 , 2 0 0 , '      K o n f i g u r á c i ó
    repeat kattint(200,200,'U','T',qf);
    until (qf='U') or (qf='T');
    if qf='T' then begin
      eger_n; olvas('IP2'); {tárolt kontrollpontok betöltése}
      if ko=1 then begin assign(ff,kn); reset(ff); read(ff,a);
                         close(ff);
                   end;
      eger_1;
    end else ko:=1;
   until ko=1;
end;

procedure kpontok; {kontrollpontok megjelenítése}
begin
   setcolor(15); for k:=1 to nh do a[k].megj;
end;

procedure spline; {splinefüggvények együtthatóinak kiszámítása)
var
   h, q, r: array [ 1 .. nm] of real;
begin
```

```
case qa of
'N':begin {nyílt görbe}
    q[2]:=1/4; for i:=3 to nh-i do q[i]:=1/(4-q(i-1]);
    for i:=2 to nh-1 do h[i]:=a[i-i].xp-2*a[i].xp+a[i+i].xp;
    e(1]:=0; e[nh] :=0; h[2] :=h[2] /4;
    for i:=3 to nh-1 do h[i]:=q[i]*(h[i]-h[i-1]); e[nh-1]:=h[nh-1];
    for i:=nh-2 downto 2 do e[i]:=h[i]-q[i]*e[i+l];
    for i:=2 to nh-1 do h[i]:=a[i-1].yp-2*a[i].yp+a[i+1].yp;
    f[i]:=0; f[nh]:=0; h[2]:=h[2]/4;
    for i:=3 to nh-1 do h[i]:=q[i]*(h[i]-h[i-1]); f[nh-1]:=h[nh-1];
    for i:=nh-2 downto 2 do f[i]:=h[i]-q[i]*f[i+l];
  end;
'Z':begin {zárt görbe}
    q[1]:=1/4; for i:=2 to nh-1 do q[i]:=1/(4-q[i-1]);
    r[1]:=q[1]; for i:=2 to nh-i do r[i]:=-r[i-1]*q[i];
    r[nh-1]:=q[nh-1]+r[nh-i]; r[nh]:=4-r[nh-1];
    for i:=1 to nh-2 do h[i]:=a[i].xp-2*a[i+1].xp+a[i+2].xp;
    h[nh-1]:=a[nh-1].xp-2*a[nh].xp+a[1].xp;
    h[nh]:=a[nh].xp-2*a[i].xp+a[2].xp;
    h[1]:=h[1]*q[1];
    for i:=2 to nh-1 do h[i]:=(h[i]-h[i-1])*q[i];
    h[nh]:=h[nh]-h[nh-1];
    r[1]:=1-r[nh]/q[1]; h[1]:=(h[1]-h[nh])/q[1];
    for i:=2 to nh-2 do begin
      r[i]:=(r[i]-r[i-1])/q[i]; h[i]:=(h[i]-h[i-1])/q[i]; end;
    e[1]:=(h[nh-1]-h[nh-2])/(r[nh-1]-r[nh-2]);
    for i:=nh downto 3 do e[i]:=h[i-2]-r[i-2]*e[1];
    e[2]:=h[nh]-r[nh]*e[1];
    r[1]:=q[1]; for i:=2 to nh-1 do r[i]:=-r[i-1]*q[i];
    r[nh-1]:=q[nh-1]+r[nh-1]; r[nh]:=4-r[nh-i];
    for i:=1 to nh-2 do h[i]:=a[i].yp-2*a[i+1].yp+a[i+2].yp;
    h[nh-1]:=a[nh-1].yp-2*a[nh].yp+a[1].yp;
    h[nh]:=a(nhJ.yp-2*a[1].yp+a[2].yp;
    h[1]:=h[1]*q[1];
    for i:=2 to nh-1 do h[i]:=(h[i]-h[i-1])*q[i];
    h[nh]:=h[nh]-h[nh-1];
    r[1]:=1-r[nh]/q[1]; h[1]:=(h[1]-h[nh])/q[1];
    for i:=2 to nh-2 do begin
      r[i]:=(r[i]-r[i-1])/q[i]; h[i]:=(h[i]¯h[i¯1])/q[i]; end;
    f[i]:=(h(nh-1]-h[nh-2])/(r[nh-i]-r[nh-2]);
    for i:=nh downto 3 do f[i]:=h[i-2]-r[i-2]*f[1];
    f[2]:=h[nh]-r[nh]*f[1];
  end;
end;
end;


  procedure rajz; {görbe megjelenítése}
  const r=50;
        nb=100;
var
        j: integer;
       u, u2, u3, w, z: real;
```

```
function xr(i, j: integer): integer; {görbepont x koordinátája}
begin
  case a Lj of
  '1':case qa of {Coons-Hermite interpoláció}
    'N':begin (nyílt görbe}
       u:=j/nr; u2:=u*u; u3:=u2*u;
       if i=1 then
              xr:=kx0+round((2*u3-3*u2+1)*a[1].xp+(-2*u3+3*u2)*a[2].xp+
              ((u3-2*u2+u)*(a[2].xp-a[1].xp)+(u3-u2)*(a[3].xp-
              a[1].xp))*k0)
       else if i=nh-1 then
            xr:=kx0+round((2*u3-3*u2+1)*a[nh-1].xp+(-2*u3+3*u2)*a[nh].xp+
                 ((u3-2*u2+u)*(a[nh].xp-a[nh-2].xp)+(u3-u2)*(a[nh].xp-
             a[nh-1].xp))*k0)
       else
              xr:=kx0+round((2*u3-3*u2+1)*a[i].xp+(-2*u3+3*u2)*a[i+1].xp+
                  ((u3-2*u2+u)*(a[i+l].xp-a[i-1].xp)+(u3-u2)*(a[i+2].xp-
              a[i].xp))*k0);
     end;
    'Z':begin (zárt görbe)
       u:=j/nr; u2:=u*u; u3:=u2*u;
       if i=1 then
              xr:=kx0+round((2*u3-3*u2+1)*a[1].xp+(-2*u3+3*u2)*a[2].xp+
              ((u3-2*u2+u)*(a[2].xp-a[nh].xp)+(u3-u2)*(a[3].xp-
              a[1] .xp)) *k0)
       else if i=nh-1 then
            xr:=kx0+round((2*u3-3*u2+1)*a[nh-1].xp+(-2*u3+3*u2)*a[nh].xp+
             ((u3-2*u2+u)*(a[nh].xp-a[nh-2].xp)+(u3-u2)*(a[1].xp-
             a[nh-l].xp))*k0)
       else if i=nh then
              xr:=kx0+round((2*u3-3*u2+1)*a[nh].xp+(-2*u3+3*u2)*a[1].xp+
              ((u3-2*u2+u)*(a[1].xp-a[nh-1].xp)+(u3-u2)*(a[2].xp-
              a[nh].xp))*k0)
       else
              xr:=kx0+round((2*u3-3*u2+1)*a[i].xp+(-2*u3+3*u2)*a[i+1].xp+
                   ((u3-2*u2+u)*(a[i+l].xp-a[i-1].xp)+(u3-u2)*(a[i+2].xp-
              a[i].xp))*k0);
     end;
    end;
  '3':begin (spline interpoláció)
       u:=j/nr; w:=1-u;
       case qa of
       'N':xr:=kx0+round(e[i]*w*w*w+e[i+1]*u*u*u+
                 (a[i].xp-e[i])*w+(a[i+1].xp-e[i+l])*u);
       'Z':if i=nh then
              xr:=kx0+round(e[nh]*w*w*w+e[1]*u*u*u+
                   (a[nh].xp-e[nh])*w+(a[1].xp-e[1])*u)
           else
              xr:=kx0+round(e[i]*w*w*w+e[i+1]*u*u*u+
                   (a[i].xp-e[i])*w+(a[i+1].xp-e[i+1])*u);
       end;
     end;
```

```
'4':case qa of (másodfokú B-spline közelítés}
  'N':begin (nyílt görbe)
      u:=j/nr;
      if nh=3 then
        xr;=kx0+round(sqr(1-u)*a[1].xp+2*u*(1-u)*a[2].xp+
             sqr(u)*a[3].xp);
      if i=2 then
        xr:=kx0+round(sqr(1-u)*a[1].xp+2*u*(1-u)*a[2].xp+
             sqr(u)*(a[2].xp+a[3].xp)/2)
      else if i=nh-1 then
        xr:=kx0+round(sqr(1-u)*(a[nh-2].xp+a[nh-1].xp)/2+
             2*u*(1-u)*a[nh-1].xp+sqr(u)*a[nh].xp)
      else
        xr:=kx0+round(sqr(1-u)*(a[i-1].xp+a[i].xp)/2+2*u*
             (1-u)*a[i].xp+sqr(u)*(a[i].xp+a[i+1].xp)/2);
   end;
  'Z':begin (zárt görbe}
      u: =j /nr;
      if i=0 then
        xr:=kx0+round(sqr(1-u)*(a[nh-1].xp+a[nh].xp)/2+2*u*
             (1-u)*a[nh].xp+sqr(u)*(a[nh].xp+a[1].xp)/2)
      else if i=1 then
        xr:=kx0+round(sqr(1-u)*(a[nh].xp+a[1].xp)/2+2*u*
             (1-u)*a[1].xp+sqr(u)*(a[1].xp+a[2].xp)/2)
      else
        xr:=kx0+round(sqr(1-u)*(a[i-1].xp+a[i].xp)/2+2*u*
             (1-u)*a[i].xp+sqr(u)*(a[i].xp+a[i+1].xp)/2);
   end;
  end;
'5':case qa of {harmadfokú B-spline közelítés}
  'N':begin (nyílt görbe)
      u:=j/nr; u2:=u*u; u3:=u2*u;
      if i=0 then
        xr:=kx0+round(((6-u3)*a[1].xp+u3*a[2].xp)/6)
      else if i=1 then
        xr:=kx0+round(((5-3*u-3*u2+2*u3)*a[1].xp+(1+3*u+3*u2-
             3*u3)*a[2].xp+u3*a[3].xp)/6)
      else if i=nh-1 then
        xr:=kx0+round(((1-3*u+3*u2-u3)*a[nh-2].xp+(4-6*u2+3*u3)*
             a[nh-1].xp+(1+3*u+3*u2-2*u3)*a[nh].xp)/6)
      else if i=nh then
        xr:=kx0+round(((1-3*u+3*u2-u3)*a[nh-1].xp+
             (5+3*u-3*u2+u3)*a[nh].xp)/6)
      else
        xr:=kx0+round(((1-3*u+3*u2-u3)*a[i-1].xp+(4-6*u2+3*u3)*
             a[i].xp+(1+3*u+3*u2-3*u3)*a[i+1].xp+u3*
             a[i+2].xp)/6);
   end;
  'Z':begin (zárt görbe)
      u:=j/nr; u2:=u*u; u3:=u2*u;
      if i=0 then
        xr:=kx0+round(((1-3*u+3*u2-u3)*a[nh-1].xp+(4-6*u2+3*u3)*
             a[nh].xp+(1+3*u+3*u2-3*u3)*a[1].xp+u3*a[2].xp)/6)
```

```
        else if i=1 then
          xr:=kx0+round(((1-3*u+3*u2-u3)*a[nh].xp+(4-6*u2+3*u3)*
           a[1].xp+(1+3*u+3*u2-3*u3)*a[2].xp+u3*a[3].xp)/6) else
           if i=nh-1 then
          xr:=kx0+round(((1-3*u+3*u2-u3)*a[nh-2].xp+(4-6*u2+3*u3)*
           a[nh-1i].xp+(1+3*u+3*u2-3*u3)*a[nh].xp+u3*a[1].xp)/6)
           else
          xr:=kx0+round(((1-3*u+3*u2-u3)*a[i-1].xp+(4-6*u2+3*u3)*
              a[i].xp+(1+3*u+3*u2-3*u3)*a[i+1].xp+u3*a[i+2].xp)/6);
      end;
     end;
  end;
end;

function yr(i, j: integer): integer; (görbepont y koordinátája)
begin
  case elj of
  '1':case qa of (Coons-Hermite interpoláció)
     'N':begin (nyílt görbe)
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=1 then
          yr:=ky0-round((2*u3-3*u2+1)*a[1].yp+(-2*u3+3*u2)*a[2].yp+
              ((u3-2*u2+u)*(a[2].yp-a[1]•yp)+(u3-u2)*(a[3].yp-
              a[1].yp))*k0)
        else if i=nh-1 then
          yr:=ky0-round((2*u3-3*u2+1)*a[nh-1].yp+(-2*u3+3*u2)*
              a[nh].yp+((u3-2*u2+u)*(a[nh].yp-a[nh-2].yp)+
              (u3-u2)*(a[nh].yp-a[nh-1].yp))*k0)
        else
          yr:=ky0-round((2*u3-3*u2+1)*a[i].yp+(-2*u3+3*u2)*a[i+1].yp+
              ((u3-2*u2+u)*(a[i+1].yp-a[i-1].yp)+(u3-u2)*(a[i+2].yp-
              a[i].yp))*k0);
      end;
     'Z':begin (zárt görbe)
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=1 then
          yr:=ky0-round((2*u3-3*u2+1)*a[1].yp+(-2*u3+3*u2)*a[2].yp+
              ((u3-2*u2+u)*(a[2].yp-a[nh].yp)+(u3-u2)*(a[3].yp-
              a[1].yp))*k0)
        else if i=nh-1 then
          yr:=ky0-round((2*u3-3*u2+1)*a[nh-1].yp+(-
           2*u3+3*u2)* a[nh].yp+((u3-2*u2+u)*(a[nh].yp-
           a[nh-2].yp)+(u3-u2)* (a[1].yp-a[nh-1].yp))*k0)
           else if i=nh then
          yr:=ky0-round((2*u3-3*u2+1)*a[nh].yp+(-2*u3+3*u2)*a[1].yp+
              ((u3-2*u2+u)*(a[1].yp-a[nh-1].yp)+(u3-u2)*(a[2].yp-
              a[nh].yp))*k0)
        else
          yr:=ky0-round((2*u3-3*u2+1)*a[i].yp+(-2*u3+3*u2)*a[i+l].yp+
              ((u3-2*u2+u)*(a[i+l].yp-a[i-1].yp)+(u3-u2)*(a[i+2].yp-
              a[i].yp))*k0);
     end;
    end;
```

```
'3':begin {spline interpoláció}
     u:=j/nr; w:=1-u;
     case qa of
     'N':yr:=ky0-round(f[i]*w*w*w+f[i+1]*u*u*u+
                 (a[i].yp-f[i])*w+(a[i+1].yp-f[i+1])*u);
     'Z':if i=nh then
           yr:=ky0-round(f[nh]*w*w*w+f[1]*u*u*u+
                  (a[nh].yp-f[nh])*w+(a[1].yp-f[1])*u)
         else
           yr:=ky0-round(f[i]*w*w*w+f[i+1]*u*u*u+
                 (a[i].yp-f[i])*w+(a[i+1].yp-f[i+1])*u);
     end;
   end;
'4':case qa of {másodfokú B-spline közelítés)
 'N':begin {nyílt görbe}
     u:=j/nr;
     if nh=3 then
       yr:=ky0-round(sqr(1-u)*a[1].yp+2*u*(1-u)*
           a[2].yp+sqr(u)*a[3].yp);
     if i=2 then
       yr:=ky0-round(sqr(1-u)*a[1].yp+2*u*(1-u)*a[2].yp+
           sqr(u)*(a[2]•yp+a[3]•yp)/2)
     else if i=nh-1 then
       yr:=ky0-round(sqr(1-u)*(a[nh-2].yp+a[nh-1].yp)/2+
           2*u*(1-u)*a[nh-1].yp+sqr(u)*a[nh].yp)
     else
       yr:=ky0-round(sqr(1-u)*(a[i-1].yp+a[i].yp)/2+2*u*(1-u)*
           a[i].yp+sqr(u)*(a[i].yp+a[i+1].yp)/2);
   end;
 'Z':begin {zárt görbe)
     U: =j /nr;
     if i=0 then
          yr:=ky0-round(sqr(1-u)*(a[nh-1].yp+a[nh].yp)/2+2*u*(1-u)*
           a[nh].yp+sqr(u)*(a[nh].yp+a[1].yp)/2)
     else if i=1 then
       yr:=ky0-round(sqr(1-u)*(a[nh].yp+a[1].yp)/2+2*u*(1-u)*
           a[1] .yp+sqr(u) * (a[1] .yp+a[2] .yp) /2)
     else
       yr:=ky0-round(sqr(1-u)*(a[i-1].yp+a[i].yp)/2+2*u*(1-u)*
           a[i].yp+sqr(u)*(a[i].yp+a[1+1).YP)/2);
   end;
 end;
'5':case qa of {harmadfokú B-spline közelítés}
 'N':begin {nyílt görbe}
     u:=j/nr; u2:=u*u; u3:=u2*u;
     if i=0 then                                          ,
       yr:=ky0-round(((6-u3)*a[1].yp+u3*a[2].yp)/6) else
     if i=1 then
          yr:=ky0-round(((5-3*u-3*u2+2*u3)*a[1].yp+(1+3*u+3*u2-3*u3)*
           a[2].yp+u3*a[3].yp)/6)
     else if i=nh-1 then
       yr:=ky0-round(((1-3*u+3*u2-u3)*a[nh-2].yp+(4-6*u2+3*u3)* a[nh-
           1].yp+(1+3*u+3*u2-2*u3)*a[nh].yp)/6)
```

```
        else if i=nh then
          yr:=ky0-round(((1-3*u+3*u2-u3)*a[nh-1].yp+
               (5+3*u-3*u2+u3)*a[nh].yp)/6)
        else
          yr:=ky0-round(((1-3*u+3*u2-u3)*a[i-1].yp+(4-6*u2+3*u3)*
               a[i].yp+(1+3*u+3*u2-3*u3)*a[i+i].yp+u3*a[i+2].yp)/6);
      end;
    'Z':begin {zárt görbe}
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=0 then
          yr:=ky0-round(((1-3*u+3*u2-u3)*a[nh-1].yp+(4-6*u2+3*u3)*
              a[nh].yp+(1+3*u+3*u2-3*u3)*a[1].yp+u3*a[2].yp)/6)
        else if i=1 then
          yr:=ky0-round(((1-3*u+3*u2-u3)*a[nh].yp+(4-6*u2+3*u3)*
              a[1].yp+(1+3*u+3*u2-3*u3)*a[2].yp+u3*a[3].yp)/6)
        else if i=nh-1 then
          yr:=ky0-round(((1-3*u+3*u2-u3)*a[nh-2].yp+(4-6*u2+3*u3)*
               a[nh-1].yp+(1+3*u+3*u2-3*u3)*a[nh].yp+u3*a[1].yp)/6)
        else
          yr:=ky0-round(((1-3*u+3*u2-u3)*a[i-1].yp+(4-6*u2+3*u3)*
              a[i].yp+(1+3*u+3*u2-3*u3)*a[i+i].yp+u3*a[i+2].yp)/6);
      end;
    end;
  end;
end;
begin
  eger_n;
  racs2; setlinesty1e(O,O,3); setcolor(15);
  if qa='N' then moveto(a[1].xr,a[1].yr);
  case elj of
  '1':begin (Coons-Hermite interpoláció}
      if qa='Z' then moveto(xr(1,0),yr(1,0));
     for i:=1 to nh-1 do for j:=1 to nr do lineto(xr(i,j),yr(i,j));
      if qa='Z' then for j:=1 to nr do lineto(xr(nh,j),yr(nh,j));
    end;
  '2':if qa='N' then begin {Bézier-közelítés}
        for j:=0 to nb-1 do begin
          u:=j/nb; u2:=1-u; u3:=u2; for 1:=2 to nh-i do u3:=u3*u2;
          w:=u3*a[1].xp; z:=u3*a[1].yp;
          for i:=2 to nh do begin
            u3:=u3*(nh-i+l)/(i-1)*u/(1-u);
            w:=w+u3*a[i].xp;
            z:=z+u3*a[i].yp;
          end;
          lineto(kx0+round(w),ky0-round(z));
        end;
        lineto(a[nh].xr,a[nh].yr);
      end;
  '3':case qa of (spline interpoláció)
    'N':for is=1 to nh-1 do
          for j:=1 to nr do lineto(xr(i,j),yr(i,j));
```

```
    'Z':begin
          moveto(xr(1,0),yr(1,0));
          for i:=1  to. nh do
            for j:=1 to nr do lineto(xr(i,j),yr(i,j));
      end;
    end;
  '4':begin (másodfokú B-spline közelítés}
      if qa='Z' then begin moveto(xr(0,0),yr(0,0));
                  for i:=0  to 1 do for j:=1  to nr do lineto(xr(i,j),yr(i,j));
end;
      for i:=2 to nh-1 do
        for j:=1 to nr do lineto(xr(i,j),yr(i,j));
    end;
  '5':begin {harmadfokú B-spline közelítés}
      if qa='Z' then moveto(xr(0,0),yr(0,0));
      for i:=0 to nh-1 do for j:=1 to nr do lineto(xr(i,j),yr(i,j));
      if qa='N' then for j:=1 to nr do lineto(xr(nh,j),yr(nh,j));
    end
  end;
  setlinestyle(0,0,1);
  eger_l;
end;

procedure tar; {kontrollponttár - ahonnan új pontot lehet egérrel
                elvenni}
begin
  k:=nh+1; a[k].kezd;
end;

begin
  grindhi; ker_2; eger_k; mm:=imagesize(0,0,2*r,2*r); ef:=0;
  repeat
    qu:='N'; fe:=0;
    for k:=0 to nm do a[k].uj(0,0); qk:='N';
    eger_n; racs2; konfig; nh:=a[0].xp; parancs(1); eljaras;
    eger_n; racs2; eger l;
    if nh>0 then (kezdőkonfiguráció megjelenítése}
      for k:=1 to nh do with a[k] do begin
        getmem(q,mm); (helyfoglalás a bitmap-nek a heap-ben)
        getimage (xr-r, yr-r, xr+r, yr+r, q^) ;
        fg:=0; megj;
    end;
    tar;
    repeat
      egert(kx0,10,620,ky0); eger_1; qr:=' '; k:=0;
      bill_ =1; {billentyűzet aktív}
      repeat eger_f; until (ef=1) or (keypressed) or gombny(valasz);
      if of=1 then bill:=0; (billentyűzet az elengedésig nem aktív)
      repeat eger_v; eger_e; until ef=0;
      if k=nh+1 then begin if a[k].bent then nh:=nh+1; end
                  else if not (a[k].bent) then begin
                                    {aktív mezőről lehúzott kontrollpont törlése}
        eger_n;
```

```
      for sz:=k+1 to nh do a[sz-1]:=a[sz];
      nh:=nh-1; end;
   if bill=1 then begin
      repeat par_katt(1,qr); until qr in valasz; eger_n;
      case qr of
        'B':if nh<nm then begin {új kontrollpont beszúrása}
            gb[6]:=1; parancs(1);
            sorszam;
            if sz>0 then begin
              if sz>nh then sz:=nh+1;
              if sz<=nh then for k:=nh downto sz do a[k+1]:=a[k];
              koordin; a [sz] .uj (x, y) ; a [sz] .koord;
              nh:=nh+1;
            end;
            gb[6]:=0; lap(208,5); fe:=0;
          end;
        'V':begin {váltortatandó kontrollpont kiválasztása}
            gb[1]:=1; parancs(1); sorszam;
            if (sz>0) and (sz<=nh) then begin
              vlt;
              case qr of
              'S':begin {sorszám váltortatása}
                    st:=sz; a[0]:=a[sz]; sorszam;
                    if (sz>st) and (sz<=nh) then
                      for k:=st to sz-1 do a[k]:=a[k+1];
                    if (sz<st) and (sz>0) then
                      for k:=st downto sz+1 do a[k]:=a[k-1];
                    a[sz]:=a[0]
                  end;
              'H':begin {hely váltortatása}
                    koordin; a [sz] . uj (x, y) ; a [sz] .koord; end;
              end;
            end;
            gb[1]:=0; lap(208,5); fe:=0;
          end;
      'T':if nh>0 then begin {kontrollpont törlése}
            gb[5]:=1; parancs(1); sorszam;
            if sz>0 then begin
              if sz<nh then for k:=sz+1 to nh do a[k-1]:=a[k];
              if sz<=nh then nh:=nh-1;
            end;
            gb[5]:=0; lap(208,5); fe:=0;
          end;
      'G':if nh>2 then begin {görbemegjelenítés}
            gb[3]:=1; parancs(1); alak;
            eqer_t(5,5,200,50); eger_l; elj:=' ';
            repeat elj katt(elj); {eljárásválasztás}
            until elj in modszer; eger n;
            for 1:=7 to 10 do gb[1]:=0;
            case elj of
            '1':begin gb[7]:=1; egyutth; lap(208,5);
                  str(k0:3:2,kn); outtextxy(262,23,'k0 = '+kn); end;
```

```
                '2':begin gb[8]:=1; end;
                '3':begin spline; gb[9]:=1; end;
                '4':begin b_spline; gb[10]:=1; end;
                end;
                eljaras; rajz;
                gb[3]:=2; fe:=1;
              end;
          'E':if fe=1 then begin {görbemegjelenítés keret nélkül)
                gb[2]:=1; parancs(1); kpl; cleardevice;
                if qm='I' then begin racs2; rajz; kpontok; end
                          else rajz;
                setcolor(7); outtextxy(5,5,'ESC - vissza');
                repeat qm:=readkey; until ord(qm)=27;
                ker_2; eljaras;
                gb[2]:=0;
              end;
          'Q':begin (kilépés)
                gb[4]:=1; parancs(1);
                repeat kilep until not ((v='N') and (qm='I'));
                vege; if qk='N' then ujkonf; gb[4]:=0;
              end;
        end; .
      end;
      if gb[3]=0 then racs2; kpontok; gb[3]:=0; parancs(1); eger_1;
      tar;
    until (qk='I') or (qu='I');

    for k:=1 to nh do dispose(a[k].q);
         {a bitmap-memóriák felszabadítása a heap-ben)
  until qu='N';
  closegraph;
end.
```

2.3.5. Bézier approximáció

A Bézier közelítés kifejlesztésének célja sima görbék és felületek több kontrollpont segítségével való szerkesztése volt, hogy megkönnyítse a számítógépes formatervezést. A Renault műveknél gépjárműkarosszéria tervezésre 1972 óta használják.

A 2.9 összefüggésben szereplő keverő- vagy súlyfüggvények a következő feltételeknek tesznek eleget:

- a görbe átmegy a szélső pontokon (po és p_n)
- az érintő a szélső pontokban p_1 - p_0 illetve p_{n-1} - p_n legyen
- a magasabb rendű deriváltak is hasonló módon álljanak elő, például a második deriváltat po pontban a $\mathbf{p}_0$, $\mathbf{p}_1$ és p_2, a p_n pontban pedig a p_{n-2}, p_{n-1} és p_n pontok határozzák meg
- a súlyfüggvények szimmetrikusak legyenek *u* és *1—u* -ra nézve, azaz a sorrend megfordítása ne befolyásolja a görbe alakját

A Bernstein polinomok kielégítik ezeket a feltételeket. Alakjuk:

$$B_{k,t} \quad \frac{n!}{k!(n-k!} \; u^k (1-u)^{n-k} \qquad u \in [0,1] . \tag{2.47}$$

A Bézier görbék előállítása tehát:

$$\mathbf{r}(u) = \sum_{k=0}^{n} B_{k,n}(u) P_k . \tag{2.48}$$

A Bézier közelítés további előnyös tulajdonsága, hogy minden kontrollpont-konfiguráció esetén a görbe benne van a karakterisztikus keret konvex burkában.

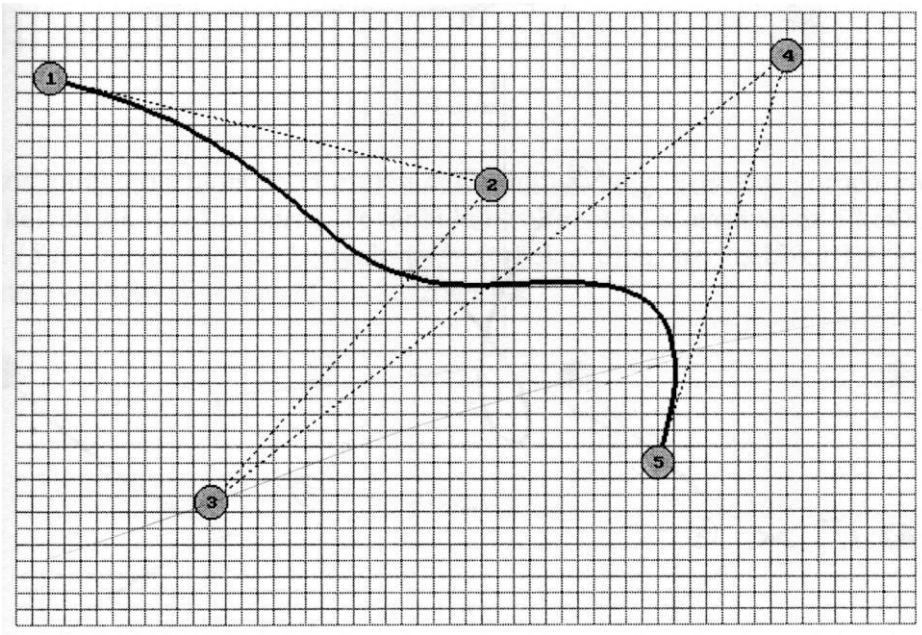
Hátránynak tudható be az a tény, hogy a keverő polinomok foka a kontrollpontok számával nő és ezek az egész értelmezési intervallumon különböznek **nul**lától, vagyis egyetlen pont elmozdítása az egész görbe alakjára kihat.

A gyakorlatban a helyi alakmódosítást úgy valósítják meg, hogy harmadfokú Bézier-görbéket szerkesztenek (4 - 4 kontrollponttal) és ezeket illesztik. A köbös Bézier görbék előállítása:

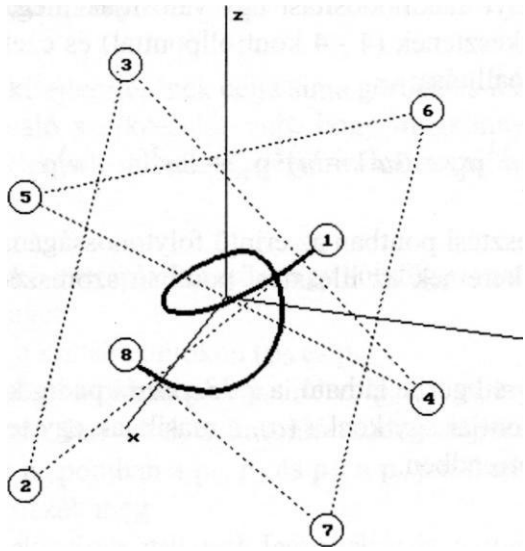
$$\mathbf{r}(u) = (1-u)^3\mathbf{p}_0 + 3u(1-u)^2\mathbf{p}_1 + 3u^2(1-u)\mathbf{p}_2 + u^3\mathbf{p}_3 \quad . \qquad (2.49$$

Az illesztés és az illesztési pontban az érintő folytonosságának feltétele, hogy a két karakterisztikus keretnek az illesztési pontban szomszédos szakaszai kolI ineári sak legyenek.

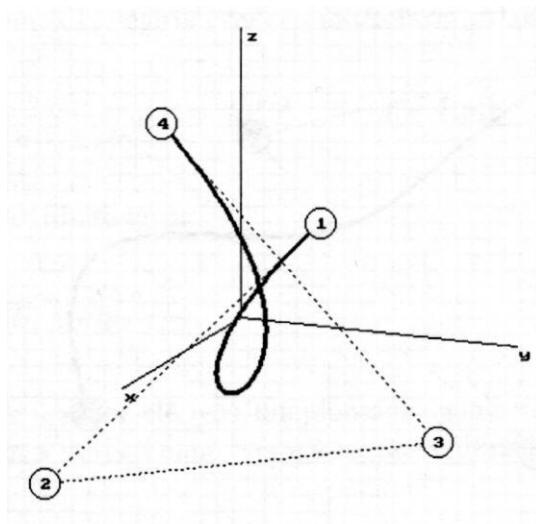
A *2.12. ábrán* Bézier-síkgörbe látható, a *2.13. ábrán* pedig két Bézier-térgörbe, egyiknek a kontrollpontjai egy kocka *(a),* a másiknak egy tetraéder *(b)* csúcsai az ábrákon látható sorrendben.



2.12. ábra



a

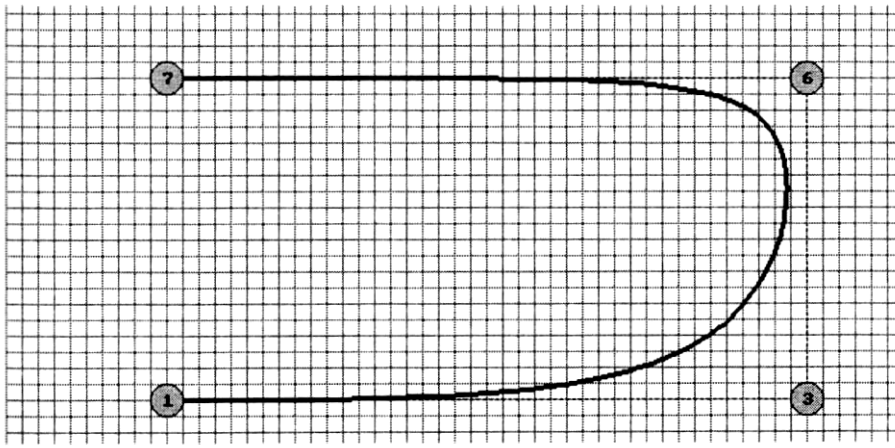


b

2.13.

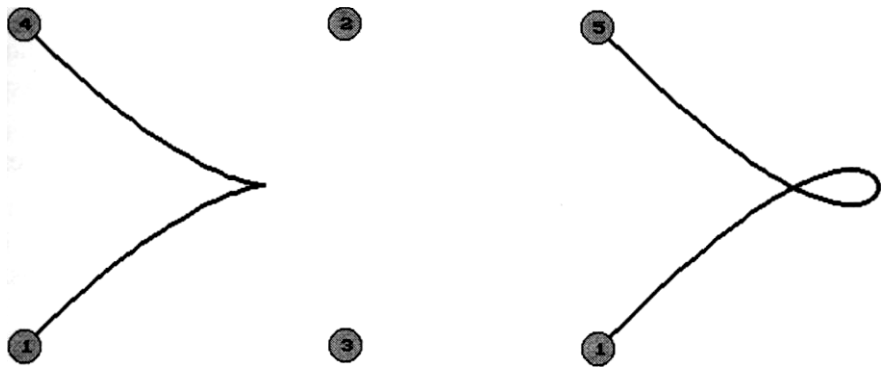
A *2.14. ábra* többszörös kontrollpontok hatását szemlélteti. Megfigyelhető ezeknek a görbe alakjára gyakorolt "vonzása", abban az értelemben, hogy minél több kontrollpont van ugyanabban a pontban, annál jobban megközelíti a

érbe az illető pontot (mivel a görbe előállításában - 2.48 képlet - a többszörös pont helyvektorát több súlyfüggvény összege szorozza). A *2.14. ábrán* a 3-mal **jelölt** kontrollpont kétszeres (a 2-es és 3-as pontok ugyanott állnak), a 6-tal **jelölt** pedig háromszoros (a 4., 5. és 6. kontrollpont egybeesik).



2.14. ábra

Hurkos Hurkos karakterisztikus keret esetén előfordulhat, hogy a Bézier görbén szög- pont keletkezik *(2.15.a ábra)*. További kontrollpont beiktatásával ez elkerülhető *(2.15.b ábra),* viszont bonyolultabb görbéket több, harmadfokú Bézier-ív illesztésével kényelmesebben szerkeszthetünk.



a b

2.15. ábra

2.3.6. B-spline approximáció

A görbealak helyi ellenőrzése és az úgynevezett globális simaság követelményeit egyidejűleg a B-spline közelítés valósítja meg a legjobban. Ezért a grafikus számítógépes tervezési szoftvercsomagok többsége ezen a módszeren alapul. A B-spline görbék előállítása:

$$\mathbf{r}(u) = \sum_{k=0}^{n} N_{k,i}(u)\mathbf{p}_k \ , \qquad (2.5$$

az u paraméter $[a, b]$ értelmezési tartományának $a = t_i <= t_{i+1} <= \ldots <= t_{n+1} = b$ felosztása mellett. Az $N_{k,i}$, i-ed fokú, úgynevezett B-spline blending- vagy súlyfüggvények rekurzív értelmezése:

, (2.51)

$$N_{k,0}(u) = \begin{cases} 1 & u \in [t_k, t_{k+1}) \\ 0 & u \notin [t_k, t_{k+1}) \end{cases}$$

$$N_{k,l}(u) = \frac{u - t_k}{t_{k+l} - t_k} N_{k,l-1}(u) + \frac{t_{k+l+1} - u}{t_{k+l+1} - t_{k+1}} N_{k+1,l-1}(u) \quad l = 1,2,\ldots,i$$

amelyhez további $b = t_{n+1} <= t_{n+2} <= \ldots <= t_{n+i+1}$ alappontokra van szükség. A rekurziós képletben azok a tényezők, amelyeknek nevezőjében 0 adódna, értelmezés szerint 0 értékűek.

A helyi ellenőrzést az a tény teszi lehetővé, hogy minden B-spline súlyfüggvény csak a felosztás i egymás utáni intervallumán nem nulla. Ezért egy kontrollpont csak a szomszédos (mindkét oldalon) i intervallumra gyakorol hatást.

A szélső alappontok meghatározása a kívánt görbealaktól függ, a következő-képpen:

- adott végpontú nyílt görbe esetén az értelmezési tartomány végein többszörös alappontok állnak:

(2.52)

$$\begin{cases} t_0 = t_1 = \ldots = t_i = a \\ t_{n+1} = t_{n+2} = \ldots = t_{n+i+1} = b \end{cases} ,$$

ezáltal a görbe végpontjai a karakterisztikus keret végein álló i+1 - szeres kontrollpontokba kerülnek:

$$\begin{cases} \mathbf{p}_0 = \mathbf{p}_1 = \ldots = \mathbf{p}_i \\ \mathbf{p}_{n+1} = \mathbf{p}_{n+2} = \ldots = \mathbf{p}_{n+i+1} \end{cases} . \qquad (2.53$$

- zárt görbék periodikus alappontokkal

$$\begin{cases} t_l = t_{n-i+l+1} + a - b \\ t_{n+l} = t_{i+l-1} + b - a \end{cases} \quad l = 0,1,\ldots,i \ , \qquad (2.5$$

feltéve, hogy a karakterisztikus keret is kielégíti

$$\mathbf{p}_{n-i+l+1} = \mathbf{p}_l \quad l = 0,1,\ldots,i \qquad (2.55$$

A B-spline függvények szakaszonként értelmezett polinomok, amelyeknek foka (i) nem függ a kontrollpontok számától, hanem a felhasználó szabja meg. Legelterjedtebbek a másod- és harmadfokú B-spline közelítések.

A B-spline közelítés esetében is érdemes egyenletes paraméterfelosztást hasz- nálni, vagyis a kontrollpontokhoz rendelt paraméterértéknek megfelelő intervallumokat egyenlő hosszúaknak venni. Ezt a hosszúságot egységnyinek is vehetjük. Egyenletes paraméterfelosztás mellett a végpontok környezetének kivé- telével a súlyfüggvények alakja azonos, csupán el vannak tolva egymáshoz viszonyítva. Ez a tény vezetett el az úgynevezett periodikus B-spline függvények bevezetéséhez, alkalmas változócsere segítségével.

A másodfokú periodikus B-spline függvények segítségével előállított approximációs görbeív:

$$\mathbf{r}(u) = \frac{u^2 - 2u + 1}{2}\mathbf{p}_{k-1} + \frac{-2u^2 + 2u + 1}{2}\mathbf{p}_k + \frac{u^2}{2}\mathbf{p}_{k+1} \ ; \qquad u \in [0,1] \qquad (2.56)$$

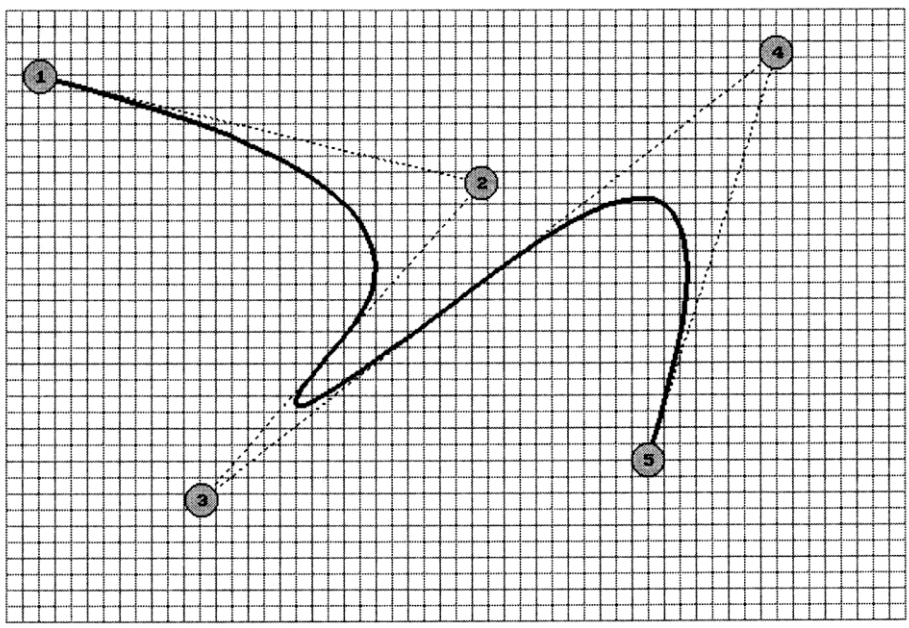
tulajdonképpen a $\boldsymbol{p}_{k-1}\boldsymbol{p}_k$ és a $\boldsymbol{p}_{k-1}\boldsymbol{p}_{k+1}$ szakaszok felezőpontját összekötő parabolaív, amelynek az említett szakaszok a végpontjaiban érintői. Ez az alábbi, a 2.56 képlettel egyenértékű felírásból is látszik:

$$\mathbf{r}(u) = (1-u)^2 \frac{\mathbf{p}_{k-1} + \mathbf{p}_k}{2} + 2u(1-u)\mathbf{p}_k + u^2 \frac{\mathbf{p}_k + \mathbf{p}_{k+1}}{2}; \qquad u \in [0,1] \tag{2.57}$$

Az egész görbét úgy állítjuk elő, hogy a *k* index értékét növelve megkapjuk a karakterisztikus keret szakaszainak középpontjait összekötő, ezen szakaszokat érintő parabolaíveket.

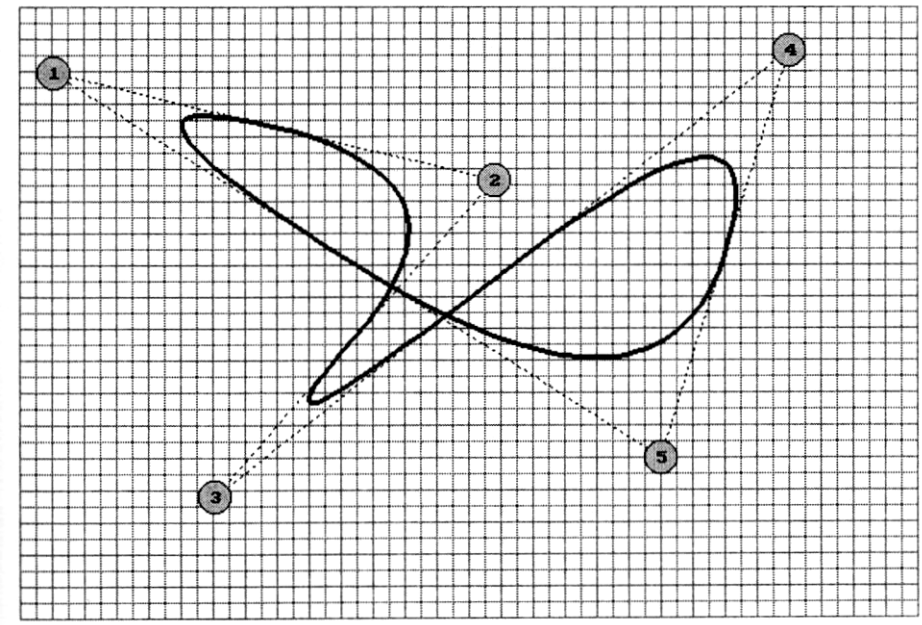
Nyílt görbéket úgy rajzolhatunk, hogy a végpontokba annyiszor többszörös kontrollpontot teszünk, ahányad fokú a B-spline közelítés. Másodfokú B-spline esetén a szélső ívek olyan parabolák, amelyek a szélső kontrollpontot a következő kettőt összekötő szakasz felezőpontjával kötik össze *(2.16. ábra)*.

2.16. ábra



Zárt görbék rajzolásánál az utolsó kontrollpontot az elsővel összekötő szakasz is a karakterisztikus kerethez tartozik, az előállítás pedig minden ívre azonos a görbék belsejére vonatkozóan leírtakkal.

A *2.17. ábrán* látható a *2.16. ábra* kontrollpontjait összekötő, ezúttal zárt másodfokú B-spline síkgörbe. A *2.18. ábrán* két másodfokú B-spline térgörbe látható, kocka, illetve tetraéder csúcspontjaiban fekvő kontrollpontokkal.



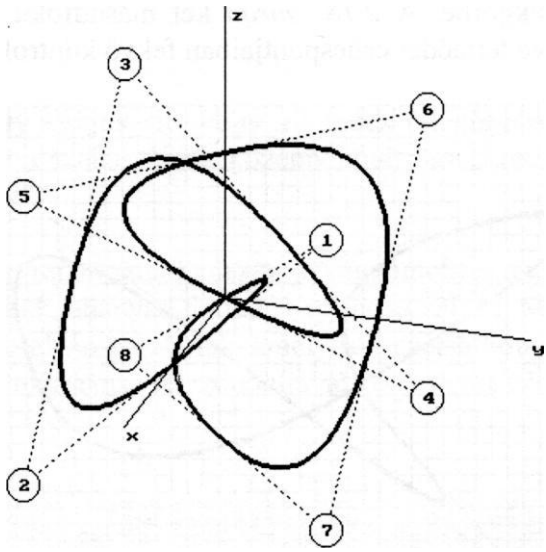
2.17. ábra

A harmadfokú periodikus B-spline függvények segítségével előállított görbeív:

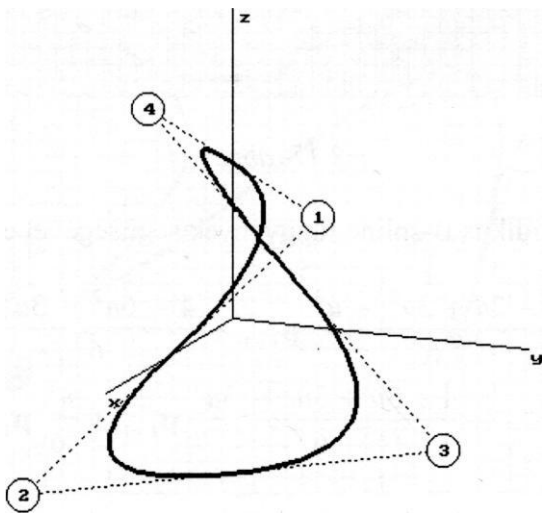
$$\mathbf{r}(u) = \frac{1 - 3u + 3u^2 - u^3}{6}\mathbf{p}_{k-1} + \frac{4 - 6u^2 + 3u^3}{6}\mathbf{p}_k + \\ + \frac{1 + 3u + 3u^2 - 3u^3}{6}\mathbf{p}_{k+1} + \frac{u^3}{6}\mathbf{p}_{k+2}\ ; \qquad u \in [0,1] \tag{2.58}$$

Az egész görbét itt is úgy állítjuk elő, hogy a ***k*** index értékét növelve megkap**juk** az egymáshoz másodrendű folytonosan (folytonos görbülettel) illeszkedő íreket.

Nyílt görbék rajzolásakor a végpontokba háromszoros kontrollpont kerül, zárt görbék esetén pedig a kontrollpontsort ciklikusan be kell zárni. Ez azt jelenti, hogy az első 3 kontrollpont azonos az utolsó hárommal (fordított sorrendben), az eredeti *n* kontrollpont helyett n+2 -t tekintve.



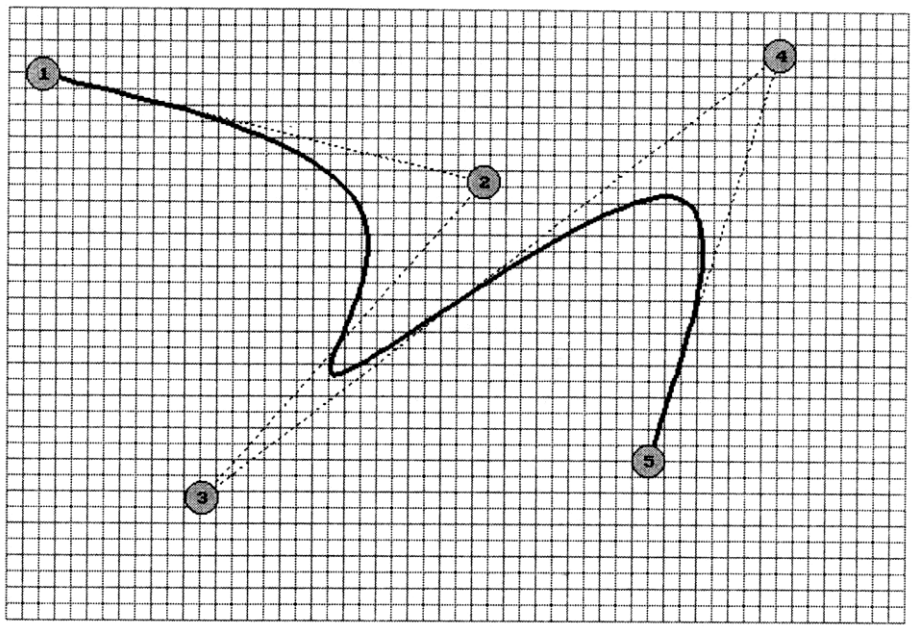
a



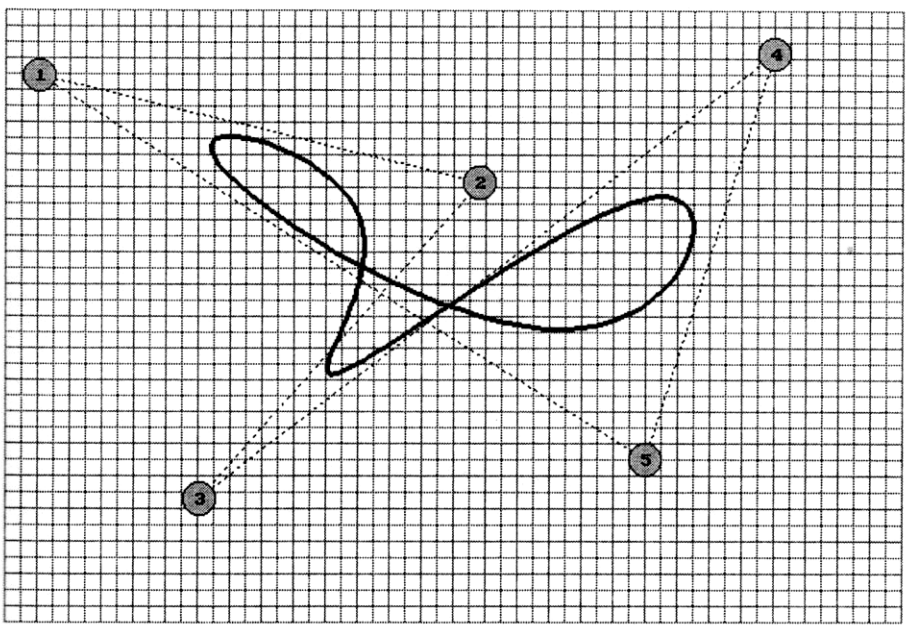
b

2.18. ábra

2.19. ábra

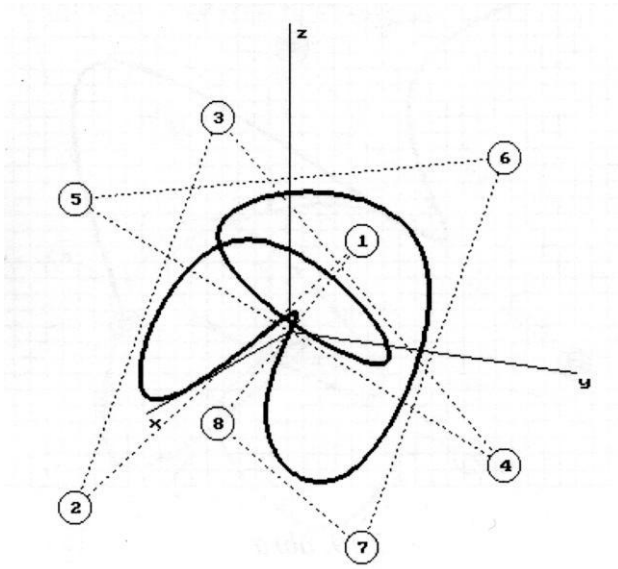


2.20. ábra

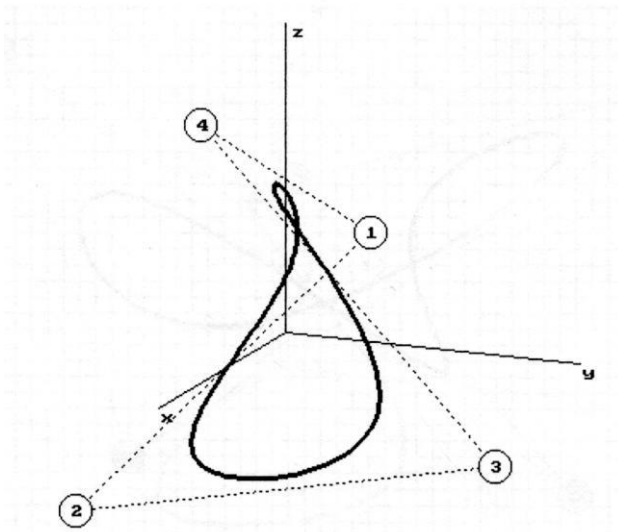


A *2.19.* ábrán látható a *2.16. ábra* kontrollpontjait összekötő nyílt-, a *2.20_ ábrán* pedig a zárt harmadfokú B-spline síkgörbe. A *2.21. ábrán* két harma fokú B-spline térgörbelátható, a *2.18. ábra* kontrollpontjaival.

2.21. ábra



a



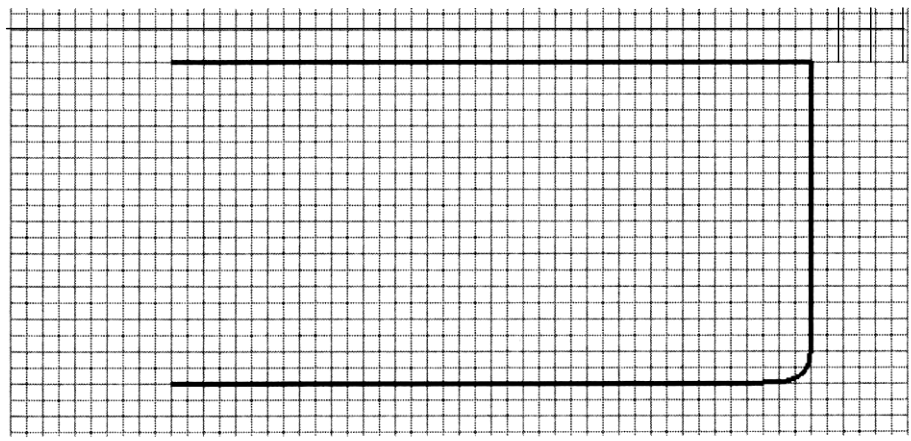
b

A B-spline görbék tulajdonságai

- egy i -ed fokú B-spline görbe átmegy az i -szeres (vagy többszörös) kontrollpontokon
- ha i +1 kontrollpont kollineáris, akkor az i -ed fokú B-spline görbe részben az általuk meghatározott szakaszon fekszik. Másodfokú B-spline görbék esetén, ha a karakterisztikus keret két egymás melletti szakasza kollineáris, a két szakasz felezőpontjait összekötő szakasz a görbéhez tartozik.
- minden B-spline görbe benne van a karakterisztikus keretének konvex burkában

A másodfokú B-spline közelítés előnye az előreláthatóbb viselkedés, a harmadfokúé pedig a jobb simaság (folytonos görbület). Ezért a legtöbb szövegszerkesztő program az arányosan méretezhető betűkészleteket (true type fonts) másodfokú B-spline-ok segítségével szerkeszti.

A *2.22. ábrán* harmadfokú B-spline görbe viselkedése figyelhető meg kétszeres (jobb alsó sarok) és háromszoros (jobb felső sarok) kontrollpont környezetében. A konfiguráció azonos a *2.14. ábrán* láthatóval.



2.22. ábra

A kontrollpontok és a térgörbék illetve felületek pontjainak képsíkra vetítését, a vetítési irány változtatását, a koordinátarendszer megjelenítését és a kontrollpontok koordinátáinak betáplálását végző eljárások a KOZOS_3D.PAS nevű unit-ban találhatók.

```
{$N+}
unit kozos 3d;

interface

uses graph, kozos;

const
  kx0=31; ky0=460; kxl=kx0+300; kyl=ky0-200;
  fel = #72; le = #80; bal = #75; jobb = #77; alf=pi/60; bet=pi/60;
  tgh = 200;
  xm = 600; ym = 400;

var
  kr: char;
  xt, yt: integer;
  ap, bp, cp, nvl, nv2: single;

procedure vetit(x,y,z: single);
procedure koordin3(sz: string);
procedure forgat;
procedure nyíl;
procedure racs;

implementation

procedure vetit(x,y,z: single);
    {képsíkravetítés és képernyőkoordináták kiszámítása}
begin
  xt:=kxl+round((-by*x+ap*y)/nvl);
  yt:=ky1-round((-cp*(ap*x+bp*y)+sqr(nvl)*z)/nv2);
end;

procedure koordin3(sz: string); {térbeli koordináták megadása}
begin
  lap(208,5);
  setcolor(0); outtextxy(300,13,sz); setcolor(13);
  outtextxy(215,30,'x = '); outtextxy(295,30,'y = ');
  outtextxy(375,30,'z = ');
  ablak(242,27,280,39); ablak(322,27,360,39); ablak(402,27,440,39);
  repeat kn:=''; 1:=1; iras(5,238,29); val(kn,x,hk);
    if hk<>0 then ablak(240,27,282,39); until (hk=0) or (qr=chr(27));
  if qr<>chr(27) then
  repeat kn:= ''; 1:=1; iras(5,318,29); val(kn,y,hk);
    if hk<>0 then ablak(320,27,362,39); until (hk=0) or (qr=chr(27));
  if qr<>chr(27) then
  repeat kn:=''; 1:=1; iras(5,398,29); val(kn,z,hk);
    if hk<>0 then ablak(400,27,442,39); until (hk=0) or (qr=chr(27));
end;
```

```
procedure forgat; {vetítési irány módosítása}
var au, bu, cu, r: single;
begin
  r:=nv1;
  case kr of
  fel: begin au:=ap*cos(bet)+ap*cp/nvl*sin(bet);
         bu:=bp*cos(bet)+bp*cp/nvl*sin(bet); cu:=cp*cos(bet)-
         nvl*sin(bet); end;
  le: begin au:=ap*cos(bet)-ap*cp/nvl*sin(bet);
         bu:=bp*cos(bet)-bp*cp/nvl*sin(bet);
         cu:=cp*cos(bet)+nvl*sin(bet); end;
  jobb: begin au:=ap*cos(alf)+bp*sin(alf);
         bu:=-ap*sin(alf)+bp*cos(alf); cu:=cp; end;
  bal: begin au:=ap*cos(alf)-bp*sin(alf);
         bu:=ap*sin(alf)+bp*cos(alf); cu:=cp; end;
  end;
  if (au*ap>0) or (bu*bp>0) then begin ap:=au; bp:=bu; cp:=cu;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp)); end;


  procedure nyil;
  const
  xk=50; yk=40;
  begin
  setcolor(7); outtextxy(xk-35,yk+45,'ESC - kilép');
  line (xk-5, yk, xk-30, yk) ; line (xk-30, yk, xk-26, yk+2) ; line (xk-30, yk, xk-
       26,yk-2);
  line (xk, yk-5, xk, yk-30) ; line (xk, yk-30, xk+2, yk-26) ; line (xk, yk-30, xk-
       2, yk-26) ;
  line(xk+5,yk,xk+30,yk); line(xk+30,yk,xk+26,yk+2);
  line (xk+30, yk, xk+26, yk-2) ;
  line (xk, yk+5, xk, yk+30) ; line (xk, yk+30, xk+2, yk+26) ; line (xk, yk+30, xk-
       2, yk+26) ;
  end;


  procedure racs; {koordinátarendszer megjelenítése}
  var x, y: integer;
  begin
  setfillstyle(1,0); bar(kx0-2,ky0-402,kx0+602,ky0+2); setcolor(15);
  if fe=0 then begin
    x:=kxl+round(-bp*tgh/nvl); y:=ky1-round(-cp*ap*tgh/nv2);
    line(kxl,kyl,x,y); outtextxy(x+2,y+2,'x');
    x:=kxl+round(ap*tgh/nvl); y:=ky1-round(-cp*bp*tgh/nv2);
    line(kxl,kyl,x,y); outtextxy(x+2,y+2,'y'); y:=kyl-
    round(sqr(nvl)*tgh/nv2);
    line(kxl,kyl,kxl,y);       outtextxy(kx1+5,y+2,'z');       end;
  end;
  end;
```

Sima térgörbéket lehet szerkeszteni és megjeleníteni az alábbi, GB3D_IP.PAS nevű program segítségével. A vizuális billentyűket egérrel is lehet kezelni. A

kontrollpontok megjelenítése a szerkesztési környezetben a képsíkhoz viszonyított magasság szerinti fordított sorrendben történik, tehát a megfigyelőtől távolabbi pontok előbb jelennek meg, így a vetítés során adódó takarás esetén **a** közelebbiek kerülnek felül. A szerkesztési környezeten kívüli eredménymegjelenítés a kontrollpontokat kis, tömör alakban jeleníti meg, hogy ne zavarják **a** görbe képét.

```
($N+)

program gb_3d_ip; {sima térgörbeszerkesztés különböző approximációs
                   illetve interpolációs eljárással)

uses
 graph, grafind, crt, dos, kozos, file_kez, kerdesek, kozos_3d,
 eger kez, gomb kez, kozos_gb;

const
 nm=99;
 valasz: kars = modszer: kars =

type
 pont3 = object {térbeli pont objektum)
   xp, yp, zp: real;
   procedure uj(ux, uy, uz: real);
 end;

 kpont = object(pont3) {kontrollpont objektum)
   xr, yr: integer;
   zr: real;
   ns: integer;
   procedure sorr(un: integer);
   procedure koord;
   procedure megj;
 end;

 mp = array[0..nm] of kpont;

var
 a: mp; ff: file of mp;
 e, f, g: array[l..nm] of real;
 nh, fv, fk: byte;
 st, i: integer;

procedure kpont.koord; {kontrollpont vetülete és mélysége}
begin
 vetit (xp, yp, zp) ;
 zr:=(ap*xp+bp*yp+cp*zp)/nv2;
 xr:=xt; yr:=yt;
end;
```

```
procedure pont3.uj(ux, uy, uz: real);
begin
  xp:=ux; yp:=uy; zp:=uz;
end;

procedure kpont.megj; {kontrollpont megjelenítése)
var sf: string;
begin
  if gb[2]=0 then begin
    setfillstyle(1,0); fillellipse(xr,yr,10,10);
    str(k, sf); if k<=9 then outtextxy(xr-2,yr-3,sf)
                        else outtextxy(xr-7,yr-3,sf);
  end else begin setfillstyle(1,15); fillellipse(xr,yr,4,4); end;
end;

procedure kpont.sorr(un: integer);
begin ns:=un; end;

procedure sorrend; (mélység szerinti sorrendbehelyezés)
mar sf, fg: byte;
begin
  for k:=1 to nh do a[k].sorr(k);
  repeat
    sf:=0;
    for k:=1 to nh do
      for 1:=1 to nh do
        if (a[1].zr-a[k].zr)*(a[1].ns-a[k].ns)<0 then begin
          fg:=a[k].ns; a[k].sorr(a[1].ns); a[1].sorr(fg); sf:=1; end;
  until sf=0;
end;

procedure kilep;
begin
  qm:=' ';
  kerdez(200,200,'  Menteni a konfigurációt ?','gen','em','I','N');
  repeat kattint(200,200,'I','N',qm);
  until (qm='I') or (qm='N');
  if qm='I' then begin
    eger_n; ment('IP3'); {kontrollpontok tárolása)
    if v='I' then begin
      a[0].sorr(nh);
      assign (ff, kn) ; rewrite (ff) ; write (ff, a) ; close (ff) ;
      a[0].sorr(0); end;
    eger_l;
  end;
end;

procedure konfig;
begin
  repeat
   of:=' ';
   kerdez(200,200,'     Konfiguráció ?','j','árolt','U','T');
```

```
    repeat kattint(200,200,'U','T',qf);
    until (qf='U') or (qf='T');
    if qf='T' then begin
      eger_n; olvas('IP3'); {tárolt kontrollpontok betöltése}
      if ko=1 then begin
        assign(ff,kn); reset(ff); read(ff,a); close(ff); end;
      eger_1;
    end else ko:=1;
  until ko=1;
end;

procedure kpontok; {kontrollpontok megjelenítése}
begin
  setcolor(15); for k:=1 to nh do a[k].koord;
  if gb[2]=0 then begin
    sorrend;
    for 1:=1 to nh do
       for k:=1 to nh do if 1=a[k].ns then a[k].megj; {sorrend szerint}
  end else for k:=1 to nh do a[k].megj; {sorrendtől függetlenül}
end;

procedure spline; {splinefüggvények együtthatóinak kiszámítása}
var
   h, q, r: array [ 1. . nm] of real;
begin
   case qa of
   'N':begin {nyílt görbe}
      q[2]:=1/4; for i:=3 to nh-1 do q[i]:=1/(4-q[i-i]);
      for i:=2 to nh-1 do h[i]:=a[i-1].xp-2*a[i].xp+a[i+i].xp;
      e [1] :=0; e [nh] :=0; h[2] :=h[2] /4;
      for i:=3 to nh-1 do h[i]:=q[i]*(h[i]-h[i-1]); e[nh-i]:=h[nh-
      1]; for i:=nh-2 downto 2 do e[i]:=h[i]-q[i]*e[i+l]; for i:=2
      to nh-1 do h[i]:=a[i-1].yp-2*a[i].yp+a[i+l].yp;
      f [1] :=0; f [nh] :=0; h[2] :=h[2] /4;
      for i:=3 to nh-1 do h[i]:=q[i]*(h[i]-h[i-1]); f[nh-i]:=h[nh-1];
      for i:=nh-2 downto 2 do f[i]:=h[i]-q[i]*f[i+1];
      for i:=2 to nh-1 do h[i]:=a[i-i].zp-2*a[i].zp+a[i+i].zp;
      g[1]:=0; g[nh]:=0; h[2]:=h[2]/4;
      for i:=3 to nh-1 do h[i]:=q[i]*(h[i]-h[i-1]); g[nh-i]:=h[nh-1];
      for i:=nh-2 downto 2 do g[i]:=h[i]-q[i]*g[i+i];
    end;
   'Z':begin {zárt görbe}
      q[1]:=1/4; for i:=2 to nh-1 do q[i]:=i/(4-q[i-1]);
      r[1]:=q[1]; for i:=2 to nh-1 do r[i]:=-r[i-1]*q[i];
      r[nh-1]:=q[nh-1]+r[nh-1]; r[nh]:=4-r[nh-i];
      for i:=1 to nh-2 do h[i]:=a[i].xp-2*a[i+1].xp+a[i+2].xp;
      h[nh-1]:=a[nh-1].xp-2*a[nh].xp+a[1].xp;
      h[nh]:=a[nh].xp-2*a[i].xp+a[2].xp;
      h[i]:=h[1]*q[1];
      for i:=2 to nh-1 do h[i]:=(h[i]-h[i-i])*q[i];
      h[nh]:=h[nh]-h[nh-1];
      r[1]:=1-r[nh]/q[1]; h[1]:=(h[1]-h[nh])/q[1];
```

```
      for i:=2 to nh-2 do begin
         r[i]:=(r[i]-r[i-1])/q[i]; h[i]:=(h[i)-h[i-1])/q[i]; end;
      e[1]:=(h[nh-1]-h[nh-2])/(r[nh-1]-r[nh-2]);
      for i:=nh downto 3 do e[i]:=h[i-2]-r[i-2]*e[1];
      e[2]:=h[nh]-r[nh]*e[1];
      r[1]:=q[i]; for i:=2 to nh-1 do r[i]:=-r[i-i]*q[i];
      r[nh-1]:=q[nh-1]+r[nh-1]; r[nh]:=4-r[nh-1];
      for i:=1 to nh-2 do h[i]:=a[i].yp-2*a[i+1].yp+a[i+2].yp;
      h[nh-1]:=a[nh-1].yp-2*a[nh].yp+a[i].yp;
      h[nh]:=a[nh].yp-2*a[1].yp+a[2].yp;
      h[1]:=h[1]*q[1];
      for i:=2 to nh-1 do h[i]:=(h[i]-h[i-i])*q[i];
      h[nh]:=h[nh]-h[nh-1];
      r[1]:=1-r[nh)/q[1]; h[1]:=(h[1]-h[nh])/q[1];
      for i:=2 to nh-2 do begin
  r[i] :=(r[i]-r[i-1] ) /q[i]; h[i] :=(h[i]-h[i-1] ) /q[i]; end;
      f[1]:=(h[nh-1]-h[nh-2))/(r[nh-1]-r[nh-2]);
      for i:=nh downto 3 do f[i]:=h[i-2]-r[i-2]*f[1];
      f [2] :=h[nh]-r[nh] *f [1] ;
      r[1]:=q[1); for i:=2 to nh-1 do r[i]:=-r[i-1]*q[i];
      r[nh-1]:=q[nh-1]+r[nh-1]; r[nh]:=4-r[nh-1];
      for i:=1 to nh-2 do h[i]:=a[i].zp-2*a[i+l].zp+a[i+2].zp;
      h[nh-1]:=a[nh-1].zp-2*a[nh].zp+a[1].zp;
      h[nh]:=a[nh].zp-2*a[1].zp+a[2].zp;
      h[1]:=h[1]*q[1];
      for i:=2 to nh-1 do h[i]:=(h[i]-h[i-1])*q[i];
      h[nh]:=h[nh]-h[nh-1];
      r[1]:=1-r[nh]/g[1]; h[1]:=(h[1]-h[nh])/q[1];
      for i:=2 to nh-2 do begin
         r[i]:=(r[i]-r[i-1])/q[i]; h[i]:=(h[i]-h[i-1])/q[i]; end;
      g[1]:=(h[nh-1]-h[nh-2])/(r[nh-1)-r[nh-2]);
      for i:=nh downto 3 do g[i]:=h[i-2]-r[i-2]*g[1];
      g[2]:=h[nh]-r[nh]*g[1];
    end;
  end;
end ;

procedure rajz; {görbe megjelenítése}
const nr=50; nb=100;
:ar
      integer;
    u2, u3, ux, uy, uz, w: real;
function xx(i, j: integer): real; {térgörbepont x koordinátája}
begin
  case elj of
  '1':case qa of {Coons-Hermite interpoláció}
    'N':begin {nyílt görbe}
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=1 then
          xx:=(2*u3-3*u2+1)*a[1].xp+(-2*u3+3*u2)*a[2].xp+
   ( (u3-2*u2+u) * (a [2] .xp-a [1] .xp) +(u3-u2) * (a [3] . xp-
             a[1).xp))*k0
        else if i=nh-1 then
```

```
          xx:=(2*u3-3*u2+1)*a[nh-1].xp+(-2*u3+3*u2)*a[nh].xp+ ((u3-
               2*u2+u)*(a[nh].xp-a[nh-2].xp)+(u3-u2)*(a[nh].xp-
               a[nh-1].xp))*k0
        else
          xx:=(2*u3-3*u2+1)*a[i].xp+(-2*u3+3*u2)*a[i+1].xp+ ((u3-
               2*u2+u)*(a[i+1].xp-a[i-1].xp)+(u3-u2)*(a[i+2].xp-
               a[i].xp))*k0;
      end;
    'Z':begin {zárt görbe}
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=1 then
          xx:=(2*u3-3*u2+1)*a[1].xp+(-2*u3+3*u2)*a[2].xp+ ((u3-
               2*u2+u)*(a[2].xp-a[nh].xp)+(u3-u2)*(a[3].xp-
               a [1] .xp) ) *k0
        else if i=nh-1 then
          xx:=(2*u3-3*u2+1)*a[nh-1].xp+(-2*u3+3*u2)*a[nh].xp+ ((u3-
               2*u2+u)*(a[nh].xp-a[nh-2].xp)+(u3-u2)*(a[1].xp-
               a[nh-1].xp))*k0
        else if i=nh then
          xx:=(2*u3-3*u2+1)*a[nh].xp+(-2*u3+3*u2)*a[1].xp+ ((u3-
               2*u2+u)*(a[1].xp-a[nh-1].xp)+(u3-u2)*(a[2].xp-
               a[nh].xp))*k0
        else
          xx:=(2*u3-3*u2+1)*a[i].xp+(-2*u3+3*u2)*a[i+1].xp+ ((u3-
               2*u2+u)*(a[i+1].xp-a[i-1].xp)+(u3-u2)*(a[i+2].xp-
               a[i] .xp) ) *k0;
      end;
    end;
  '3':begin {spline interpoláció}
        u:=j/nr; w:=1-u;
        case qa of
        'N':begin {nyílt görbe}
            xx:=e[i]*w*w*w+e[i+1]*u*u*u+(a[i].xp-e[i])*w+(a[i+1].xp-
                 e[i+1])*u;
          end;
        'Z':if i=nh then begin {zárt görbe}
            xx:=e[nh]*w*w*w+e[1]*u*u*u+(a[nh].xp-e[nh])*w+(a[1].xp-
                 e[1])*u;
          end else begin
            xx:=e[i]*w*w*w+e[i+1]*u*u*u+(a[i].xp-e[i])*w+(a[i+1].xp-
                 e[i+1])*u;
          end;
        end;
      end;
  '4':case qa of {másodfokú B-spline közelítés}
    'N':begin {nyílt görbe}
        U: =j /nr;
        xx:=sqr(1-u)*a[1].xp+2*u*(1-u)*a[2].xp+sqr(u)*a[3].xp;
        if i=2 then
          xx:=sqr(1-u)*a[1].xp+2*u*(1-u)*a[2].xp+
               sqr(u)*(a[2].xp+a[3].xp)/2
        else if i=nh-1 then
          xx:=sqr(1-u)*(a[nh-2].xp+a[nh-1].xp)/2+
```

```
              2*u*(1-u)*a[nh-1].xp+sqr(u)*a[nh].xp
        else
          xx:=sqr(1-u)*(a[i-1].xp+a[i].xp)/2+2*u*(1-u)*a[i].xp+
              sqr(u)*(a[i].xp+a[i+1].xp)/2;
      end;
    'Z':begin {zárt görbe}
        U: =j /nr;
        if i=0 then
          xx:=sqr(1-u)*(a[nh-1].xp+a[nh].xp)/2+2*u*(1-u)*a[nh].xp+
              sgr(u)*(a[nh].xp+a[1].xp)/2
        else if i=1 then
          xx:=sqr(1-u)*(a[nh].xp+a[1].xp)/2+2*u*(1-u)*a[1].xp+
              sgr(u)*(a[1].xp+a[2].xp)/2
        else
          xx:=sqr(1-u)*(a[i-1].xp+a[i].xp)/2+2*u*(1-u)*a[i].xp+
              sgr(u}*(a[i].xp+a[i+1].xp)/2;
      end;
    end;
  '5':case qa of (harmadfokú B-spline közelítés)
    'N':begin {nyílt görbe}
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=0 then
          xx:=((6-u3)*a[1].xp+u3*a[2].xp)/6
        else if i=1 then
          xx:=((5-3*u-3*u2+2*u3)*a[1].xp+(1+3*u+3*u2-3*u3)*a[2].xp+
              u3*a[3].xp)/6
        else if i=nh-1 then
          xx:=((1-3*u+3*u2-u3)*a[nh-2].xp+(4-6*u2+3*u3)*a[nh-1].xp+
               (1+3*u+3*u2-2*u3)*a[nh].xp)/6
        else if i=nh then
          xx:=((1-3*u+3*u2-u3)*a[nh-1].xp+(5+3*u-3*u2+u3)*a[nh].xp)/6
        else
          xx:=((1-3*u+3*u2-u3)*a[i-1].xp+(4-6*u2+3*u3)*a[i].xp+
              (1+3*u+3*u2-3*u3)*a[i+1].xp+u3*a[i+2].xp)/6;
      end;
    'Z':begin (zárt görbe)
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=0 then
          xx:=((1-3*u+3*u2-u3)*a[nh-1].xp+(4-6*u2+3*u3)*a[nh].xp+
                (1+3*u+3*u2-3*u3)*a[1].xp+u3*a[2].xp)/6
        else if i=1 then
          xx:=((1-3*u+3*u2-u3)*a[nh].xp+(4-6*u2+3*u3)*a[1].xp+
                (1+3*u+3*u2-3*u3)*a[2].xp+u3*a[3].xp)/6
        else if i=nh-1 then
          xx:=((1-3*u+3*u2-u3)*a[nh-2].xp+(4-6*u2+3*u3)*a[nh-1].xp+
               (1+3*u+3*u2-3*u3)*a[nh].xp+u3*a[1].xp)/6
        else
          xx:=((1-3*u+3*u2-u3)*a[i-1].xp+(4-6*u2+3*u3)*a[i].xp+
              (1+3*u+3*u2-3*u3)*a[i+1].xp+u3*a[i+2].xp)/6;
      end;
    end;
end;
```

```
function yy(i, j: integer): real; (térgörbepont y koordinátája}
begin
  case elj of
  '1':case qa of (Coons-Hermite interpoláció)
    'N':begin {nyílt görbe)
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=1 then
          yy:=(2*u3-3*u2+1)*a[1].yp+(-2*u3+3*u2)*a[2].yp+
              ((u3-2*u2+u)*(a[2].yp-a[1].yp)+(u3-u2)*(a[3].yp-
              a[1].yp))*k0
        else if i=nh-1 then
          yy:=(2*u3-3*u2+1)*a[nh-1].yp+(-2*u3+3*u2)*a[nh].yp+ ((u3-
              2*u2+u)*(a[nh].yp-a[nh-2].yp)+(u3-u2)*(a[nh].yp-
              a[nh-1].yp))*k0
        else
          yy:=(2*u3-3*u2+1)*a[i].yp+(-2*u3+3*u2)*a[i+l].yp+ ((u3-
              2*u2+u)*(a[i+l].yp-a[i-1].yp)+(u3-u2)*(a[i+2].yp-
              a[i] .yp) ) *k0;
      end;
    'Z':begin (zárt görbe}
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=1 then
          yy:=(2*u3-3*u2+1)*a[1].yp+(-2*u3+3*u2)*a[2].yp+ ((u3-
              2*u2+u)*(a[2].yp-a[nh].yp)+(u3-u2)*(a[3].yp-
              a[1].yp))*k0
        else if i=nh-1 then
          yy:=(2*u3-3*u2+1)*a[nh-1].yp+(-2*u3+3*u2)*a[nh].yp+
              ((u3-2*u2+u)*(a[nh].yp-a[nh-2].yp)+(u3-u2)*(a[1].yp-
              a[nh-1].yp))*k0
        else if i=nh then
          yy:=(2*u3-3*u2+1)*a[nh].yp+(-2*u3+3*u2)*a[1].yp+ ((u3-
              2*u2+u)*(a[1].yp-a[nh-1].yp)+(u3-u2)*(a[2].yp-
              a[nh].yp))*k0
        else
          yy:=(2*u3-3*u2+1)*a[i].yp+(-2*u3+3*u2)*a[i+l].yp+ ((u3-
              2*u2+u)*(a[i+l].yp-a[i-1].yp)+(u3-u2)*(a[i+2].yp-
              a[i] .yp) )*k0;
      end;
    end;
  '3':begin (spline interpoláció}
        u:=j/nr; w:=1-u;
        case qa of
        'N':begin {nyílt görbe}
            yy:=f[i]*w*w*w+f[i+l]*u*u*u+(a[i].yp-f[i])*w+(a[i+l].yp-
                f[i+l])*u;
          end;
        'Z':if i=nh then begin {zárt görbe}
             yy:=f[nh]*w*w*w+f[1]*u*u*u+(a[nh].yp-f[nh])*w+(a[1].yp-
                f[1])*u;
          end else begin
            yy:=f[i]*w*w*w+f[i+1]*u*u*u+(a[i].yp-f[i])*w+(a[i+1].yp-
                f [ i+l ]) *u;
          end;
```

```
      end;
    end;
'4':case qa of {másodfokú B-spline közelítés)
 'N':begin {nyílt görbe}
      u:=j/nr;
      yy:=sqr(1-u)*a[1].yp+2*u*(1-u)*a[2].yp+sqr(u)*a[3].yp;
      if i=2 then
        yy:=sqr(1-u)*a[1].yp+2*u*(1-u)*a[2].yp+
            sqr(u)*(a[2].yp+a[3].yp)/2
      else if i=nh-1.then
        yy:=sqr(1-u)*(a[nh-2).yp+a[nh-1].yp)/2+
            2*u*(1-u)*a[nh-1].yp+sqr(u)*a[nh].yp
      else
        yy:=sqr(1¯u)*(a[i¯1].yp+a[i].yp)/2+2*u*(1¯u)*a[i].yp+
            sqr(u) *(a[i] .yp+a[i+1] .yp) /2;
    end;
 'Z':begin {zárt görbe)
      U: =j /nr;
      if i=0 then
        yy:=sqr(1-u)*(a[nh-1].yp+a[nh].yp)/2+2*u*(1-u)*a[nh].yp+
            sqr(u) * (a[nh) .yp+a[1] .yp) /2
      else if i=1 then
        yy:=sqr(1-u)*(a[nh].yp+a[1].yp)/2+2*u*(1-u)*a[1].yp+
            sqr(u)*(a[1] .yp+a[2].yp)/2
      else
        yy:=sqr(1-u)*(a[i-1).yp+a[i].yp)/2+2*u*(1-u)*a[i].yp+
            sgr(u)*(a[i] .yp+a[i+1] .yp)/2;
    end;
 end;
'5':case qa of (harmadfokú B-spline közelítés)
 'N':begin (nyílt görbe)
      u:=j/nr; u2:=u*u; u3:=u2*u;
      if i=0 then
        yy:=((6¯u3)*a[1].yp+u3*a[2].yp)/6
      else if i=1 then
        yy:=((5-3*u-3*u2+2*u3)*a[1].yp+(1+3*u+3*u2-3*u3)*a[2].yp+
            u3*a[3].yp)/6
      else if i=nh-1 then
        yy:=((1-3*u+3*u2-u3)*a[nh-2].yp+(4-6*u2+3*u3)*a[nh-1].yp+
            (1+3*u+3*u2-2*u3)*a[nh].yp)/6
      else if i=nh then
        yy:=((1-3*u+3*u2-u3)*a[nh-1].yp+(5+3*u-3*u2+u3)*a[nh].yp)/6
      else
        yy:=((1-3*u+3*u2-u3)*a[i-1].yp+(4-6*u2+3*u3)*a[i].yp+
            (1+3*u+3*u2-3*u3)*a[i+l].yp+u3*a[i+2).yp)/6;
    end;
 'Z':begin (zárt görbe)
      u:=j/nr; u2:=u*u; u3:=u2*u;
      if i=0 then
        yy:=((1-3*u+3*u2-u3)*a[nh-1].yp+(4-6*u2+3*u3)*a[nh].yp+
              (1+3*u+3*u2-3*u3)*a[1].yp+u3*a[2].yp)/6
      else if i=1 then
        yy:=((1-3*u+3*u2-u3)*a[nh].yp+(4-6*u2+3*u3)*a[1].yp+
```

```
                (1+3*u+3*u2-3*u3)*a[2].yp+u3*a[3].yp)/6
          else if i=nh-1 then
              yy:=((1-3*u+3*u2-u3)*a[nh-2].yp+(4-6*u2+3*u3)*a[nh-1].yp+
                (1+3*u+3*u2-3*u3)*a[nh].yp+u3*a[1].yp)/6
          else
            yy:-((1-3*u+3*u2-u3)*a[i-1].yp+(4-6*u2+3*u3)*a[i].yp+
                (1+3*u+3*u2-3*u3)*a[i+l].yp+u3*a[i+2].yp)/6;
       end;
     end;
   end;
end;
function zz(i, j: integer): real; {térgörbepont z koordinátája}
begin
  case elj of
   '1':case qa of {Coons-Hermite interpoláció}
     'N':begin {nyílt görbe)
         u:=j/nr; u2:=u*u; u3:=u2*u;
         if i=1 then
           zz:=(2*u3-3*u2+1)*a[1].zp+(-2*u3+3*u2)*a[2].zp+
                ((u3-2*u2+u)*(a[2].zp-a[1].zp)+(u3-u2)*(a[3].zp-
                a[1].zp))*k0
         else if i=nh-1 then
           zz:=(2*u3-3*u2+1)*a[nh-1].zp+(-2*u3+3*u2)*a[nh].zp+
                ((u3-2*u2+u)*(a[nh].zp-a[nh-21.zp)+(u3-u2)*(a[nh].zp-
                a[nh-l].zp))*k0
         else
           zz:=(2*u3-3*u2+1)*a[i].zp+(-2*u3+3*u2)*a[i+l].zp+
                ((u3-2*u2+u)*(a[i+1).zp-a[i-l].zp)+(u3-
                u2)*(a[i+2].zp-
                a[i].zp))*k0;
       end;
     'Z':begin {zárt görbe}
         u:=j/nr; u2:=u*u; u3:=u2*u;
         if i=1 then
           zz:=(2*u3-3*u2+1)*a[1].zp+(-2*u3+3*u2)*a[2].zp+
                ((u3-2*u2+u)*(a[2].zp-a[nh].zp)+(u3-u2)*(a[3].zp-
                a[1].zp))*k0
         else if i=nh-1 then
           zz:=(2*u3-3*u2+1)*a[nh-1].zp+(-2*u3+3*u2)*a[nh].zp+
                ((u3-2*u2+u)*(a[nh].zp-a[nh-2].zp)+(u3-u2)*(a[1].zp-
                a[nh-l].zp))*k0
         else if i=nh then
           zz:=(2*u3-3*u2+1)*a[nh].zp+(-2*u3+3*u2)*a[1].zp+
                ((u3-2*u2+u)*(a[1].zp-a[nh-1].zp)+(u3-u2)*(a[2].zp-
                a[nh].zp))*k0
         else
           zz:=(2*u3-3*u2+1)*a[i].zp+(-2*u3+3*u2)*a[i+l].zp+
                ((u3-2*u2+u)*(a[i+1].zp-a[i-1].zp)+(u3-
                u2)*(a[i+2].zp-
                a[i].zp))*k0;
       end;
     end;
   '3':begin {spline interpoláció)
         u : =j /nr; w:=1-u;
```

```
      case qa of
      'N':begin {nyílt görbe}
          zz:=g[i]*w*w*w+g[i+1]*u*u*u+(a[i].zp-g[i])*w+(a[i+1].zp-
              g[i+1])*u;
        end;
      'Z':if i=nh then begin {zárt görbe}
           zz:=g[nh]*w*w*w+g[1]*u*u*u+(a[nh].zp-g[nh])*w+(a[1].zp-
              g[1])*u;
        end else begin
          zz:=g[i]*w*w*w+g[i+1]*u*u*u+(a[i].zp-g[i])*w+(a[i+1].zp-
              g[i+1])*u;
        end;
      end;
    end;
'4':case qa of {másodfokú B-spline közelítés}
 'N':begin {nyílt görbe)
      u: =j /nr;
      zz:=sqr(1-u)*a[1].zp+2*u*(1-u)*a[2].zp+sgr(u)*a[3].zp;
      if i=2 then
        zz:=sqr(1-u)*a[1].zp+2*u*(1-u)*a[2].zp+
            sqr(u)*(a[2].zp+a[3].zp)/2
      else if i=nh-1 then
        zz:=sqr(1-u)*(a[nh-2].zp+a[nh-1].zp)/2+
            2*u*(1-u)*a[nh-1].zp+sgr(u)*a[nh].zp
      else
        zz:=sqr(1-u)*(a[i-1].zp+a[i].zp)/2+2*u*(1-u)*a[i].zp+
            sqr(u)*(a[i].zp+a[i+1].zp)/2;
    end;
 'Z':begin (zárt görbe)
      u:=j/nr;
      if i=0 then
        zz:=sqr(1-u)*(a[nh-1].zp+a[nh].zp)/2+2*u*(1-u)*a[nh].zp+
            sqr(u)*(a[nh].zp+a[1].zp)/2
      else if i=1 then
        zz:=sqr(1-u)*(a[nh].zp+a[1].zp)/2+2*u*(1-u)*a[1].zp+
            sqr(u)*(a[1].zp+a[2].zp)/2
      else
        zz:=sqr(1-u)*(a[i-1].zp+a[i].zp)/2+2*u*(1-u)*a[i].zp+
            sqr(u)*(a[i].zp+a[i+1].zp)/2;
    end;
 end;
'5':case qa of (harmadfokú B-spline közelítés)
 'N':begin (nyílt görbe)
      u:=j/nr; u2:=u*u; u3:=u2*u;
      if i=0 then
        zz:=((6-u3)*a[1].zp+u3*a[2].zp)/6
      else if i=1 then
        zz:=((5-3*u-3*u2+2*u3)*a[1].zp+(1+3*u+3*u2-3*u3)*a[2].zp+
            u3*a[3].zp)/6
      else if i=nh-1 then
        zz:=((1-3*u+3*u2-u3)*a[nh-2].zp+(4-6*u2+3*u3)*a[nh-1].zp+
            (1+3*u+3*u2-2*u3)*a[nh].zp)/6
      else if i=nh then
```

```
          zz:=((1-3*u+3*u2-u3)*a[nh-1].zp+(5+3*u-
        3*u2+u3)*a[nh].zp)/6 else
          zz:=((1-3*u+3*u2-u3)*a[i-1].zp+(4-6*u2+3*u3)*a[i].zp+
               (1+3*u+3*u2-3*u3)*a[i+1].zp+u3*a[i+2].zp)/6;
      end;
    'Z':begin {zárt görbe}
        u:=j/nr; u2:=u*u; u3:=u2*u;
        if i=0 then
          zz:=((1-3*u+3*u2-u3)*a[nh-1].zp+(4-6*u2+3*u3)*a[nh].zp+
           (1+3*u+3*u2-3*u3)*a[1].zp+u3*a[2].zp)/6 else if i=1
           then
          zz:=((1-3*u+3*u2-u3)*a[nh].zp+(4-6*u2+3*u3)*a[1].zp+
           (1+3*u+3*u2-3*u3)*a[2].zp+u3*a[3].zp)/6 else if
           i=nh-1 then
          zz:=((i-3*u+3*u2-u3)*a[nh-2].zp+(4-6*u2+3*u3)*a[nh-
           1].zp+ (1+3*u+3*u2-3*u3)*a[nh].zp+u3*a[1].zp)/6
           else
          zz:=((1-3*u+3*u2-u3)*a[i-i].zp+(4-6*u2+3*u3)*a[i].zp+
               (1+3*u+3*u2-3*u3)*a[i+1].zp+u3*a[i+2].zp)/6;
      end;
    end;
  end;
end;

begin
  eger n;
  racs; if gb[2]=1 then nyil; setlinestyle(0,0,3); setcolor(15);
  if qa='N' then begin vetit(a[1].xp,a[1].yp,a[1].zp); moveto(xt,yt);
end;
  case elj of
  '1':begin (Coons-Hermite interpoláció)
      if qa='Z' then begin
        vetit(xx(1,0),yy(1,0),zz(1,0)); moveto(xt,yt); end;
      for is=1 to nh-1 do for j:=1 to nr do begin
        vetit(xx(i,j),yy(i,j),zz(i,j)); lineto(xt,yt); end;
      if qa='Z' then for j:=1 to nr do begin
        vetit(xx(nh,j),yy(nh,j),zz(nh,j)); lineto(xt,yt); end;
    end;
  '2':if qa='N' then begin {Bézier-közelítés)
        for j:=0 to nb-1 do begin
          u:=j/nb; u2:=1-u; u3:=u2; for 1:=2 to nh-i do u3:=u3*u2;
          ux:=u3*a[1].xp; uy:=u3*a[1].yp; uz:=u3*a[1].zp;
          for i:=2 to nh do begin
           u3:=u3*(nh-i+1)/(i-1)*u/(1-u);
            ux:=ux+u3*a[i].xp; uy:=uy+u3*a[i].yp; uz:=uz+u3*a[i].zp;
          end;
          vetit (ux, uy, uz) ; lineto(xt,yt);
        end;
        vetit(a[nh].xp,a[nh].yp,a[nh].zp); lineto(xt,yt);
      end;
  '3':case qa of (spline interpoláció)
      'N':for is=1 to nh-1 do
```

```
            for j:=1 to nr do begin
              vetit(xx(i,j),yy(i,j),zz(i,j)); lineto(xt,yt); end;
      'Z':begin
          vetit(xx(1,0),yy(1,0),zz(1,0)); moveto(xt,yt);
          for is=1 to nh do
            for j:=1 to nr do begin
              vetit(xx(i,j),yy(i,j),zz(i,j)); lineto(xt,yt); end;
        end;
      end;
  '4':begin (másodfokú B-spline közelítés}
        if qa='Z' then begin
          vetit(xx(0,0),yy(0,0),zz(0,0)); moveto(xt,yt);
      for i:=0 to 1 do
        for j:=1 to nr do begin
          vetit(xx(i,j),yy(i,j),zz(i,j)); lineto(xt,yt); end;
      end;
      for i:=2 to nh-1 do
        for j:=1 to nr do begin
        vetit(xx(i,j),yy(i,j),zz(i,j)); lineto(xt,yt); end;
    end;
  '5':begin (harmadfokú B-spline közelítés)
        if qa='Z' then begin
          vetit(xx(0,0),yy(0,0),zz(0,0) ); moveto(xt,yt); end;
      for i:=0 to nh-1 do
          for j:=1 to nr do begin
          vetit(xx(i,j),yy(i,j),zz(i,j)); lineto(xt,yt); end;
    if qa='N' then for j:=1 to nr do begin
      vetit(xx(nh,j),yy(nh,j),zz(nh,j)); lineto(xt,yt); end;
    end
  end;
  setlinestyle(0,0,1);
  eger l;
end;

procedure inform;
        (kiválasztott kontrollpont térbeli helyének kiírása}
var ks, kx, ky, kz: string;
begin
  setcolor(0); str(sz,ks);
  str(a[sz].xp:3:0,kx); str(a[sz].yp:3:0,ky); str(a[sz].zp:3:0,kz);
  outtextxy(130,470,'Sorszám: '+kn+'    Hely:   x = '+
            kx+'   y = '+ky+'    z = '+kz);
end;

procedure fg; {vetítési irány változtatásának vezérlése)
begin
  repeat
    repeat kr:=readkey;
    until (kr=fel) or (kr=1e) or (kr=jobb) or (kr=bal) or
          (ord(kr)=27);
    cleardevice;
    if kr<>chr(27) then begin forgat; rajz;
      if qm='I' then kpontok; end;
```

```
  until kr=chr(27);
end;

begin
  grindhi; keret(kx0, ky0); eger_k;
  repeat
    qu:='N'; fe:=0; fk:=0;
    ap:=0.7; bp:=0.2; cp:=0.3; {kezdeti vetítési irány}
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nm do begin a[k].uj(0,0,0); a[k].sorr(0); end;
    qk:='N';
    eger_n; racs; konfig; nh:=a[0].ns; parancs(1); eljaras;
    eger_n; racs; kpontok; eger_l;
    repeat
      eger_t(kx0,10,620,ky0); eger_l; qr:=' '; k:=0;
      repeat par_katt(l,qr); until qr in valasz; eger_n;
      case qr of
        'B':if nh<nm then begin {új kontrollpont beszúrása)
            gb[6]:=1; parancs(1);
            sorszam;
            if sz>0 then begin
              koordin3('Hely');
              if qr<>chr(27) then begin
                if sz>nh then sz:=nh+l;
                 if sz<=nh then for k:=nh downto sz do a[k+l]:=a[k];
                a[sz].uj(x,y,z);
                nh:=nh+1;
              end;
            end;
            gb[6]:=0; lap(208,5); fk:=0;
          end;
        'V':begin (váltortatandó kontrollpont kiválasztása)
            gb[1]:=1; parancs(1); sorszam;
            if (sz>0) and (sz<=nh) then begin
              inform; vlt;
              case qr of
              'S':begin (sorszám váltortatása}
                    st:=sz; a[0]:=a[sz]; sorszam;
                    if (sz>st) and (sz<=nh) then
                      for k:=st to sz-1 do a[k]:=a[k+l];
                    if (sz<st) and (sz>0) then
                      for k:=st downto sz+l do a[k]:=a[k-1];
                    a[sz]:=a[0]
                  end;
              'H':begin {hely váltortatása}
                    koordin3('Hely');
                    if qr<>chr (27) then a [sz] . uj (x, y, z) ; end;
              end;
            end;
            setfillstyle(1,11); bar(130,470,590,480);
            gb[1]:=0; lap(208,5); fk:=0;
          end;
```

```
      'T':if nh>0 then begin {kontrollpont törlése)
            gb[5]:=1; parancs(1); sorszam;
            if sz>0 then begin
              if sz<nh then for k:=sz+1 to nh do a[k-1]:=a[k];
              if sz<=nh then nh:=nh-l;
            end;
            gb[5]:=0; lap(208,5); fk:=0;
          end;
            nh>2 then begin {görbemegjelenítés}
            gb[3]:=1; parancs(1); alak;
            eger_t(5,5,200,50); eger_1; elj:=' ';
            repeat elj katt(elj); {eljárásválasztás}
            until elj in modszer; eger_n;
            for 1:=7 to 10 do gb[1]:=0;
            case elj of
            '1':begin gb[7]:=1; egyutth; lap(208,5);
                  str(k0:3:2,kn); outtextxy(262,23,'k0 = '+kn); end;
            '2':begin gb[8]:=1; end;
            '3':begin spline; gb[9]:=1; end;
            '4':begin b_spline; gó[10]:=1; end;
            end;
            eljaras; rajz;
            gb[3]:=0; fk:=1;
          end;
      'E':if fk=1 then begin (görbemegjelenítés keret nélkül)
            gb[2]:=1; parancs(1); kpl; cleardevice;
            rajz; if qm='I' then kpontok; fg;
            keret(kx0, ky0); eljaras;
            gb[2] :=0; fk:=0;
          end;
      'Q':begin {kilépés}
            gb[4]:=1; parancs(1);
            repeat kilep until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf; gb[4]:=0; fk:=0;
          end;
      end;
      if fk=1 then rajz else racs; kpontok; parancs(1); eger_l;
    until (qk='I') or (qu='I');
  until qu='N';
  closegraph;
```

2.3.7. β -spline közelítés

A harmadfokú B-spline approximáció általánosítható, abban az értelemben, hogy az előállítási függvények első- és második deriváltjainak folytonossága helyett az érintő egységvektor:

$$\mathbf{t}(u) = \frac{\mathbf{r}'(u)}{|\mathbf{r}'(u)|} \quad , \tag{2.59}$$

illetve a

$$\mathbf{k}(u) = \frac{\mathbf{r}''(u) - \mathbf{t}(u)(\mathbf{r}''(u)\cdot\mathbf{t}(u))}{|\mathbf{r}'(u)|^2} = \frac{(\mathbf{r}'(u)\times\mathbf{r}''(u))\times\mathbf{r}'(u)}{|\mathbf{r}'(u)|^4} \tag{2.60}$$

folytonosságát kötjük ki. A k(u) vektort azért jogosult görbületvektornak nevezni, mert abszolút értéke a görbe görbületével egyenlő, iránya pedig megegyezik a görbe főnormálisának az irányával. Ezen a főnormálison van a görbeponttól 1 / I **k(u) I** távolságra a görbületi középpont. Ez azt jelenti, hogy a görbeívek csatlakozási pontjaiban a két ívhez tartozó érintővektorok hossza különbözhet:
illetve a második deriváltak egyenlősége helyett elegendő ha közöttük fennáll az

$$\mathbf{r}_2'(0) = \beta_1 \mathbf{r}_1'(1) \quad , \tag{2.61}$$

összefüggés, ami biztosítja $k_2(0)$ és $k_1(1)$ egyenlőségét. Ezáltal $r_2''(0)$ is benne van a csatlakozási ponthoz tartozó simulósíkban. Két alakparaméter, β_1 és β_2 jelent meg így, amelyeknek változtatásával a görbe alakja módosítható (a simaság rovására). Ha $\beta_1 = 1$ és $\beta_2 = 0$, akkor

$$\mathbf{r}_2''(0) = \beta_1^2 \mathbf{r}_1''(1) + 2\beta_2 \mathbf{r}_1'(1) \tag{2.62}$$

a paraméteres előállítás első és második deriváltjai folytonosak lesznek és a harmadfokú B-spline görbét kapjuk. Az alakparamétereket a görbére gyakorolt hatásuk alapján eltolásnak (β_1 - bias) illetve feszültségnek (β_2 - tension) is nevezhetjük.

A (3 -spline

$$\mathbf{r}(u)=\sum_{k=0}^{3} f_k(u)\mathbf{p}_{i+k-1} \quad ; \quad u\in[0,1], \quad i=0,1,\ldots,n-1 \qquad (2.63$$

előállításában szereplő

$$\begin{aligned}
f_0(u) &= \frac{1}{\delta}\beta_1^3(1-u)^3 \\
f_1(u) &= \frac{1}{\delta}\left[\beta_1^3 u\left(u^2-3u+3\right)+\beta_1^2\left(u^3-3u^2+2\right)+\right. \\
&\qquad \left.+\beta_1\left(u^3-3u+2\right)+\beta_2\left(2u^3-3u^2+1\right)\right] \\
f_2(u) &= \frac{1}{\delta}\left[\beta_1^2 u^2(3-u)+\beta_1 u\left(3-u^2\right)+\beta_2 u^2(3-2u)+1-u^3\right] \\
f_3(u) &= \frac{u^3}{\delta} \\
\delta &= \beta_1^3+2\left(\beta_1^2+\beta_1\right)+\beta_2 \quad ; \quad u\in[0,1]
\end{aligned} \qquad . (2.64)$$

Nyílt görbék szerkesztésénél a karakterisztikus keret végein többszörös kontrollpontokat használhatunk, vagy egy-egy se édkontroll ontot tekintünk, amelyeket úgy határozunk meg, hogy a görbe végpontjai a karakterisztikus keret határpontjaiba kerüljenek. A 2.63. előállítás ennek az esetnek felel meg, ahol p_1 és p_{n+1} a segédkontrollpontokat jelentik és helyzetük a

$$\begin{aligned}
\mathbf{p}_{-1} &= \frac{1}{\beta_1^3}\left(\mathbf{p}_0-\mathbf{p}_1\right)+\mathbf{p}_0 \\
\mathbf{p}_{n+1} &= \beta_1^3\left(\mathbf{p}_n-\mathbf{p}_{n-1}\right)+\mathbf{p}_n
\end{aligned} \qquad (2.65$$

képletekkel számítható. A segédkontrollpontokat az ábrákon nem tüntetjük fel. szerepük a görbe meghosszabbítása a szélső csatlakozási pontoktól az eredeti karakterisztikus keret - $i = 0$ és $i = n$ -1 -nek megfelelő - végpontjáig. Az érintő a végpontokban a karakterisztikus keret megfelelő szélső szakasza.

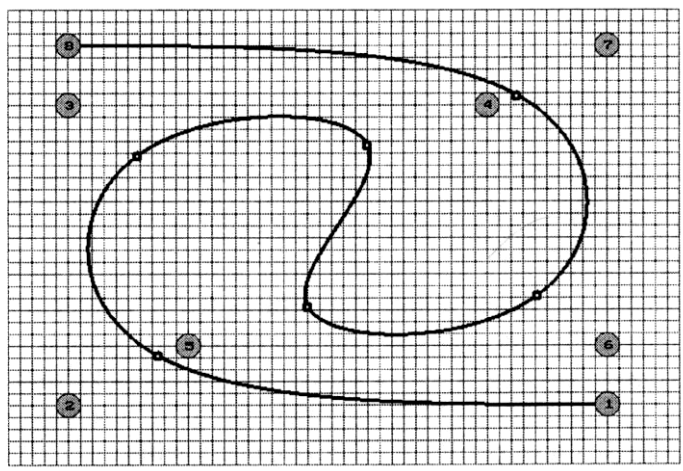
Ez a módszer természetesen a harmadfokú B-spline approximációnál (amely *a* β-spline közelítés sajátos esetének tekinthető) is alkalmazható. Alapvető elő-

nye a gömb topológiájú zárt felületek szerkesztésénél mutatkozik meg, ugyanis lehetővé válik a pólusokban is sima felület előállítása.

Zárt görbék esetén a 2.63 előállításban az *i* index felső határa *n,* az n+1 pontból álló karakterisztikus keretet pedig ciklikusan bezárjuk:

$$\mathbf{p}_{-1} = \mathbf{p}_n \; ; \quad \mathbf{p}_{n+1} = \mathbf{p}_0 \; ; \quad \mathbf{p}_{n+2} = \mathbf{p}_1 \quad . \tag{2.66}$$

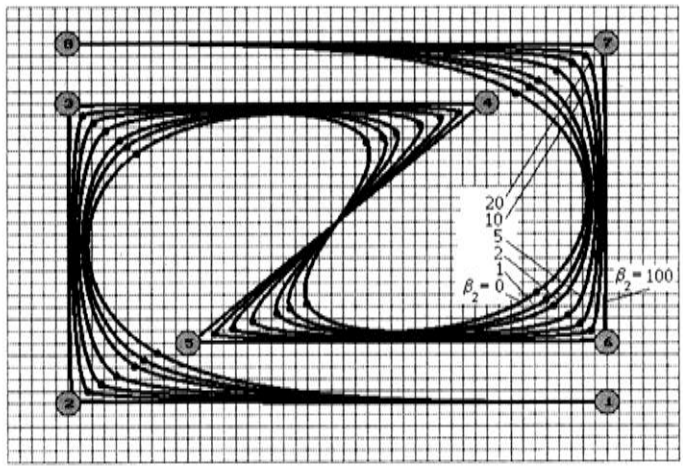
Az alakparamétereknek a görbére gyakorolt hatását a *2.23 - 2.28. ábrák* szemléltetik nyílt síkgörbék, illetve a *2.29 - 2.30. ábrák* zárt síkgörbék esetén.



$\beta_1 = 1 \, , \; \beta_2 = 0$

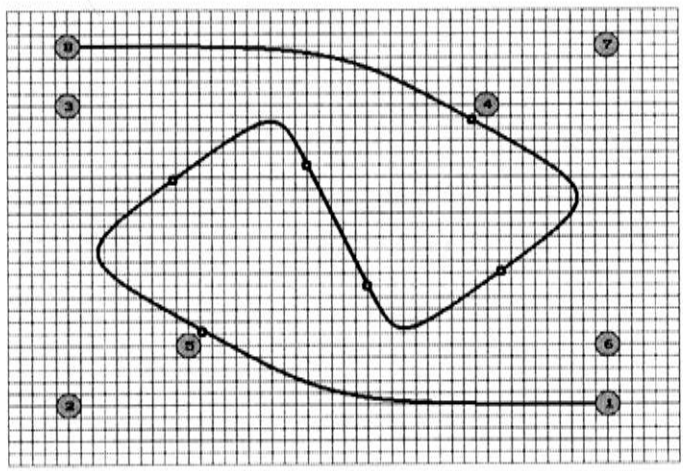
2.23. ábra

A *2.23. ábrán* egy B-spline görbe látható, a *2.24. ábrán* pedig ugyanazzal **a** karakterisztikus kerettel szerkesztett Í -spline görbék, különböző pozitív feszültségértékekre. Megfigyelhető, hogy a feszültség növekedése az apró körökkel jelölt csatlakozási pontoknak a kontrollpontok felé való vonzását eredményezi.



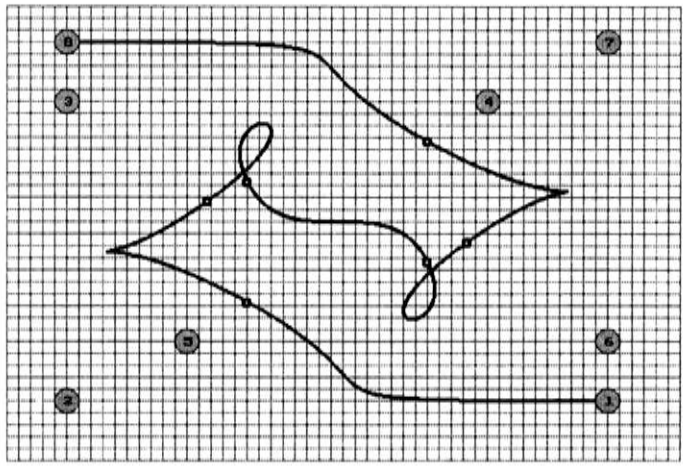
$\beta_1 = 1$

2.24. ábra



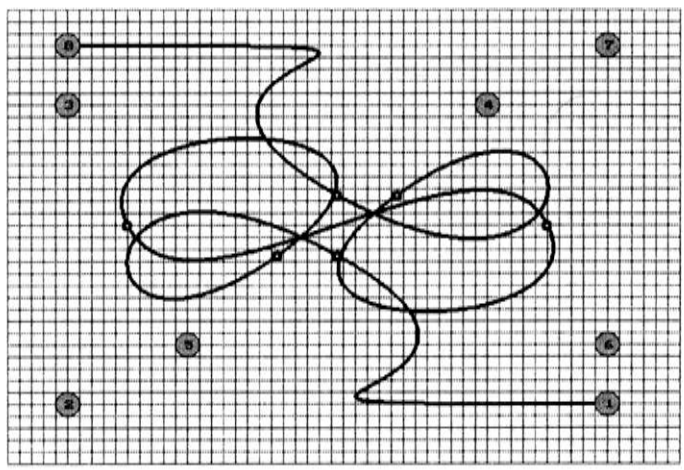
$\beta_1 = 1$, $\beta_2 = -2$

a)



$\beta_1 = 1\,,\ \beta_2 = -3$

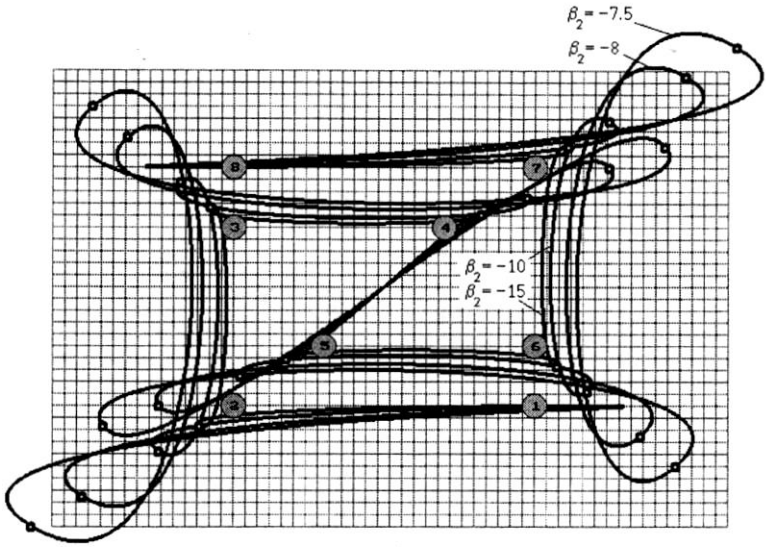
b)



$\beta_1 = 1\,,\ \beta_2 = -4$

c)

2.25. ábra

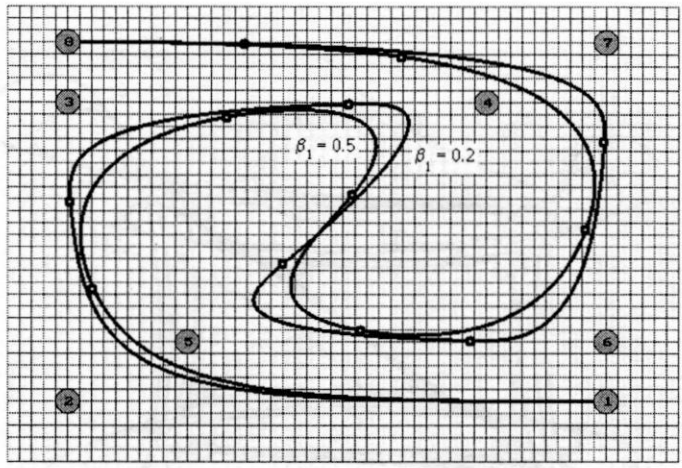


A 2.26. ábra 2.25. ábra negatív feszültség hatását szemlélteti a

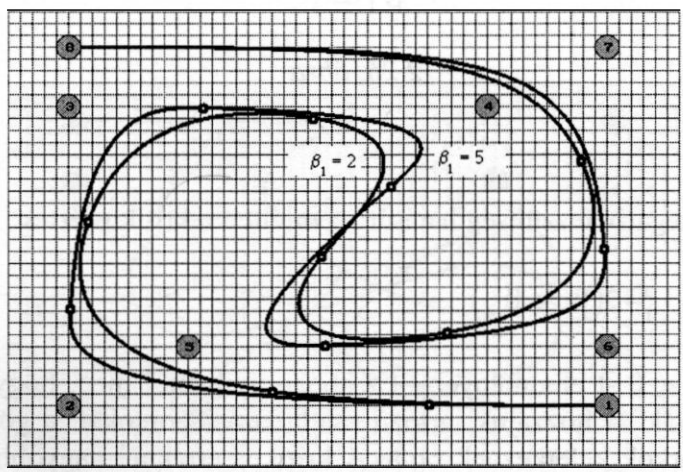
kritikus érték:

$$\beta_2 = -\beta_1^3 - 2(\beta_1^2 + \beta_1) \qquad (2.67)$$

alatt (amely 8 = 0 -t eredményez). Ez abban nyilvánul meg, hogy a kontrollpontok "taszítják" a hozzájuk tartozó csatlakozási pontokat, a görbe "meglazul", újabb hurkok keletkeznek rajta. A kritikus értéknél nagyobb negatív feszültség hatására a kontrollpontok a görbe csatlakozási pontjait az ellenkező irányból - "kívülről" - vonzzák *(2.26. ábra)*.



a)



b)

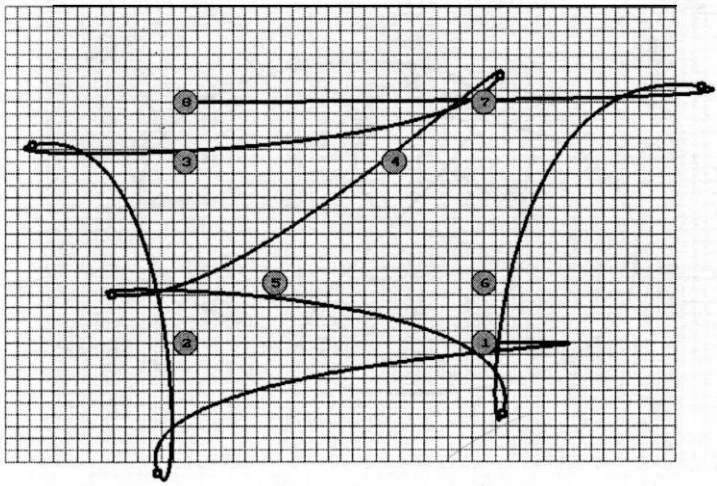
$\beta_2 = 0$

2.27. ábra

Az egységtől különböző eltolásérték a görbe csatlakozási pontjait az alakját viszonylag kis mértékben változtató görbén valamelyik végpont irányába tolja

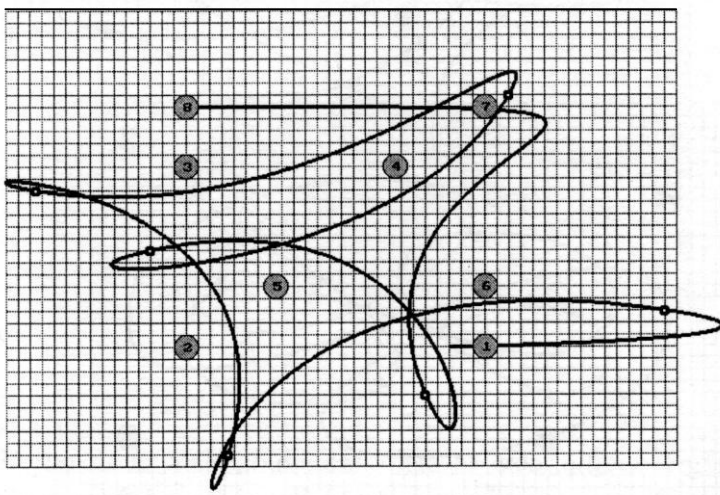
el *(2.27. ábra)*. Az eltolás és feszültség együttes hatását a *2.28. ábra* szemlélteti. Amint a 2.67 képlet is mutatja, a kritikus feszültségérték az eltolás függvénye.

2.28. ábra



a)

$\beta_1 = 0.5\,,\ \beta_2 = -4$

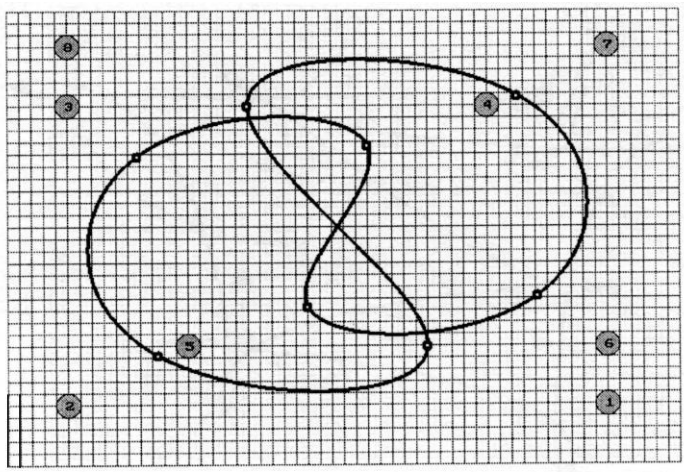


b)

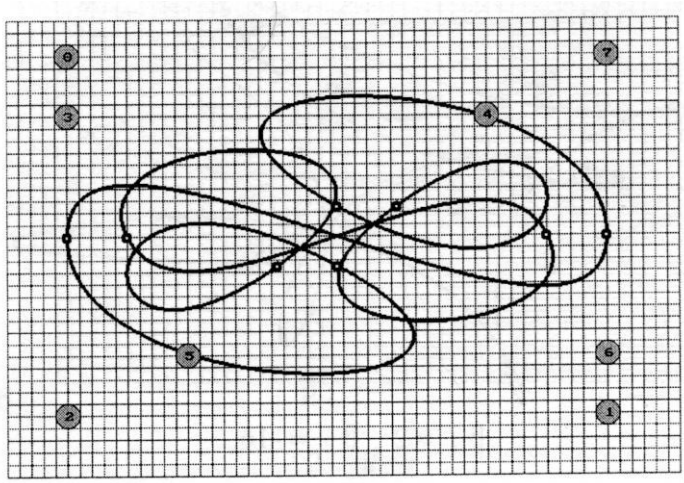
$\beta_1 = 2\,,\ \beta_2 = -16$

Ugyanazzal a karakterisztikus kerettel szerkesztett zárt B-spline görbe látható a *2.29. ábrán,* a *2.30. ábra* pedig az alakparaméterek hatását szemléltetik negatív feszültségértékek esetén.

2.29. ábra

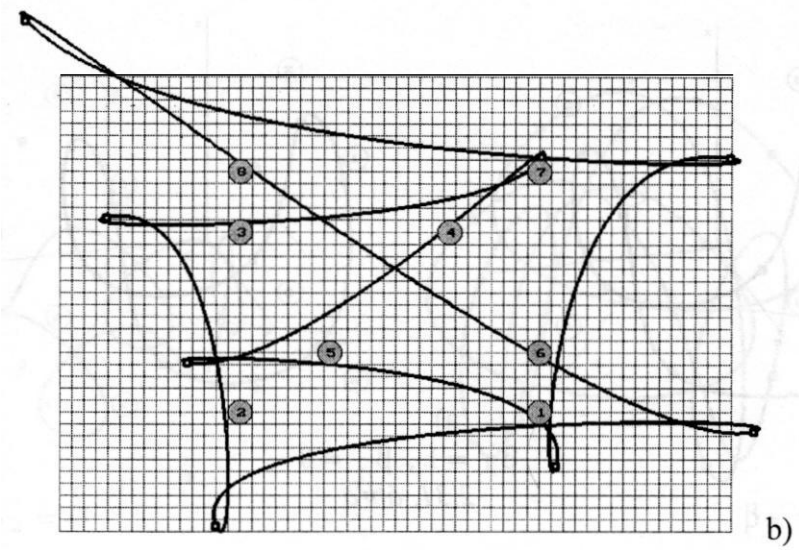


$\beta_1 = 1\,,\ \beta_2 = 0$



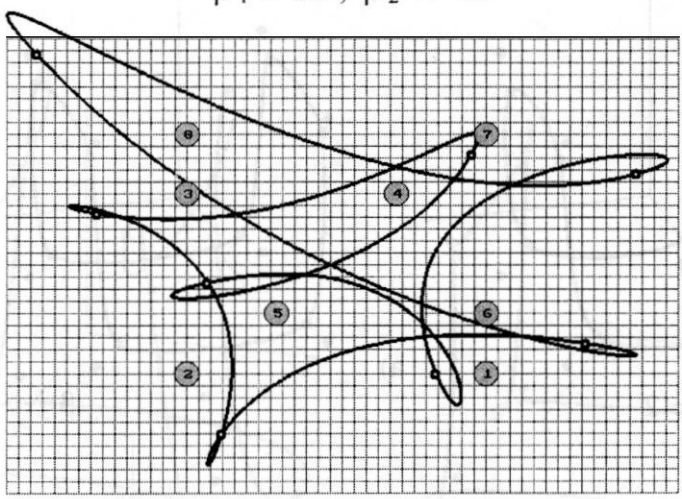
a)

$\beta_1 = 1\,,\ \beta_2 = -4$



b)

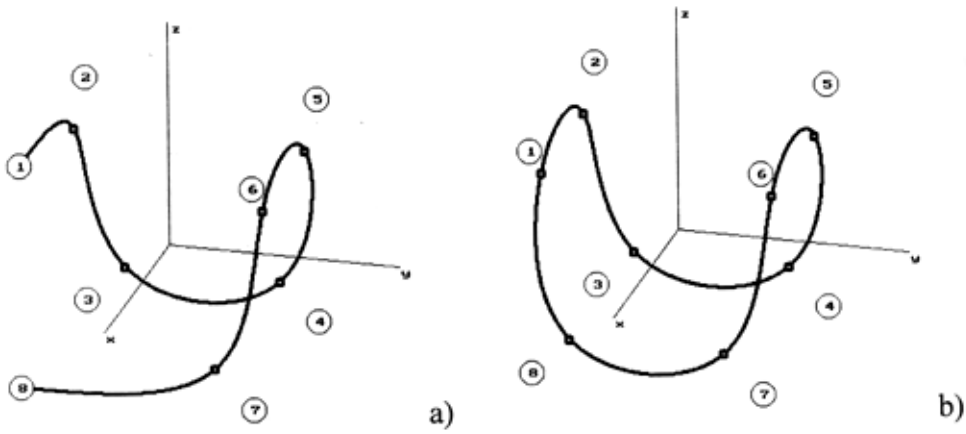
$\beta_1 = 0.5$, $\beta_2 = -4.2$



c)

13_1=2, R2=-15
2.30. ábra

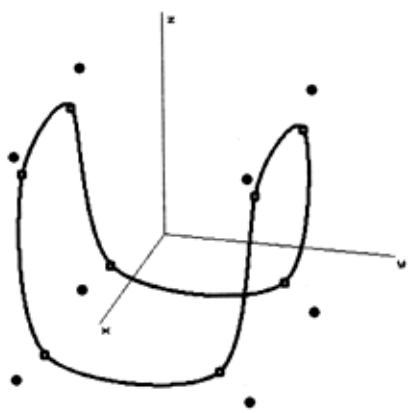
A *2.31 - 2.33. ábrák* az alakparaméterek hatását szemléltetik térgörbék esetén. A *2.31. ábrán* egy kocka néhány éléből álló karakterisztikus kerettel szerkesz- tett nyílt illetve zárt B-spline térgörbe, a *2.32. ábrán* fokozatosan növekvő feszültségű zárt β-spline térgörbék, míg a *2.33. ábrán* β_1=2 eltolással és a megfelelő kritikus érték alatti, illetve fölötti negatív feszültséggel szerkesztett görbék láthatók.



a) b)

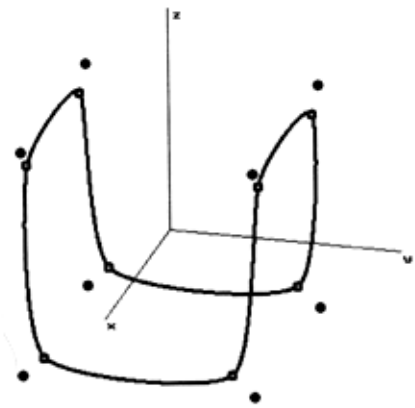
$\beta_1 = 1$, $\beta_2 = 0$

2.31. ábra



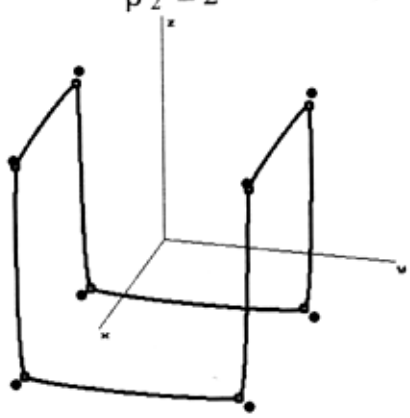
a)

$\beta_2 = 2$



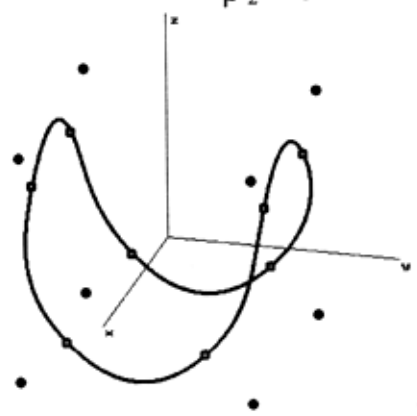
b)

$\beta_2 = 5$



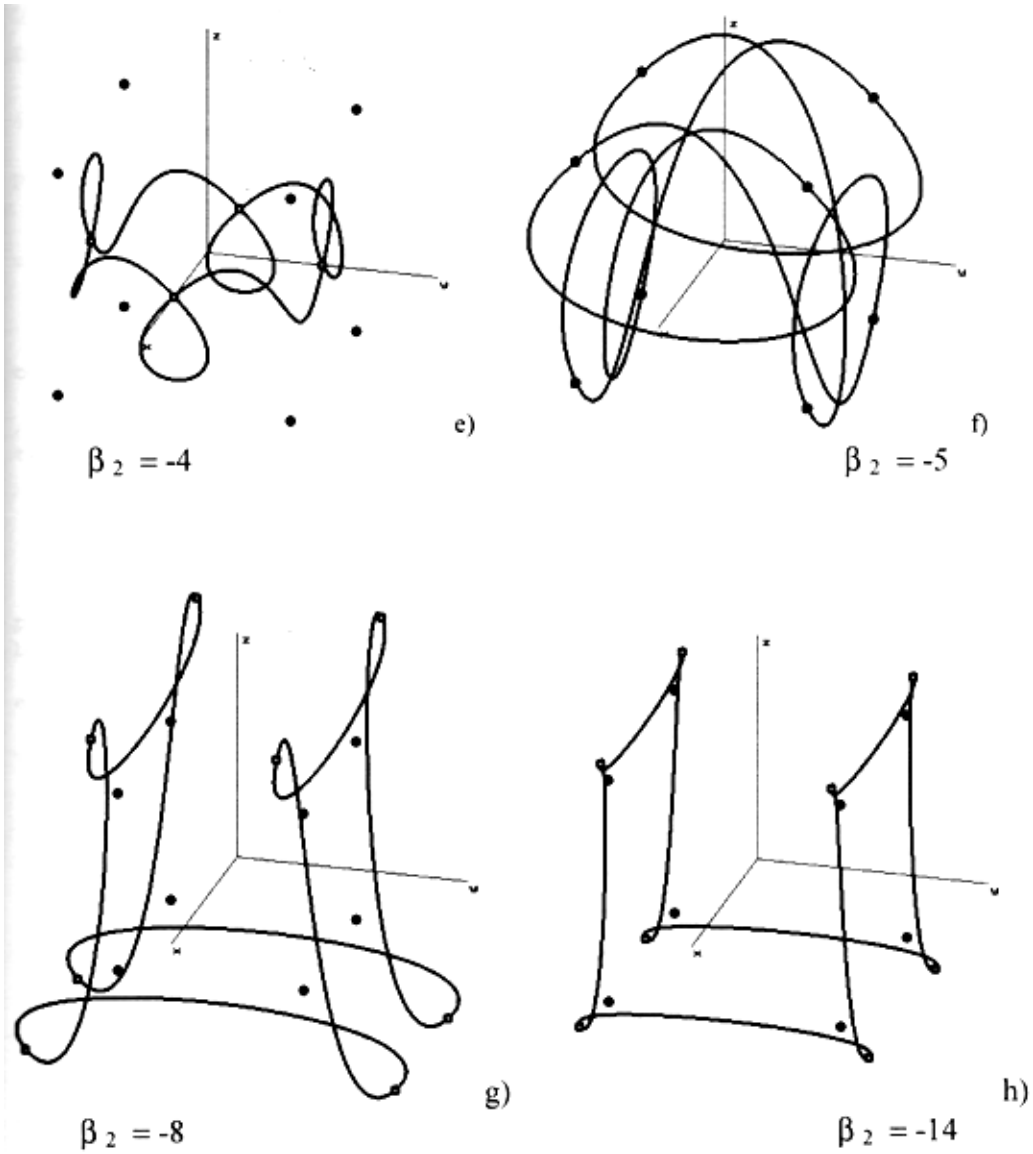
c)

$\beta_2 = 20$

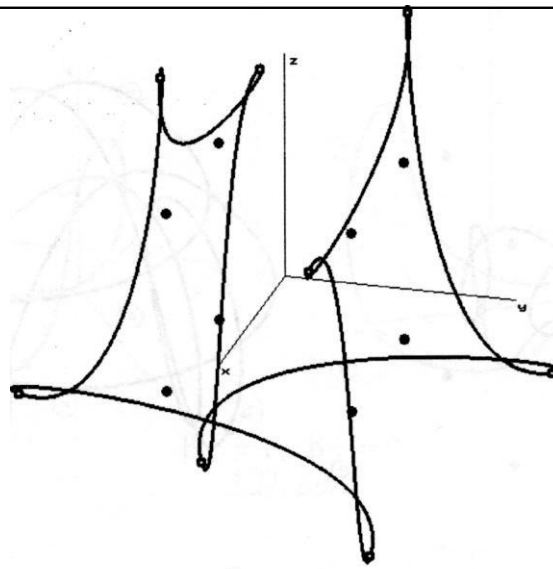


c

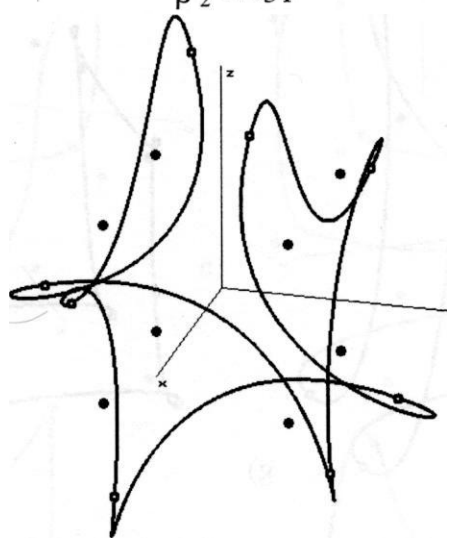
$\beta_2 = -2$



2.32. ábra



$\beta_2 = -31$



$\beta_2 = -16$

$\beta_1 = 2$

2.33. ábra

A lemezmellékleten található GB_2D_BT.PAS nevű programmal szerkeszthetők β -spline síkgörbék. Az alábbi forráskódból hiányzik a GB_2D_IP.PAS nevű programmal azonos néhány részlet.

```
 -SN+}

 program gb_2d_bt; {sima síkgörbeszerkesztés β-spline
t                    approximációs eljárással}

' uses
   graph, grafind, crt, dos, kozos, file_kez, konf_2d, eger_kez,
   eger_kp, kerdesek, gomb_kez, kozos_gb;

 const
   valasz: kars = ['B','E','G','Q','T','V','A','C');

 type
   pontr = object (valós koordinátájú síkbeli pont)
     xp, yp: real;
     procedure uj(ux, uy: real);
   end;

 procedure pontr.uj(ux, uy: real);
 begin
   xp:=ux; yp:=uy;
 end;

 yaw
   ff: file of mp;
   e, f: array[1..nm] of real;              .
   nh, fv, bill: byte;
   st, i: integer;
   b12, b13, d: real;
   b: array[-2..1] of real;
   p: array[-2..1] of pontr;
   dp: array[1..nm] of pont;

 procedure kilep;


 .........
 procedure konfig;


 ........
 procedure kpontok; {kontrollpontok megjelenítése}


 .........
 procedure rajz; {görbemegjelenítés}
 const nr=500;
 var
   j, w, nf, ux, uy: integer;
   u, u2, u3, f1, f2, f3, f4: real;
```

```
  function xr(i: integer): integer; {görbepont x koordinátája}
  begin
    xr:=kx0+round((b[-2]*p[-2].xp+b[-1]*p[-
          1].xp+ b[0]*p[0].xp+b[1]*p[1].xp)/d);
          end;

  function yr(i: integer): integer; {görbepont y koordinátája)
  begin
    yr:=ky0-round((b[-2]*p[-2].yp+b[-1]*p(-1].yp+
                   b[0]*P[0].yp+b[1]*P[1].yp)/d);
  end;

  function mezo(bx,by: integer): boolean;
    {kivágási feltétel a szerkesztési mező határain)
  begin
    if (bx<kx0+xx+12) and (bx>kx0-12) and (by<ky0+12) and
        (by>ky0-yx-12)
    then mezo:=true else mezo:=false;
  end;

  procedure delta;
  begin
    b12:=bl*bl; b13:=b12*bl; d:=b13+2*(b12+b1)+b2+1;
  end;

  procedure jel(a: pont); {görbeívek csatlakozásán álló jel}
  begin
    if mezo(a.xp,a.yp) then fillellipse(a.xp,a.yp,3,3);
  end;

begin
  eger_n; delta;
  if abs(d)<0.1 then begin
    if d<0 then b2:=b2-1 else b2:=b2+1;
    delta; end; (nullaosztás elkerülése)
  racs2; setlinestyle(0,0,3); setcolor(15);
  for j:=0 to nr do begin
    u:=j/nr; u2:=u*u; u3:=u2*u;
    fl:=1-3*u+3*u2-u3; f2:=u2*(3-u); f3:=u*(3-u2); f4:=f2-u3;
    b[-2]:=b13*fl; { β-spline súlyfüggvények}
    b[-1]:=b13*(1-fl)+b12*(2-f2)+bl*(2-f3)+b2*(1-f4);
    b[0]:=b12*f2+bl*f3+b2*f4+1-u3;
    b[1]:=u3;
    if qa='N' then nf:=nh else nf:=nh+1;
    for i:=2 to of do begin
      if i=2 then begin
        if qa='N' then
          p[-2].uj((a[1].xp-a[2].xp)/b13+a[1].xp,    (segédvégpont)
                   (a[1].yp-a[2].yp)/b13+a[1].yp)
        else p[-2].uj(a[nh].xp,a[nh].yp);
        for w:=-1 to 1 do p[w].uj(a[i+w].xp,a[i+w].yp);
      end else if i=nh then begin
        if qa='N' then
```

```
        p[1].uj((a[nh].xp-a[nh-1].xp)*b13+a[nh].xp, {segédvégpont}
                (a[nh].yp-a[nh-l].yp)*b13+a[nh].yp)
      else p[1] .uj (a[1] .xp,a[1] .yp);
        for w:=-2 to 0 do p[w].uj(a[i+w].xp,a[i+w).yp);
    end else if i=nh+1 then begin
      p[1].uj (a[2].xp,a[2].yp); p[0] •uj (a[1].xp,a[l].yp);
      p[-1].uj(a[nh].xp,a[nh].yp); p[-2].uj(a[nh-1].xp,a[nh-1].yp);
    end else for w:=-2 to 1 do p[w].uj(a[i+w].xp,a[i+w].yp);
    ux:=xr(i); uy:=yr(i);
    if j>0 then if (mezo(dp[i].xp,dp[i].yp)) and (mezo(ux,uy))
                then line(dp[i].xp,dp[i].yp,ux,uy);
    dp[i] .uj (ux,uy) ;
    if (j=nr) and (mezo(dp[i].xp,dp[i].yp)) then jel(dp[i]);
  end;
end;
setlinestyle(0,0,1);
eger l;
end;


  procedure tar; {kontrollponttár ahonnan új pontot lehet
                  egérrel elvenni)
begin k:=nh+1; a[k].kezd; end;

begin
obj =' G' ;
grindhi; ker_2; eger_k; mm:=imagesize(0,0,2*r,2*r); ef:=0;
repeat
  qu:='N'; fe:=0;
  for k:=0 to nm do a[k].uj(0,0); qk:='N';
  eger_n; racs2; konfig; nh:=a[0].xp; parancs(4); bvalt;
  eger_n; racs2; eger 1;
  if nh>0 then {kezdőkonfiguráció megjelenítése}
    for k:=1 to nh do with a[k] do begin
      getmem(q,mm); (helyfoglalás a bitmap-nek a heap-ben)
      getimage (xr-r, yr-r, xr+r, yr+r, q^) ;
      fg:=0; megj;
  end;
  tar;
  repeat

    repeat par_katt(1,qr); until qr in valasz; eger_n;
      case qr of
      'B':if nh<nm then begin {új kontrollpont beszúrása}

        end;
      'V':begin {változtatandó kontrollpont kiválasztása}

        end;
      'T':if nh>0 then begin {kontrollpont törlése}

        end;
```

```
        'G':if nh>2 then begin {görbemegjelenítés}
              gb [ 3 ] : =1; parancs(4); alak;
              rajz;
              gb[3]:=2; fe:=1;
            end;
        'E':if fe=1 then begin {görbemegjelenítés keret nélkül)
              gb[2]:=1; parancs(4); kpl; cleardevice;
              if qm='I' then begin racs2; rajz; kpontok; end
                        else rajz;              .
              setcolor(7); outtextxy(5,5,'ESC - vissza');
              repeat qm:=readkey; until ord(qm)=27;
              ker_2; bvalt;
              gb[2]:=0;
            end;
        'A':begin {eltolásmódosítás}
              gb[7]:=1; bvalt;
              eltl; if bl<0.05 then b1:=0.05; if bl>20 then bl:=20;
              gb[7] :=0; lap(208,5);
            end;
        'C':begin {feszültségmódosítás}
              gb[8]:=1; bvalt;
              fesz;
              gb[8]:=0; lap(208,5);
            end;
        'Q':begin {kilépés}
              ..........
              end;
        end;
      end;
           if gb[3]=0 then racs2; kpontok; gb[3]:=0; parancs(4); bvalt;
      eger_l; tar;
    until (qk='I') or (qu='I');
  until qu=' N' ;
    ..........
end.
```

A program lehetővé teszi a kontrollpont-konfiguráció és az alakparaméterek interaktív módosítását. 13-spline térgörbéket a GB_3D_BT.PAS program szerkeszt. Az itt kinyomtatott forráskódból hiányzik néhány, a GB_3D_IP.PAS programban megtalálható részlet.

```
{$N+}

program gb_3d_bt; {sima térgörbeszerkesztés 0-spline
                   approximációs eljárással}

uses
  graph, grafind, crt, dos, kozos, file_kez, kerdesek, kozos_3d,
  eger kez, gomb kez, kozos_gb;
```

```
const
  nm=99
  ;
  valasz: kars = ['B','E','G','Q','T','V','A','C'];

type
  pont3 = object {térbeli pont objektum)


  kpont = object(pont3) {kontrollpont objektum}


  mp = array [ 0 .. nm] of kpont;


  a: mp; ff: file of mp;
  e, f, g: array[1..nm] of real;
  nh, fv, fk: byte;
  st, i: integer;
  b12, b13, d: real;
  b: array(-2..1] of real;
  p: array[-2..1] of pont3;
  dp: array[1..nm] of pont;

procedure kpont.koord; {kontrollpont vetülete és
mélysége}


procedure pont3.uj(ux, uy, uz: real);
begin
  xp:=ux; yp:=uy; zp:=uz; end;

procedure kpont.megj; {kontrollpont megjelenítése}

procedure kpont.sorr(un: integer);


procedure sorrend; {mélység szerinti sorrendbehelyezés)


procedure kilep; procedure konfig; procedure kpontok;


{kontrollpontok megjelenítése}


procedure rajz; {görbemegjelenítés}
:must nr=500;
var
  j, w, nf, ux, uy: integer;
  u, u2, u3, f1, f2, f3, f4: real;
```

```
  function xx(i: integer): real; {görbepont x koordinátája}
  begin
    xx:=(b[-2]*p[-2].xp+b[-1]*p[-
  1].xp+b[0]*p[0].xp+b[1]*p[1].xp)/d; end;

  function yy(i: integer): real; {görbepont y koordinátája}
  begin
    yy:=(b[-2]*p[-2].yp+b[-1]*p[-
  1].yp+b[0]*p[0].yp+b[1]*p[1].yp)/d; end;

  function zz(i: integer): real; {görbepont z koordinátája)
  begin
    zz:=(b[-2]*p[-2].zp+b[-1]*p[-1].zp+b[0]*p[0].zp+b[1]*p[1].zp)/d;
  end;

  function mezo(bx,by: integer): boolean;
    (kivágási feltétel a szerkesztési mező határain)
  begin
    if (bx<kx0+xm) and (bx>kx0) and (by<ky0) and (by>ky0-ym)
    then mezo:=true else mezo:=false;
  end;

  procedure delta;
  begin
    b12:=bl*bl; b13:=b12*b1; d:=b13+2*(b12+b1)+b2+1;
  end;

  procedure jel(a: pont); {görbeívek csatlakozásán álló jel}
  begin
    if mezo(a.xp,a.yp) then fillellipse(a.xp,a.yp,3,3);
  end;

begin
  eger_n; delta;
  if abs(d)<0.1 then begin
    if d<0 then b2:=b2-1 else b2:=b2+1;
    delta; end; {hullaosztás elkerülése}
  racs; if gb[2]=1 then nyil; setlinestyle(0,0,3); setcolor(15);
  for j:=0 to nr do begin
    u:=j/nr; u2:=u*u; u3:=u2*u;
    f1:=1-3*u+3*u2-u3; f2:=u2*(3-u); f3:=u*(3-u2); f4:=f2-u3;
    b[-2]:=b13*f1; {β-spline súlyfüggvények}
    b[-1]:=b13*(1-f1)+b12*(2-f2)+b1*(2-f3)+b2*(1-f4);
    b[0]:=b12*f2+b1*f3+b2*f4+1-u3;
    b[1] :=u3;
    if qa='N' then nf:=nh else nf:=nh+1;
    for i:=2 to of do begin
      if i=2 then begin
        if qa='N' then
          p[-2].uj((a[1].xp-a[2].xp)/b13+a[1].xp,       {segédvégpont}
           (a[1].yp-a[2).yp)/b13+a[1].yp,
           (a[1].zp-a[2].zp)/b13+a[1].zp)
      else p[-2].uj(a[nh].xp,a[nh].yp,a[nh].zp);
```

```
      for w:=-1 to 1 do p[w].uj(a[i+w].xp,a[i+w].yp,a[i+w].zp);
      end else if i=nh then begin
       if qa='N' then
          p[1].uj((a[nh].xp-a[nh-1].xp)*b13+a[nh].xp, (segédvégpont)
                  (a[nh].yp-a[nh-1].yp)*bi3+a[nh].yp,
                  (a[nh].zp-a[nh-1].zp)*b13+a[nh].zp)
       else p[1] .uj (a[1] .xp,a[1] .yp,a[1] .zp);
            for w:=-2 to 0 do p[w].uj(a[i+w].xp,a[i+w].yp,a[i+w].zp);
      end else if i=nh+1 then
       begin
         p[i].uj (a[2] .xp,a[2].yp,a[2] .zp);
         p[0].uj(a[1].xp,a[1].yp,a[1].zp);
         p[-1].uj(a[nh].xp,a[nh].yp,a[nh].zp);
         p[-2].uj(a[nh-1].xp,a[nh-1].yp,a[nh-1].zp);
       end else
          for w:=-2 to 1 do [w].uj(a[i+w].xp,a[i+w].yp,a[i+w].zp);
       vetit(xx(i),yy(i),zz(i)); ux:=xt; uy:=yt;
       if j>0 then if (mezo(dp[i].xp,dp[i].yp)) and (mezo(ux,uy))
                   then line(dp[i].xp,dp[i].yp,ux,uy);
       dp[i] .uj (ux,uy) ;
      if (j=nr) and (mezo(dp[i].xp,dp[i].yp)) then jel(dp[i]);
    end;
  end;
  setlinestyle(0,0,1);
  eger_l;
end;

procedure inform;

  ........
procedure fg; {vetítési irány váltortatásának vezérlése}

  .........
begin
  obj:='G';
  grindhi; keret(kx0, ky0); eger_k;
  repeat

    ..........
    repeat par_katt(1,qr); until qr in valasz; eger_n;
      case qr of
      'B':if nh<nm then begin (új kontrollpont beszúrása)
             .........

          end;
      'V':begin (váltortatandó kontrollpont kiválasztása)


          end;.........
      'T':if nh>0 then begin {kontrollpont törlése}

         end; .........
      'G':if nh>2 then begin {görbemegjelenítés)

          end;
```

```
        'E':if fk=1 then begin (görbemegjelenítés keret nélkül)

          end;
      'A':begin {eltolásmódosítás}
            gb[7]:=1; bvalt;
            eltl; if bl<0.05 then b1:=0.05; if bl>20 then bl:=20;
            gb[7]:=0; lap(208,5);
          end;
      'C':begin {feszültségmódosítás}
            gb[8]:=1; bvalt;
            fesz;
            gb[8] :=0; lap(208,5);
          end;
      'Q':begin {kilépés)

          end;
      end;       .....
      if fk=0 then racs; kpontok; parancs(4); bvalt; eger_l;
    until (qk='I') or (qu='I');
  until qu=' N' ;
  closegraph;
end.
```

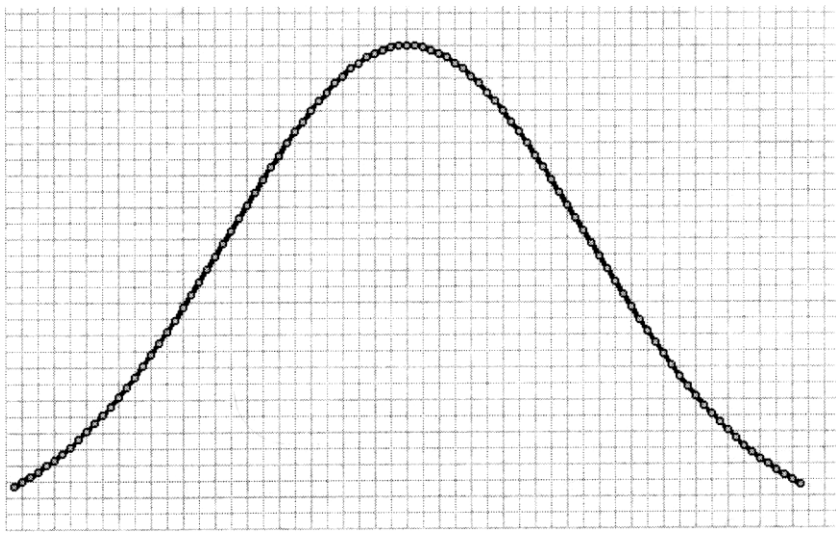
A β-spline közelítés tovább általánosítható, folytonosan változó alakparaméterek használata irányában. Nyílt görbe esetén a felhasználó az n+l kontrollpont között található *n* csatlakozási pontban és a görbe végpontjaiban megadja az eltolása $\alpha_{-1}, \alpha_0, \alpha_1, ..., \alpha_{n-1}, \alpha_n$ illetve a feszültség $\varphi_{-1}, \varphi_0, \varphi_1, ..., \varphi_{n-1}, \varphi_n$ értékeit. Az alakparaméterek a görbeívek mentén folytonosan változnak a

képletek szerint, ahol az *s* függvény első és második deriváltjainak értéke a paraméterintervallum határain nulla.

$$\begin{aligned} \beta_{1i}(u) &= (1-s(u))\alpha_{i-1} + s(u)\alpha_i \\ \beta_{2i}(u) &= (1-s(u))\varphi_{i-1} + s(u)\varphi_i \\ s(u) &= 10u^3 - 15u^4 + 6u^5 \quad ; \quad u \in [0,1], \quad i = 0,1,\ldots,n \end{aligned} \tag{2.6}$$

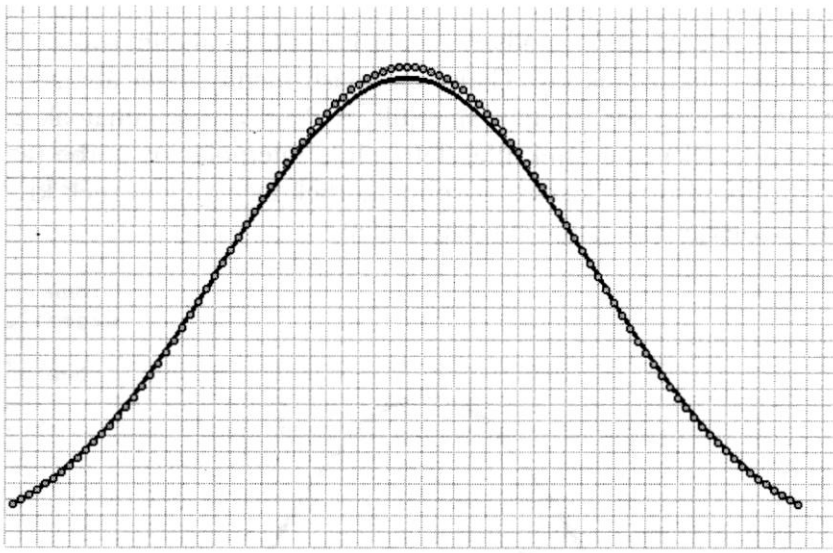
23.8. Alkalmazás

A különböző interpolációs módszereket a következő alkalmazással is összehasonlíthatjuk. Tekintsünk egy Gauss eloszlású mennyiséget ábrázoló görbét, amelynek viszonylag sok pontját ismerjük. A *2.34. ábrán* láthatóak az ismert pontok (apró körök) és a rájuk illesztett spline interpolációs görbe (az ugyanerre a kontrollpontsorra illesztett B-spline görbék ugyanígy néznek ki - a kontrollpontok sűrűsége és a görbe simasága miatt).



2.34. ábra

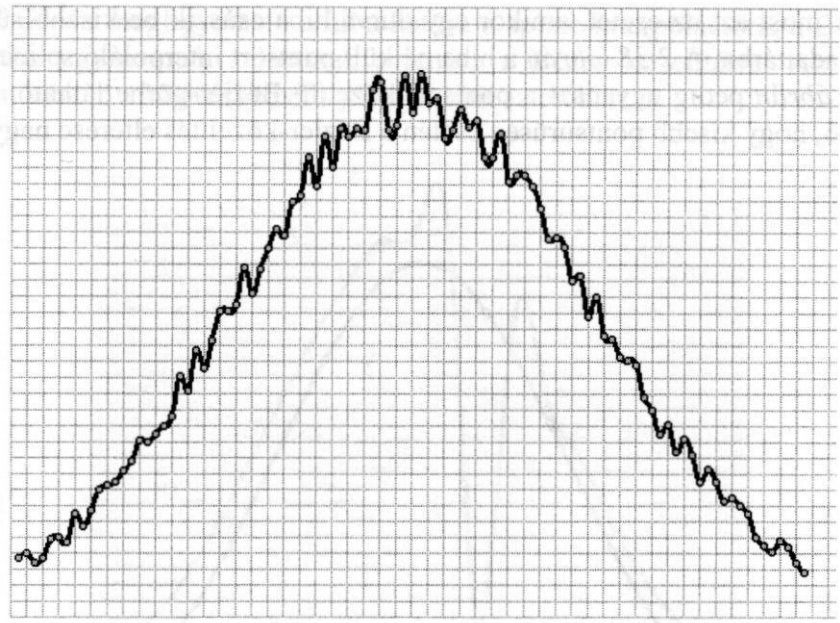
A *2.35. ábrán* ugyanazokra a pontokra szerkesztett Bézier görbe található. Látható, hogy a sűrű kontrollponthalmaz ellenére a közelítő görbe valamelyest eltér az adott értékektől. Arra lehet következtetni, hogy a spline interpoláció ebben az esetben jobb.



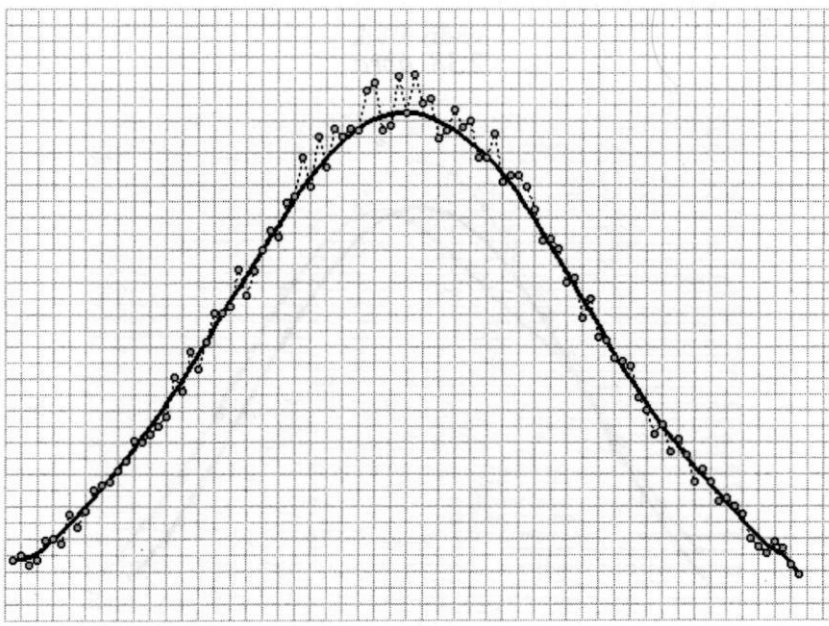
2.35. ábra

Most tegyük fel, hogy a görbét nem ismerjük, hanem egy sor mérési adat ál rendelkezésünkre, amelyet bizonyos zavaró tényezők által okozott, úgynevezet mérési zaj befolyásol. Jelen esetben a fenti görbe ordinátáinak véletlenszeri módosításával modellezzük a "zajt".

A *2.36. ábrán* a mérési adatokra spline interpolációval illesztett görbe látható, míg a *2.37. ábrán* a mérési adatokat Bézier görbével közelítettük. Nyilvánvaló a Bézier közelítés alkalmazásának előnye az adott körülmények között, elsősorban a sokkal jobb simaság miatt, főleg ha a mérési eredményeket további feldolgozásnak vetjük alá, amelynek során deriválásra is sor kerül, ugyanis ez a művelet annyira felerősítheti a zajt, hogy használhatatlanná teszi az adatokat. A Bézier-közelítés kiszűri a véletlenszerű, váltakozó előjelű zajt és jó becslést nyújt a görbének a zaj rátevődése előtti alakjáról.

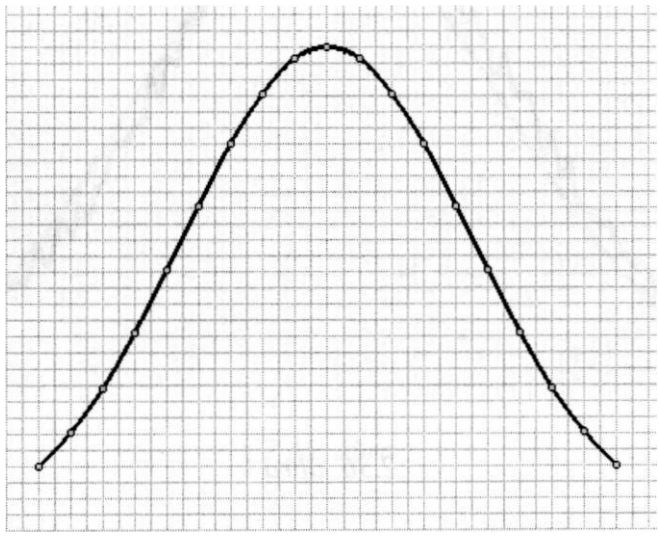


2.36. ábra

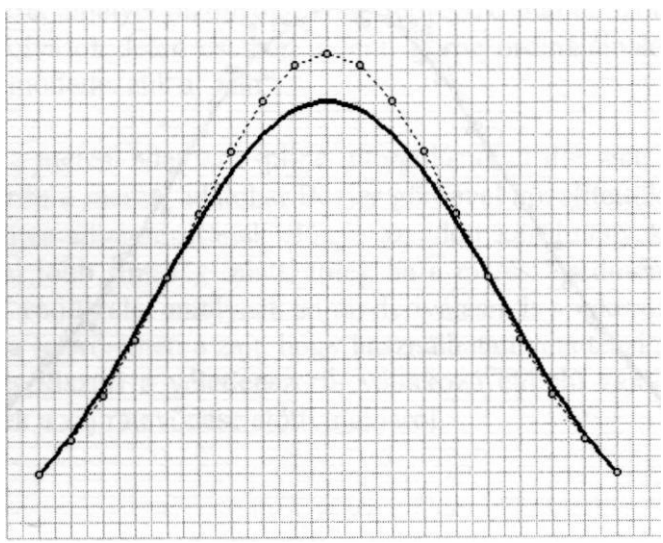


2.37. ábra

Vegyük most azt az esetet, amikor egy durva hiba csúszik be kisebb sűrűségű mérési adattárba. A *2.38. ábrán* a hiba nélküli pontsort interpoláló spline görbe. a *2.39. ábrán* pedig ugyanazt a pontsort közelítő Bézier-görbe látható. Megfigyelhető, hogy kisebb pontsűrűség esetén a Bézier-közelítés eltérése nagyobb.

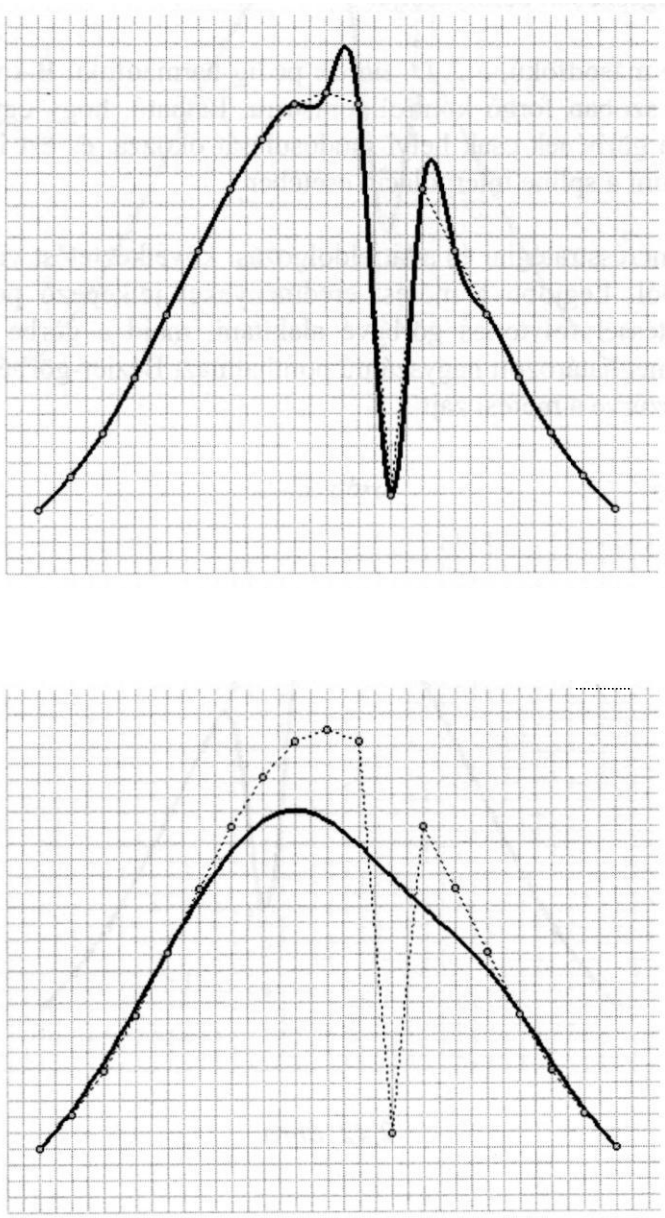


2.38. ábra



2.39. ábra

A *2.40. ábrán* megfigyelhető a hiba hatása a spline interpolációra, a *2.41. ábrán* pedig ugyanannak a hibának a hatása a Bézier-közelítésre. Az utóbbit egyetlen hiba jóval kevésbé befolyásolja mint az előbbit.

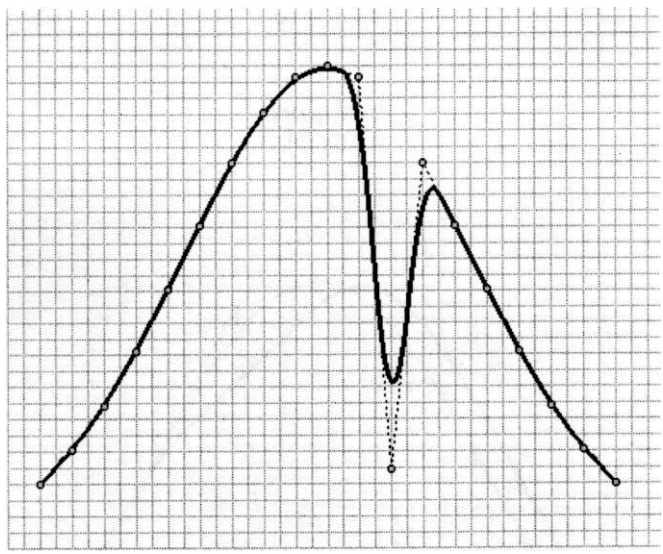


2.41. ábra

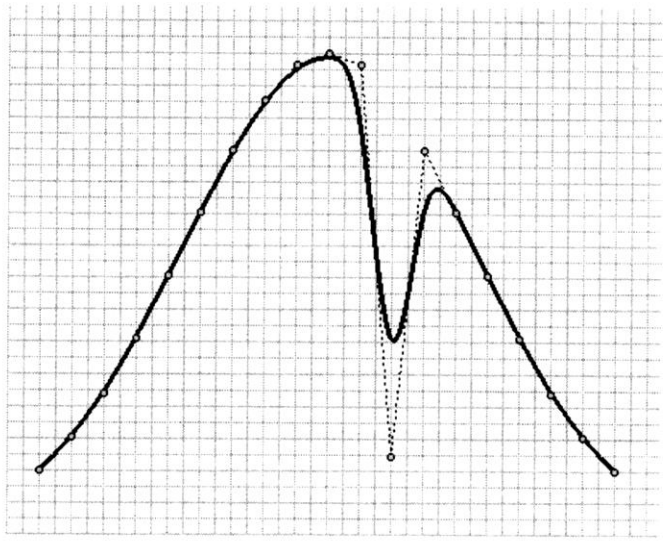
Más (például formatervezési) szempontból az is megfigyelhető, hogy a spline görbét egy kontrollpont elmozdítása a kontrollponthoz tartozókon kívül néhány szomszédos intervallumon is módosítja (csillapított mértékben), míg a Bézier-görbét teljes egészében befolyásolja.

A *2.42. ábrán* másodfokú, a *2.43. ábrán* pedig harmadfokú B-spline görbét illesztettünk a szóban forgó kontrollpontsorra. Látható, hogy egy kontrollpont elmozdítása a görbének csak helyi módosulását okozza, és ezt is jóval kisebb mértékben, mint a spline interpoláció esetében.

Egy értékes mérésstratégiai és adatfeldolgozási következtetést vonhatunk le az elmondottakból. Legjobb a mérési eredményeket jelképező ponthalmazt B-spline közelítéssel folytonos görbévé alakítani, majd a görbe előállításának segítségével meghatározott nagyszámú ponthalmazt Bézier-görbével közelíteni, ily módon kiszűrve a mérési zajt.



2.42. ábra



2.43. ábra

3. Sima felületek interaktív előállítása

A felületeknek a számítógépes szerkesztés szempontjából legelőnyösebb megadási módja a paraméteres előállítás, amelyben a felület pontjainak koordinátái kétváltozós függvények:

$$\mathbf{r}(u, v) = \mathbf{r}\big(x(u, v),\ y(u, v),\ z(u, v)\big)\ ; \qquad (u, v) \in D \subset \Re^2\ . \tag{3.1}$$

Ide sorolhatjuk az úgynevezett Euler-Monge előállítást is, amelyben a paraméterek maguk az x és y koordináták (domborzat jellegű felületek):

$$z = z(x\ ,\ y)\ ; \qquad (x, y) \in D \subset \Re^2\ . \tag{3.2}$$

A felületek leggyorsabb és egyik legszemléletesebb megjelenítési módja az, ha a felületre görbékből álló hálózatot rajzolunk. Paraméteres előállítású felületre görbét úgy rajzolhatunk, hogy a paraméter értelmezési tartományában értelmezünk újabb paraméter bevezetésével egy síkgörbét, majd ennek előállítását behelyettesítjük a felület képletébe:

$$\mathbf{r}(t) = \mathbf{r}\big(u(t),\ v(t)\big)\ ; \qquad t \in I \subset \Re\ . \tag{3.3}$$

Az ***u = áll.*** és v = *áll.* egyenesekből a fenti módon származtatott görbéket paramétergörbéknek nevezzük. Ezek érintővektorai a felület 3.1. előállításának az illető paraméter szerinti parciális deriváltjai:

$$\begin{aligned} \mathbf{r}_u &= \frac{\partial \mathbf{r}}{\partial u}(u\ ,\ v) = \left(\frac{\partial x}{\partial u}\ ,\ \frac{\partial y}{\partial u}\ ,\ \frac{\partial z}{\partial u}\right) \\ \mathbf{r}_v &= \frac{\partial \mathbf{r}}{\partial v}(u\ ,\ v) = \left(\frac{\partial x}{\partial v}\ ,\ \frac{\partial y}{\partial v}\ ,\ \frac{\partial z}{\partial v}\right) \end{aligned}\ . \tag{3.4}$$

A 3.3 előállítású tetszőleges felületi görbe érintője

$$\mathbf{r}'(t) = \mathbf{r}_u u'(t) + \mathbf{r}_v v'(t) \tag{3.5}$$

az illető ponton áthaladó két paramétervonal érintője által meghatározott síkban (érintősík) található. A felület normálisvektorát (az érintősíkra merőleges vektor) az

$$\mathbf{n} = \mathbf{r}_u \times \mathbf{r}_v \qquad (3.6$$

képlet adja.

Egy felületet simának nevezünk, ha érintősíkja a felület minden pontjában értelmezett, és változása folytonos. További simasági követelmény lehet az, hogy a másodrendű parciális deriváltak is folytonosak legyenek.

Bizonyos követelményeknek eleget tevő felületek előállításának meghatározását többféleképpen megpróbálhatjuk:

- interaktív paraméteres előállítással
  A 3.2 vagy 3.3 előállításban szereplő kétváltozós analitikus függvényeket megadva, ezek konstansait módosítva változtatjuk a felület alakját. Előnye, hogy analitikus előállítás lévén, a kapott felület sima lesz, hátránya, hogy a szóba jöhető függvények sokfélesége és az együttható-módosítás kevéssé előrelátható volta miatt nem létezik biztosan célravezető tervezési stratégia, a felhasználó leginkább tapasztalatára és intuíciójára támaszkodhat.
- görbecsaládok segítségével történő előállítás
  Tekintsünk két görbét, az egyiket mozgassuk a másik mentén. Az elsőt leírógörbének, a másodikat pályagörbének nevezzük. A leírógörbét nem tekintjük merevnek, hanem a pályagörbe menti helyzetétől függően változtatja alakját. Ez azt jelenti, hogy két paraméterünk van, az egyik a leírógörbe saját paramétere, a másik a pályagörbe ívhossza, amelytől a leírógörbe ugyancsak függ, tehát egy felületet értelmeztünk. Ez a módszer eredményes lehet például már létező görbe felületű test (modell) számítógépre viteléhez, ha rajta valamely görbecsalád paraméteres előállítását sikerül meghatározni.
- kontrollpontrácsra illesztéssel
  Ez az egyik legkényelmesebb és legcélravezetőbb felületmodellező módszer. Megfelelő kontrollpontrács szerkesztése után a felületet, úgy, ahogy az előző fejezetben görbéket is megfelelő interpolációs vagy approximációs módszerrel állítjuk elő. A kontrollpontokat összekötő törtvonalak hálóját karakterisztikus poliédernek is nevezik, bár a háló szemei általában nem síkidomok.

A KONF_3D.PAS nevű unit tartalmazza a kontrollpontrács és a felületmegjelenítési háló értelmezését valamint a kontrollpontok kiválasztását és megjelenítését végző eljárásokat. A kontrollpontokat sorszámuk beütésével vagy - gyorsabban - egérrel lehet kiválasztani a képernyő bal felső sarkában látható térkép megfelelő pontjára kattintva. A kiválasztott pont színe a térképen is, a szerkesztési mezőben is megváltozik, a képernyő alján pedig megjelenik koordinátáinak aktuális értéke. Az 'Esc' billentyű leütésével meg lehet szakítani a helyzetváltoztatást és a kontrollpont a módosítási eljárás hívása előtti helyén marad.

```
{$N+}
unit konf 3d; {kontrollpontstruktúra meghatározása a térben}

interface

uses graph, crt, kozos, kozos_3d, eger_kez;

const
 um=7; vm=4; nm=16;

type
 pont3 = object
   xp, yp, zp: single;
   procedure uj(ux, uy, uz: single);
 end;

 p3 = Apont3;

 kpont = object(pont3) xr,
   yr: integer; procedure
   koord; procedure megj(s:
   byte);
 end;

 p4 = array[1..4] of pont;

 szem = object {a felület négy pontja által meghatározott
                torz négyszög képsíkra eső vetülete}
   c: p4;
   zm: single;
   n: integer;
   procedure uj(uc: p4; uz: single);
   procedure sr(un: integer);
 end;

 kpsz = array[1..(um+1)*(vm+1)] of szem; {kontrollpontrács-szemek}
 rsz = array[1..nm*nm] of szem; (hálószemek egy kontrollpontrács-
                                 szemben)
 psz = ^rsz;
 halo = array[1..(um+1)*(vm+1)] of psz; (megjelenítési háló)
```

```
  mp = array(0..um,0..vm) of kpont; {kontrollpontok}
  mr = array[0..(um+1)*nm,0..(vm+1)*nm] of p3; {megjelenítési háló
                                                térbeli csomópontjai}

var
  a:  mp; ff: file of mp;
  fl, fr, fp: byte;
  i,j, km, lm, kp, ip, p, q, r: integer;
  rp: mr; hl: halo; kp_sz: kpsz;
  b:  byte;

procedure sorszam(nu, nv: byte);
procedure kpontok (nu, nv, ztu, ztv: byte);
procedure kp_kor(i,j,s: byte); procedure
kpracs(nu, nv: byte); procedure
kp_r_katt(nu, nv: byte); procedure
inform(i,j: byte; szg: string);

implementation

procedure kpont.koord; {képernyőkoordináták}
begin
  vetit(xp, yp, zp) ; xr:=xt; yr:=yt;
end;

procedure pont3.uj(ux, uy, uz: single);
begin
  xp:=ux; yp:=uy; zp:=uz; end;

procedure kpont.megj(s: byte); {kontrollpontok megjelenítése}
var sf: string;
begin
  setfillstyle(i,$); fillellipse(xr,yr,4,4);
end;

procedure szem.uj(uc: p4; uz: single);
begin c : =uc; zm:=uz; end;
procedure szem.sr(un: integer);
begin n:=un; end;
procedure sorszam(nu, nv: byte);
  {kiválasztandó kontrollpont sorszámának lekérdezése}
var ki: boolean;
begin
  lap(208,5); setcolor(0); outtextxy(262,26,'Sorszám:');
  setcolor(12); outtextxy(350,12,'k'); setcolor(0);
  outtextxy(350,30,'1');
  ablak(360,28,383,41);
  repeat
    ablak(360,10,383,23); szx:=ord(readkey)-48; setcolor(15);
    ki:=(szx>=0) and (szx<=nu);
    if ki then begin str(szx,kn); outtextxy(368,13,kn); end;
  until ki;
  setcolor(0); outtextxy(350,12,'k'); setcolor(12);
```

```
  outtextxy(350,30,'1');
  repeat
    ablak(360,28,383,41); szy:=ord(readkey)-48; setcolor(15);
    ki:=(szy>=0) and (szy<=nv);
    if ki then begin str(szy,kn); outtextxy(368,31,kn); end;
  until ki; delay (500) ;
end;

procedure kpontok(nu, nv, ztu, ztv: byte);
    {kontrollpontstruktúra megjelenítése
      ztu=1 ha a felület u irányban zárt, 0 ellenkező esetben)
begin
  setcolor(15);
  for k:=0 to nu-ztu do for 1:=0 to nv-ztv do a[k,l].koord;
  if fe=0 then begin
    setcolor(7);
    for 1:=0 to nv-ztv do begin
      vetit(a[0,1].xp,a[0,1].yp,a[0,1].zp); moveto(xt,yt);
      for k:=1 to nu-ztu do begin
        vetit(a[k,1].xp,a[k,1].yp,a[k,1].zp); lineto(xt,yt); end;
      if ztu=1 then
        vetit(a[0,1].xp,a[0,1].yp,a[0,1].zp); lineto(xt,yt);
    end;
    for k:=0 to nu-ztu do begin
      vetit(a[k,0].xp,a[k,0].yp,a[k,0].zp); moveto(xt,yt);
      for 1:=1 to nv-ztv do begin
        vetit(a[k,l].xp,a[k,1].yp,a[k,1].zp); lineto(xt,yt); end;
      if ztv=1 then
        vetit(a[k,0].xp,a[k,0].yp,a[k,0].zp); lineto(xt,yt);
    end;
    setcolor(15);
    for 1:=0 to nv-ztv do
      for k:=0 to nu-ztu do a[k,1].megj(0);
  end;
end;
procedure kp_kor(i,j,s: byte);
begin
  setfillstyle(1,$); setcolor(0); fillellipse(50+10*i,36-8*j,3,3);
  if s>8 then a[szx,szy].megj(s);
end;

procedure kp_racs(nu, nv: byte);
    {kontrollpontrács megjelenítése egérrel való kiválasztáshoz)
var sf: string;
begin
  for k:=0 to nu do begin
    str(k, sf); outtextxy(47+10*k,44,sf);
    for 1:=0 to nv do kpkor(k,1,7);
  end;
  for 1:=0 to nv do begin
    str(1, sf); outtextxy(36,34-8*1,sf);
  end;
  line(150,45,150,5); line(150,5,152,9); line(150,5,148,9);
```

```
  line(130,27,175,27); line(175,27,171,29);
  line(175,27,171,25); setcolor(12); outtextxy(20,20,'1');
  outtextxy(60+nu*10,44,'k'); outtextxy(170,30,'u');
  outtextxy(140,4,'v'); end;

procedure kp_r_katt(nu, nv: byte); {kontrollpont egérrel való kivá-
lasztása)
var ny: byte;
begin
  for k:=0 to nu do
    for 1:=0 to nv do begin
       kp_katt(50+10*k,36-8*1,ny);
       if ny=1 then begin szx:=k; szy:=1; qr:='V'; b:=1; end;
    end;
end;

procedure inform(i,j: byte; szg: string);
      {kontrollpont térbeli helyének kiírása}
var ksl, ks2, kx, ky, kz: string;
begin
  setfillstyle(1,11); bar(0,465,640,479);
  setcolor(0); str(szx,ksl); str(szy,ks2); str(a[i,j].xp:3:0,kx);
  str(a[i,j].yp:3:0,ky); str(a[i,j].zp:3:0,kz);
  outtextxy(130,470,'Sorszám: '+ksl+','+ks2+szg+'x = '+kx+
            '   y = '+ky+'    z = '+kz);
end;

end.
```

A kontrollpont-konfigurációt file-ba lehet menteni más alkalommal való felhasználás céljával, az ide tartozó kommunikációt a felhasználóval a KOMM_FEL.PAS nevű unit eljárásai végzik:

```
{$N+)
unit komm_fel; {konfigurációval kapcsolatos kommunikáció felületszer
                kesztésnél)

interface
uses crt, kozos, kerdesek, file kez, konf 3d, eger kez;

procedure kilep(ext: string);
procedure konfig(ext: string);

implementation

procedure kilep(ext: string);
      {felhasználó válaszától függő konfigurációmentés kilépéskor)
begin

  kerdez(200,200,'  Menteni a konfigurációt ?','gen','em','I','N');
  repeat kattint(200,200,'I','N',qm);
```

```
  until (qm='I') or (qm='N');
  if qm='I' then begin
    eger n; ment(ext);
    if v='I' then begin
       assign(ff,kn); rewrite(ff); write(ff,a); close(ff); end;
    eger_l;
  end;
end;

procedure konfig(ext: string);
       (felhasználó válaszától függő konfigurációbetáplálás)
begin
  repeat
    qf=' ';
    kerdez(200,200,'       Konfiguráció ?','j','árolt','U','T');
    repeat kattint(200,200,'U','T',qf);
    until (qf='U') or (qf='T');
    if qf='T' then begin
       eger_n; olvas(ext);
       if ko=1 then begin assign(ff,kn); reset(ff); read(ff,a);
                          close(ff); end;
       eger_1;
    end else ko:=1;
  until ko=1;
end;
end.
```

3.1. Felületek megjelenítése

Amint már említettük, a felületeket paramétervonalak hálójának segítségével ábrázoljuk. A drótvázas megjelenítés az egyszerűbb, de kevésbé szemléletes (bonyolult felületek ábrázolásakor meglehetősen kusza rajzolatot eredményez). Ez úgy történik, hogy a felület paramétergörbéit térgörbékként megjelenítjük, az esetleges takarásokat nem véve figyelembe (mintha a felület átlátszó lenne).

A képies, "tele" megjelenítéshez ugyancsak a paramétergörbék hálójából indulunk ki. A hálószemeket síkidomoknak tekintjük (ez általában nem igaz, de eléggé sűrű háló esetén a planeitástól való eltérés elhanyagolható) és képsíkhoz viszonyított magasságuknak megfelelő fordított sorrendben megjelenítjük, így a megfigyelőhöz közelebb álló felületelemek fogják a távolabb levőket takarni és helyes képet kapunk. Ez a sorrendbe helyezés azonban sűrű háló esetén temérdek gépidőt igényel, amelynek nagy része fölösleges, ugyanis a legtöbb felületelem nem is kerül egymás fölé. Ezért érdemes a következő eljárást használni:

- a felületet egy durvább hálóval felosztjuk. Ez az interaktív szerkesztésnél lehet éppen a kontrollpontrács csúcsait összekötő görbék hálója
- ennek a hálónak a szemeit súlypontjuk képsíkhoz viszonyított magasságának megfelelő fordított sorrendbe rendezzük
- a kapott sorrendben megjelenítjük a hálószemekhez tartozó felületdarabokat:
  - a hálószemet felosztjuk sűrű hálóval
  - ha a nagy hálószem húrnégyszögének vetülete konvex, akkor a megjelenítés sorrendje közömbös, az elemi felületdarabok nem fogják egymást fedni
  - ha a nagy hálószem húrnégyszögének vetülete konkáv, akkor a hozzátartozó sűrű háló szemeit is megfelelő sorrendben kell megjeleníteni

Másik gyorsítási lehetőség, hogy a már egyszer meghatározott sorrendet használjuk kiindulási sorrendnek a vetítési irány változtatása utáni megjelenítéshez, ugyanis így kisebb lesz a helycserék száma és kevesebb ciklust igényel a rendező eljárás.

A RAJZ F3D.PAS unit tartalmazza a felületek megjelenítését végző eljárásokat.

```
(SN+)
unit rajz f3d; (szerkesztett felületek megjelenítése}

interface

uses graph, kozos, kozos_3d, konf_3d;

procedure megjelen(nu, nv, n, kq, ztu, ztv, bs: byte);

implementation

var dh: array[1..5] of pont; t: byte;

procedure megjelen(nu, nv, n, kq, ztu, ztv, bs: byte);

  procedure rajzol;
  var
   x, y, z : array [ 1.. 4 ] of real;
   sk: p4;
   ql, q2: byte;
   procedure negyszog; {vetített hálószem meghatározása)
```

```
  begin
    x[2]:=rp[k*n+p,1*n+r]^.xp; y[2]:=rp[k*n+p,1*n+r]^.yp;
    z[2]:=rp[k*n+p,1*n+r]^.zp;
    x[3]:=rp[k*n+p-1,1*n+r]^.xp; y[3]:=rp[k*n+p-i,1*n+r]^.yp;
    z[3]:=rp[k*n+p-1,1*n+r]A.zp;
    x[4]:=rp[k*n+p-1,1*n+r-1]^.xp; y[4]:=rp[k*n+p-1,1*n+r-i]^.yp;
    z[4]:=rp[k*n+p-1,1*n+r-1]^.zp;
    x[1]:=rp[k*n+p,1*n+r-1]^.xp; y[1]:=rp[k*n+p,1*n+r-1]^.yp;
    z[1]:=rp[k*n+p,1*n+r-1]^.zp;
    if r=1 then begin if p=1 then begin q1:=1; q2:=4; end
                                 else begin q1:=1; q2:=2; end;
                end
           else if p=1 then begin qi:=2; q2:=3; end
                       else begin q1:=2; q2:=2; end;
    for q:=q1 to q2 do begin
      vetit (x [q] , y [q] , z [q]) ; sk [q] . uj (xt, yt) ;
      if q=2 then z[q]:=(ap*x[q]+bp*y[q]+cp*z[q])*nv1/nv2;
    end;
    if r>1 then begin
        sk[1]:=hl[k*nv+1+1]^[(p-1)*n+r-1].c[2];
        sk[4]:=h1[k*nv+l+1]^[(p-1)*n+r-i].c[3];
        if p>1 then sk[3]:=hl[k*nv+1+1]^[(p-2)*n+r].c[2]; end
      else if p>1 then begin
        sk[3]:=hl[k*nv+1+1]^[(p-2)*n+r].c[2];
        sk[4]:=hl[k*nv+l+1]^[(p-2)*n+r].c[1]; end;
        hl[k*nv+l+1]^((p-i)*n+r].uj(sk,z[2]);
      if fr=0 then hl[k*nv+1+i]^[(p-1)*n+r].sr((p-i)*n+r-1);
      (kezdősorrend)
  end;
  procedure sor; {hálószemekből álló sor}
  begin for r:=1 to n do negyszog; end;
  procedure keret; {sorokból álló háló}
  begin for p:=1 to n do sor; end;
begin
  for k:=0 to nu-1 do
    for 1:=0 to nv-1 do keret;
end;

procedure kp_szem; {kontrollponthálószem}
var
  pt: array[1..41 of pont3;
  sk: p4;
  procedure kp_nsz;
  begin
    if (ztu=1) and (k=nu) then
      if (ztv=1) and (1=nv) then begin
           pt[1]:=a[0,nv-1]; pt[2]:=a[0,0]; pt[3]:=a[nu-1,0];
           pt[4]:=a[nu-1,nv-1];
      end else begin
           pt[1]:=a[0,1-1]; pt[2]:=a[0,1]; pt[3]:=a[nu-1,1];
           pt[4]:=a[nu-i,l-1];
      end
    else if (ztv=1) and (1=nv) then begin
```

```
      pt[1]:=a[k,nv-1]; pt[2]:=a[k,0]; pt[3]:=a[k-1,0];
      pt[4]:=a[k-1,nv-1];
      end else begin
      pt[1]:=a[k,1-1]; pt[2]:=a[k,1]; pt[3]:=a[k-i,1];
      pt[4]:=a[k-1,1-1];
      end;
      for q:=1 to 4 do begin
        vetit(pt[q].xp,pt[q].yp,pt[q].zp); sk[q].uj(xt,yt);
        pt[q].zp:=ap*pt[q].xp+bp*pt[q]•yp+cp*pt[q].zp;
      end;
      kp_sz[(k-1)*nv+1].uj(sk,bs*(pt[1].zp+pt[2].zp+pt[3].zp)+
                           pt[4].zp);
      kp sz[(k-i)*nv+1].sr((k-1)*nv+1); (kezdősorrend}
  end;
begin
  for k:=1 to nu do for 1:=1 to nv do kp_nsz;
end;

procedure sorrend_h; {kontrollpontszemek rendezése}
var sf: byte;
     fg: integer;
begin
  repeat
     sf:=0;
     for k:=1 to nu do for 1:=1 to nv do
     for kp:=1 to nu do for 1p:=1 to nv do
       if (kp sz[(k-1)*nv+1].zm−kp_sz[(kp-1)*nv+lp].zm)*
          (kp sz[(k-1)*nv+1].n−kp_sz[(kp-1)*nv+lp].n)<0 then begin
           fg =kp_sz[(k-1)*nv+l].n; kp_sz[(k-1)*nv+i].sr(kp sz[(kp-1)
                  *nv+lp].n);
         kp_sz[(kp-1)*nv+lp].sr(fg); sf:=1; end;
  until sf=0;
end;

procedure sorrend_sz; {hálószemek rendezése egy kontrollpontszemen
                       belül)
var sf: byte;
     fg: integer;
begin
  repeat
     sf:=0;
     for km:=1 to n*n do for lm:=1 to n*n do
       if (hl[(k-1)*nv+1]^[km].zm-hl[(k-1)*nv+1)^[lm].zm)*
          (hl[(k-1)*nv+1]^[km].n-hl[(k-1)*nv+1]^[lm].n)<0 then begin
        fg:=h1[(k-1)*nv+1]^[km].n;
        hl[(k-1)*nv+1]^[km].sr(h1[(k-1)*nv+1]^[lm].n);
        hl[(k-1)*nv+1]^[lm].sr(fg); sf:=1; end;
  until sf=0;
end;

function konkav(k,l: byte): boolean;
     {megállapítja, hogy a kontrollpontszem vetülete konkáv-
e} var x,y: array[1..4] of real;
```

```
      kv1, kv2: real;
  begin
    if kq=1 then begin
      with kpsz[(k-1)*nv+l] do for q:=1 to 4 do begin
          x[q]:=c[q].xp; y[q]:=c[q].yp; end;
      kv1:=(y[2] * (x[3]-x[1]) -x[2] * (y[3]-y[1])+x[1] *y[3]-x[3] *y[i]) *
            (y[4]*(x[3]-x[1])-x[4]*(y[3]-y[i])+x[i]*y[3]-x[3]*y[i]);
      kv2:=(y[i] * (x[4]-x[2])-x[i] * (y[4]-y[2])+x[2] *y[4]-x[4] *y[2]) *
            (y[3] * (x[4]-x[2])-x[3] * (y[4]-y[2])+x[2] *y[4]-x[4] *y[2])
      ; if (kv1>0) or (kv2>0) then konkav:=true else konkav:=false;
    end else konkav:=true;
  end;

begin
  if fl*fp=0 then rajzol;
  case qj of
  'T':begin ("tele" (átlátszatlan) megjelenítés)
      setfillstyle(1,0);
      if fp=0 then begin
        kp_szem; sorrend_h;
      end;
      for lp:=1 to nu*nv do
        for kp:=1 to nu*nv do
          if lp=kp_sz[kp].n then begin
            l:=kp mod nv; if l=0 then l:=nv; k:=round((kp-1)/nv)+1;
            if konkav(k,l)=true then begin sorrend_sz;
              for lm:=1 to n*n do
                for km:=1 to n*n do
                  if lm=hl[(k-i)*nv+l]A[km].n+i then
                    fillpoly(4, hl[(k-1)*nv+1]^[km]); end
              else for km:=1 to n*n do fillpoly(4,
                                          hl[(k-1)*nv+l]^[km]);
            end;
      fp:=i;
    end;
  'D':begin (drótvázas megjelenítés)
      for k:=1 to nu do
        for l:=1 to nv do
          for km:=1 to n*n do begin
            for t:=1 to 4 do dh[t]:=h1[(k-1)*nv+1]A[km].c[t];
            dh[5]:=dh[1];
            drawpoly(5, dh);
        end;
    end;
  end;
end;

end.
```

Ezt a módszert használva rajzolja a CSAV_GY.PAS nevű program a *3.1. ábrán* látható "csavart gyűrűt", amelynek paraméteres előállítása egy, a tórusz

meridiánsíkjában fekvő leírókörnek középpontját a tórusz felületén futó csavarvonal mentén mozgatva nyerhető.

A leírókör középpontjának pályagörbéjét hordozó tórusz paraméteres előállítása:

ahol R a tórusz középkörének (a leírókör középpontja által leírt kör) sugara, r a leírókör sugara. A paraméterek közötti összefüggés egy görbét értelmez a tóruszon. Az ábrázolt esetben $k = 6$. Legyen ez a görbe a felület keresett felület középköre. A leírókörének sugara q.

$$\mathbf{r}(\varphi,\psi) = \big((R + r\sin\varphi)\cos\psi, (R + r\sin\varphi)\sin\psi, r\cos\varphi\big) \quad \varphi \in [0,2\pi]\ ; \quad \psi \in [0,2\pi] \tag{3.7}$$

$$\varphi = k\psi \quad ; \quad k \in N \tag{3.8}$$

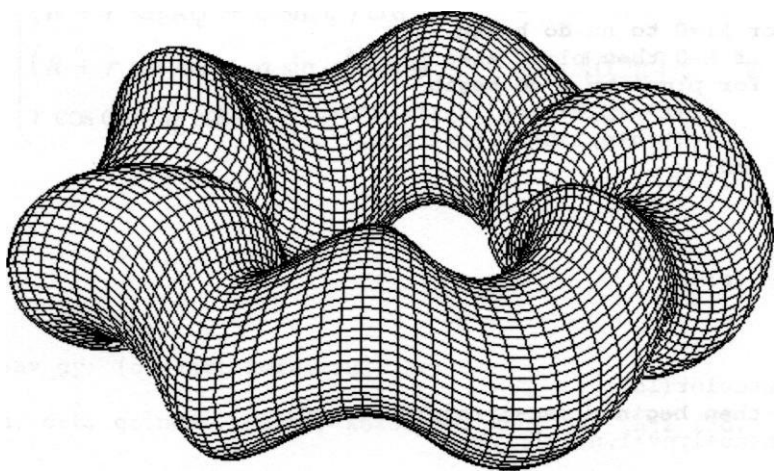
Akkor a keresett előállítás:

$$\mathbf{r}(\varphi,\psi) = \begin{pmatrix} (R + r\sin k\psi + q\sin\varphi)\cos\psi, \\ (R + r\sin k\psi + q\sin\varphi)\sin\psi, \\ r\cos\varphi + q\cos\varphi \end{pmatrix} \quad \varphi \in [0,2\pi]\ ; \quad \psi \in [0,2\pi] \tag{3.9}$$

```
{$N+}

program csau gy; (csavart töruszfelület)

uses graph, crt, grindhi, kozos, kozos 3d, konf 3d, rajz f3d;

const
  nu=5; nv=2; n=12;
  (nu=7; nv=3; n=16;) (így csak a fejlesztési környezeten (IDE)
              kívül fut, különben túlcsodul a heap) ru=160;
              ry=20; rg=60; t=6; (méretek)
  valasz f: kars = ('D','T', fel, le, jobb, bal, char(27));
```

```
var
 fi, psi, om: single;
 pim, pin: single;

procedure raj z; { felületmegjelení tés}
                                          (paraméteres előállítás)
function fx(k,l: integer): real;
 begin fx:=(ru+ry*sin(t*k*pim)+rg*sin(l*pin))*cos(k*pim); end;

 function fy(k,l: integer): real;
 begin fy:=(ru+ry*sin(t*k*pim)+rg*sin(l*pin))*sin(k*pim); end;

 function fz(k,l: integer): real;
 begin fz:=ry*cos(t*k*pim)+rg*cos(l*pin); end;

procedure d_racs; (osztórács csomópontjai}
 begin
    for 1:=0 to nv do
     for k:=0 to nu do
       a[k,1].uj(fx(k*n,l*n),fy(k*n,l*n),fz(k*n,l*n));
 end;

 procedure rajzracs; {felületháló csomópontjainak kiszámítása}
 var rl, pl: integer;
 begin
    for 1:=0 to nv do begin
     if 1=0 then r1:=0 else r1:=1;
     for r:=r1 to n do begin
       for k:=0 to nu do begin
         if k=0 then pl:=0 else pl:=1;
         for p:=p1 to n do begin
            rp[k*n+p, l*n+r) .uj (fx (k*n+p, 1*n+r) ,
            fy(k*n+p,l*n+r),fz(k*n+p,1*n+r));
         end;
       end;
     end;
    end;
    fl:=1; fp:=0;
 end;

begin
 nyil; setcolor(15);
 if f1=0 then begin d_racs; rajzracs; end;
 megjelen(nu+1,nv+1,n,0,1,1,1);
 fr:=1;
end;

procedure fg; (vetítési irány váltortatásának vezérlése)
begin
 repeat
    setcolor(7);
    case qj of
    'D':outtextxy(15,102,'T - tele');
```

```
      'T':outtextxy(15,102,'D - drótváz');
      end;
      repeat kr:=upcase(readkey); until kr in valasz f;
      cleardevice;
      case kr of
      'D','T': qj=kr;
      else if kr>chr(27) then begin forgat; fp:=0; end;
      end;
      if kr<>chr(27) then rajz;
  until kr=chr(27);
end;

begin
  pim:=2*pi/(nu+1)/n; pin:=2*pi/(nv+1)/n;
  ap:=0.3; bp:=0.4; cp:=0.2; {kezdeti vetítési irány}
  nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
  fl:=0; fp:=0; fr:=0;
  grafind;
  for k:=0 to (nu+1)*n do
     for 1:=0 to (nv+1)*n do new(rp(k,1]);
  for k:=1 to (nu+1)*(nv+1) do new(hl[k]);
  qj =' D' ;
  rajz; fg;
  closegraph;
end.
```



3.1. ábra

3.2. Felületek illesztése

Bonyolultabb felületek szerkesztésekor gyakran adódik az illesztés problémája, ugyanis egyszerűbb felületrészleteket külön-külön megszerkesztünk, majd belőlük összeállítjuk a keresett objektumot. Interaktív szerkesztésnél a kontrollpontokat összekötő görbék a felületet foltokra (angolul patch) osztják, amelyek pontosan az említett görbék mentén illeszkednek.

Ha a kontrollpontrács a paraméter értelmezési tartományának téglalapokra való felosztásának felel meg, akkor egy patch előállítása:

$$\mathbf{r}(u, v) = \mathbf{r}\big(x(u, v),\ y(u, v),\ z(u, v)\big)\ ; \qquad (u, v) \in [a, b] \times [c, d]. \quad (3.10$$

Két szomszédos patch 0-ad rendű folytonosan illeszkedik, ha a közös határgörbe azonos, például:

$$\begin{aligned} &\mathbf{r}_1 : [a, b] \times [c, d] \to \Re^3\ ;\ \mathbf{r}_2 : [b, e] \times [c, d] \to \Re^3 \\ &\mathbf{r}_1(b, v) = \mathbf{r}_2(b, v)\ ; \qquad v \in [c, d] \end{aligned} \quad . \qquad (3.11$$

Az illesztés elsőrendben folytonos, ha a két patch érintővektorai a határgörbe minden pontjában egyenlőek. A fenti példát tekintve, a v irányú érintők önműködően egyenlőek, hiszen a közös határgörbe érintői. Még az *u* irányú érintők egyenlőségét kell kikötni:

$$\begin{aligned} &\mathbf{r}_1 : [a, b] \times [c, d] \to \Re^3\ ;\ \mathbf{r}_2 : [b, e] \times [c, d] \to \Re^3 \\ &\frac{\partial \mathbf{r}_1(u, v)}{\partial u} = \left.\frac{\partial \mathbf{r}_2(u, v)}{\partial u}\right|_{u=b}\ ; \qquad v \in [c, d] \end{aligned} \quad . \qquad \mathbf{(3.12)}$$

Ez azt jelenti, hogy az érintősík folytonos változású lesz a határgörbe mentén. A gyakorlatban egy ennél gyengébb feltételt is alkalmaznak, éspedig azt, hogy az érintősík egyáltalán létezzen a határgörbe mentén. Ehhez elég, ha a 3.12 összefüggésben szereplő érintővektorok kollineárisak, nem kötelező hogy egyenlőek legyenek.

3.3. Sima felületek interaktív szerkesztése

A továbbiakban a kontrollpontrács segítségével történő felületmodellezési eljárásokkal foglalkozunk. A görbeszerkesztéshez hasonlóan a kontrollpontok helyvektorainak súlyozott közepét jelentő előállítást használunk, a paramétertartomány egységnyi oldalú négyzetét feleltetve meg a kontrollpontrács egy-egy szemének:

$$\mathbf{r}(u,v) = \sum_{k=0}^{m}\sum_{l=0}^{n} f_k(u) f_l(v) \mathbf{p}_{kl} \; ; \qquad (u,v) \in [0,1] \times [0,1] \; . \tag{3.13}$$

A súlyfüggvények ezúttal is polinomok lesznek. A harmadfokúnál nem nagyobb fokszámú polinomok a símaság szempontjából megfelőek, előállításuk sem különösen bonyolult. Magasabb fokú polinomok használata igencsak megnöveli a szükséges műveletek mennyiségét és bonyolítja az együtthatók meghatározását.

Az interpolációs módszerek pontosan a kontrollpontok közé "feszítenek ki" egy-egy patch-et, az approximációs eljárások esetén azonban a felület nem megy át a kontrollpontokon.

3.3.1. Hermite interpoláció

A patch négy sarkában megadva a paramétervonalak érintőit, ezek a görbék Coons-Hermite interpolációval előállíthatók, megadva a patch határgörbéit:

$$\begin{aligned}
\mathbf{r}(0,v) &= f_1(v)\mathbf{p}_{00} + f_2(v)\mathbf{p}_{01} + f_3(v)\mathbf{p}^{v}_{00} + f_4(v)\mathbf{p}^{v}_{01} \\
\mathbf{r}(1,v) &= f_1(v)\mathbf{p}_{10} + f_2(v)\mathbf{p}_{11} + f_3(v)\mathbf{p}^{v}_{10} + f_4(v)\mathbf{p}^{v}_{11} \\
\mathbf{r}(u,0) &= f_1(u)\mathbf{p}_{00} + f_2(u)\mathbf{p}_{10} + f_3(u)\mathbf{p}^{u}_{00} + f_4(u)\mathbf{p}^{u}_{10} \\
\mathbf{r}(u,1) &= f_1(u)\mathbf{p}_{01} + f_2(u)\mathbf{p}_{11} + f_3(u)\mathbf{p}^{u}_{01} + f_4(u)\mathbf{p}^{u}_{11}
\end{aligned} \; . \tag{3.14}$$

A felületfolt meghatározásához szükség van azonban a határgörbék pontjaiban a másik irányú érintőkre. Ezeket akkor tudjuk meghatározni, ha ismertek a csúcspontokban a másodrendű vegyes parciális deriváltak, az úgynevezett twist- (csavaró) vektorok:

$$\mathbf{r}^{uv} = \frac{\partial^2 \mathbf{r}}{\partial v \partial u}(u,v) \quad . \tag{3.15}$$

Az alkalmazott függvények (algebrai polinomok) folytonossági és deriválhatósági feltételei mellett a vegyes parciális deriváltak a deriválási sorrendtől függetlenek.

Ezzel meghatározhatóak a keresett

$$\begin{aligned}
\mathbf{r}^{u}(0,v) &= f_1(v)\mathbf{p}^{u}_{00} + f_2(v)\mathbf{p}^{u}_{01} + f_3(v)\mathbf{p}^{uv}_{00} + f_4(v)\mathbf{p}^{uv}_{01} \\
\mathbf{r}^{u}(1,v) &= f_1(v)\mathbf{p}^{u}_{10} + f_2(v)\mathbf{p}^{u}_{11} + f_3(v)\mathbf{p}^{uv}_{10} + f_4(v)\mathbf{p}^{uv}_{11} \\
\mathbf{r}^{v}(u,0) &= f_1(u)\mathbf{p}^{v}_{00} + f_2(u)\mathbf{p}^{v}_{10} + f_3(u)\mathbf{p}^{uv}_{00} + f_4(u)\mathbf{p}^{uv}_{10} \\
\mathbf{r}^{v}(u,1) &= f_1(u)\mathbf{p}^{v}_{01} + f_2(u)\mathbf{p}^{v}_{11} + f_3(u)\mathbf{p}^{uv}_{01} + f_4(u)\mathbf{p}^{uv}_{11}
\end{aligned} \tag{3.16}$$

A felület meghatározása tehát a határfeltételmátrix

$$\mathbf{A} = \begin{bmatrix}
\mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}^{v}_{00} & \mathbf{p}^{v}_{01} \\
\mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}^{v}_{10} & \mathbf{p}^{v}_{11} \\
\mathbf{p}^{u}_{00} & \mathbf{p}^{u}_{01} & \mathbf{p}^{uv}_{00} & \mathbf{p}^{uv}_{01} \\
\mathbf{p}^{u}_{10} & \mathbf{p}^{u}_{11} & \mathbf{p}^{uv}_{10} & \mathbf{p}^{uv}_{11}
\end{bmatrix} \tag{3.17}$$

a következőképen állítható elő:

$$\begin{aligned}
\mathbf{r}(u,v) &= \left[f_1(u), f_2(u), f_3(u), f_4(u)\right]\mathbf{A}\left[f_1(v), f_2(v), f_3(v), f_4(v)\right]^T = \\
&= \sum_{k=1}^{4}\sum_{l=1}^{4} f_k(u) f_l(v)\mathbf{a}_{kl} \; ; \qquad (u,v) \in [0,1] \times [0,1]
\end{aligned} \tag{3.18}$$

A F_P_HERM.PAS nevű program ezt a szerkesztést könnyíti meg. Interaktív módon, a program futása közben lehet változtatni a határfeltételeket. A határgörbét és az érintővektorokat (kisebb léptékkel) a program a szerkesztés közben is megjeleníti.

```
{$N+}

program f_p_herm; {felületfolt (patch) Hermite-interpolációval}

uses
  graph, grafind, crt, dos, kozos, kozos_3d, kerdesek, rajz_f3d,
  konf_3d, komm_fel, eger_kez, gomb_kez;

const
  pu=1; pv=1;
  nu=3; nv=3;
  ax=300; ay=300; n=16;
  valasz: kars = ['E','F','Q','V'];
  vsz: kars = ['H','T','U','V',chr(27)];
  valasz_f: kars = ['D','T', fel, le, jobb, bal, char(27)];

var qv: char;

procedure rajz(kep: byte); {felületmegjelenítés}
var
  u, u2, u3, v, v2, v3: single;
  fu, fv: array[1..4] of single;
  gp: array[1..2,0..3,0..n] of kpont;

  procedure sulyfugg; {súlyfüggvények}
  begin
    u:=p/n; u2:=u*u; u3:=u2*u;
    v:=r/n; v2:=v*v; v3:=v2*v;
    fu[1]:=2*u3-3*u2+1; fu[2]:=-2*u3+3*u2; fu[3]:=u3-2*u2+u;
    fu[4]:=u3-u2;
    fv[1]:=2*v3-3*v2+1; fv[2]:=-2*v3+3*v2; fv[3]:=v3-2*v2+v;
    fv[4]:=v3-v2;
  end;

  function xx: single;
  var s: single;
  begin
    s:=0;
    for i:=1 to 4 do
      for j:=1 to 4 do
        s:=s+a[i-1,j-1].xp*fu[i]*fv[j];
    XX:=s
  end;

  function yy: single;
  var s: single;
  begin
    s:=0;
    for i:=1 to 4 do
      for j:=1 to 4 do
        s:=s+a[i-1,j-1].yp*fu[i]*fv[j];
    yy:=s
  end;
```

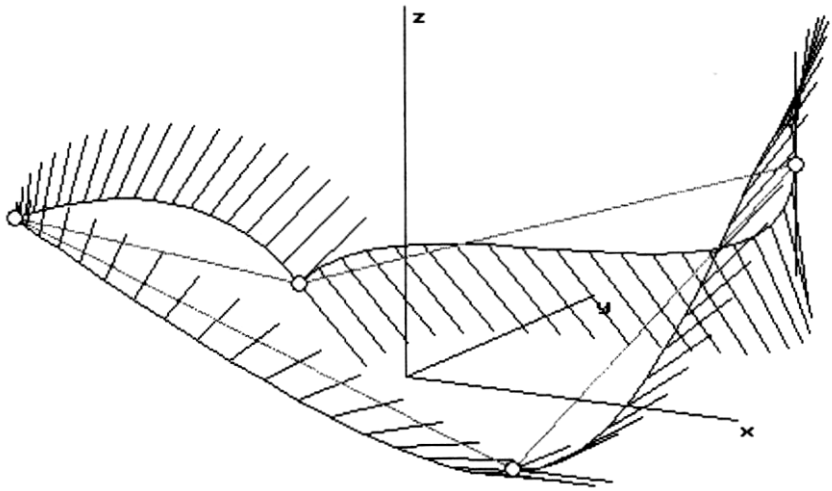
```
function zz: single;
var s: single;
begin
  s:=0;
  for is=1 to 4 do
    for j:=1 to 4 do
       s:=s+a[i-l,1-1].zp*fu[i]*fv[j];
  zz:=s
end;

procedure gorbek; {szegélygörbék és érintők
megjelenítése} const el=0.2; {érintőmegjelenítési lépték}
var g: byte; s, gu, gv: pont3;

procedure   gbu(q:   byte;   var   gu:   pont3);   {u   irányú
szegélygörbék} begin
  s.uj (0,0,0);
  for i:=1 to 4 do
    s.uj(s.xp+a[i-1,g+2*q].xp*fu[i],s.yp+a[i-
         1,g+2*q].yp*fu[i], s.zp+a[i-1,g+2*q].zp*fu[i]);
  gu:=s;
end;

procedure gbv(q: byte; var gv: pont3); {v irányú szegélygörbék}
begin
  s.uj (0,0,0);
  for i:=1 to 4 do
    s.uj(s.xp+a[g+2*q,i-1].xp*fv[i],s.yp+a[g+2*q,i-
         1].yp*fv[i], s.zp+a[g+2*q,i-1].zp*fv[i]);
  gv:=s;
end;

begin
  for p:=0 to n do begin
    r:=p;
    sulyfugg;
    for q:=0 to 1 do
       for g:=0 to 1 do begin
          gbu(q,gp[q+l,g,p]); gbv(q,gp[q+l,g+2,p]);
       end;
    for g:=0 to 3 do begin
       with gp[2,g,p] do
           uj(gp[1,g,p].xp+xp*el,gp[1,g,p].yp+yp*el'gp[l,g,p].zp+zp*el);
       gp[i,g,p].koord; gp[2,g,p].koord;
       line(gp[l,g,p] .xr,gp[l,g,p].yr,gp[2,g,p] .xr,
            gp[2,g,p].yr); {érintők}
    end;
  end;
  for g:=0 to 3 do {szegélygörbék meghúzása}
    for p:=0 to n-1 do
       line(gp[l,g,p] .xr,gp[l,g,p] .yr,gp[l,g,p+1] .xr,gp[l,g,p+1] .yr);
end;
```

```
  procedure rajzracs; {felületháló csomópontjainak kiszámítása}
  begin
     for p:=0 to n do
      for r:=0 to n do begin
         sulyfugg;
         rp [p, r] ^.uj (xx, yy, zz) ;
      end;
     fl:=1; fp:=0;
  end;

begin
  racs;
  case kep of
  1:begin gorbek; kpontok(pu,pv,0,0); end;
  2:begin
      if gb[2]=1 then nyil; setcolor(15);
      if f1=0 then rajzracs;
      eger_n; megjelen(pu,pv,n,0,0,0,1); eger_l;
      fr:=1;
     end;
  end;
end;

procedure fg; {vetítési irány változtatásának vezérlése}
begin
  repeat
     setcolor(7);
     case qj of
     'D':outtextxy(15,102,'T - tele');
     'T':outtextxy(15,102,'D - drótváz');
     end;
     repeat kr:=upcase(readkey); until kr in valasz_f;
     cleardevice;
     case kr of
     'D','T': qj:=kr;
     else if kr<>chr(27) then begin forgat; fp:=0; end;
     end;
     if kr<>chr(27) then rajz(2);
  until kr=chr(27);
end;

procedure v_v; {módosítandó mennyiség (kontrollpont helye - u/v
                irányú érintő - twistvektor) kiválasztása}
var n: byte;
begin
  qv:='';
  lap(208,5); setcolor(0); outtextxy(290,6,'Mi változik
  ?'); outtextxy(310,18,'.rintö');
  ablak(220,15,280,28); ablak(375,15,435,28);
  ablak(270,31,330,44); ablak(335,31,395,44); setcolor(12);
  outtextxy(225,18,'H'); outtextxy(380,18,'T');
  outtextxy(275,34,'U'); outtextxy(340,34,'V'); setcolor(15);
  outtextxy(233,18,'ely'); outtextxy(388,18,'wist');
```

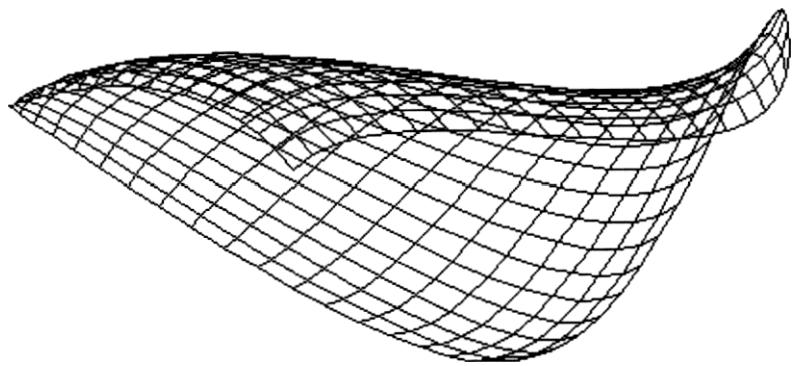
```
  outtextxy(280,34,' irány'); outtextxy(345,34,' irány');
  eger_t(208,5,448,45); eger_1; eger_h(328,10);
  repeat
    with reg do begin
       ax:=3; intr($33,reg);
       if bx=1 then begin
        katt(220,15,280,28,n); if n=1 then qv:='H';
        katt(270,31,330,44,n); if n=1 then qv:='U';
        katt(335,31,395,44,n); if n=1 then qv:='V';
        katt(375,15,435,28,n); if n=1 then qv:='T';
        bx:=0; eger_h(328,10);
       end;
    end;
    if keypressed then qv:=upcase(readkey);
  until qv in vsz;
  eger_n;
end;

begin
  grindhi; keret(kx0, ky0); kp_racs(pu,pv); eger_k;
  for k:=0 to pu*n do
    for 1:=0 to pv*n do new(rp[k,l]);
  for k:=1 to pu*pv do new(hl[k]);
  repeat
    qu:='N';
    ap:=0.4; bp:=-0.7; cp:=0.2;
    fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nu do for 1:=0 to nv do a[k,l].uj(0,0,0);
    for k:=0 to pu do for 1:=0 to pv do a[k,l].uj((k-pu/2)*ax,
                                          (1-pv/2)*ay,0);
    qk:='N';
    racs; konfig('FHC'); eger_n; parancs(2); rajz(1);
    repeat
       eger_t(kx0,2,620,ky0); eger_1; qr:=' '; b:=0;
       repeat par_katt(2,qr); kp_r_katt(pu,pv); until qr in valasz;
       eger_n;
       case qr of
        'V':begin {kontrollpont kiválasztása adatainak módosítása}
           gb[3]:=0; fe:=0; rajz(1);
           gb[1]:=1; parancs(2); if b=0 then sorszam(pu,pv);
           kp_kor(szx,szy,12); v_v;
           case qv of
           'H':begin
             inform(szx,szy,'    Hely: '); koordin3('Hely');
           if qr<>chr (27) then a [szx, szy] . uj (x, y, z) ;
           end; 'U':begin
             inform(szx+2,szy,' Érintö-u: '); koordin3('Érintö-
           u'); if qr<>chr(27) then a[szx+2,szy].uj(x,y,z); end;
           'V':begin
             inform(szx,szy+2,' Érintö-v: '); koordin3('Érintö-
             v'); if qr<>chr(27) then a[szx,szy+2].uj(x,y,z); end;
```

```
            'T':begin
              inform(szx+2,szy+2,'  Twist: '); koordin3('Twist');
              if qr<>chr(27) then a[szx+2,szy+2].uj(x,y,z); end;
            end;
            setfillstyle(1,11); bar(130,470,590,480);
            kp_kor(szx,szy,7); rajz(1);
            gb[1]:=0; lap(208,5);
            fl:=0;
          end;
        'F':begin (felületmegjelenítés)
            gb[3]:=1; parancs(2); fe:=1;
            megj; rajz(2);
            gb[3]:=0;
          end;
        'E':if fe=1 then begin (felületmegjelenítés keret nélkül)
            gb[2]:=1; parancs(2);
            megj; cleardevice;
            rajz(2); fg; gb[2] :=0;
            keret(kx0, ky0); kp_racs(pu,pv);
            megj; rajz(2);
          end;
        'Q':begin {kilépés}
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('FHC') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin
              gb[3] :=0; rajz(1); end;
            gb[4]:=0;
          end;
        end;
        parancs(2);
      until (qk='I') or (qu='I');
      gb[3]:=0; parancs(2);
    until qu='N';
    closeqraph;
end.
```

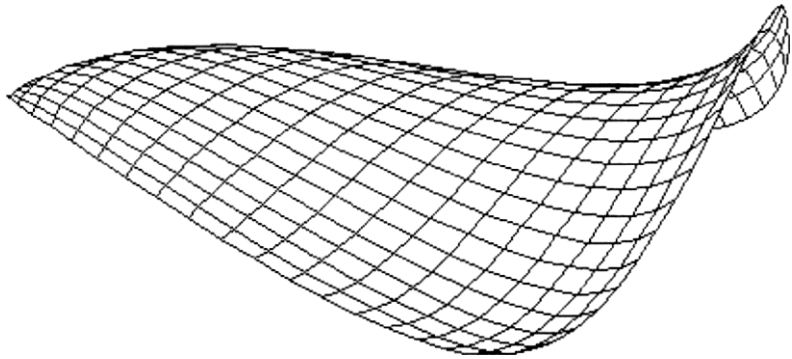
A *3.2. ábrán* egy, ezzel a programmal meghatározott határfeltételeknek megfelelő határgörbék és ezek menti érintők, a *3.3. ábrán* a megfelelő Hermitepatch drótvázas, a *3.4. ábrán* pedig ennek képies megjelenítése látható.



3.2. ábra



3.3. ábra



3.4. ábra

A Hermite-patch-ek illesztésének feltétele, hogy a közös csúcsokhoz tartozó határfeltételek (érintő- és twist-vektorok) egyenlőek legyenek.

33.2. Spline interpoláció

A kontrollpontoknak az értelmezési tartomány ugyancsak egységnyi oldalhosszúságú négyzetes felosztásának csomópontjait feleltetve meg, a térgörbék spline-interpolációjánál (2.3.4. *pont)* használt módszert alkalmazva, a következő módon állítható elő a

$$\begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \cdots & \mathbf{p}_{0n} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \cdots & \mathbf{p}_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{p}_{m0} & \mathbf{p}_{m1} & \cdots & \mathbf{p}_{mn} \end{bmatrix} \qquad (3.19$$

kontrollpontokra illeszkedő spline-felület:

- megszerkesztjük a

$$\begin{bmatrix} \mathbf{p}_{k0} & \mathbf{p}_{k1} & \cdots & \mathbf{p}_{kn} \end{bmatrix} ; \quad k = 0,1,\ldots,m \qquad (3.20$$

  kontrollpontokra illeszkedő spline-görbéket
  a kapott görbéken felveszünk egy, a megjelenítéshez szükséges sűrűségű paraméter-felosztásnak megfelelő pontsort:

$$\mathbf{r}_p\left[k \cdot s, l \cdot s + q\right] ; q = 0,1,\ldots,s ; l = 0,1,\ldots,n$$
$$k = 0,1,\ldots,m \qquad (3.2$$

  ahol *s* a két szomszédos kontrollpont közötti ívet felosztó pontok száma
- a kapott pontsorok azonos $l \cdot s + q$ oszlopszámának megfelelő pontokat kontrollpontoknak tekintve megszerkesztjük az ezekre illeszkedő spline-görbéket, ezzel kitöltve a megjelenítéshez szükséges háló csomópontjainak helyvektorait tartalmazó

$$\mathbf{r}_p\left[k \cdot s + p, l \cdot s + q\right]$$
$$q = 0,1,\ldots,s ; l = 0,1,\ldots,n \qquad (3.2$$
$$p = 0,1,\ldots,s ; k = 0,1,\ldots,m$$

mátrixot

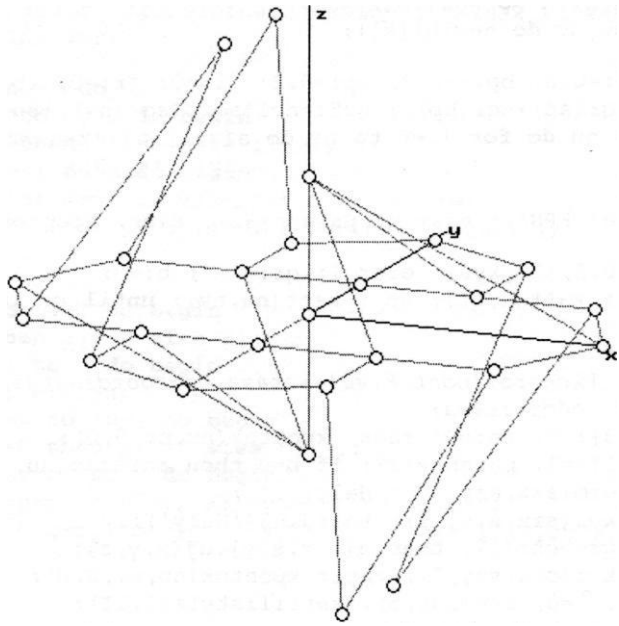
A F_P_SPLN.PAS program segítségével készült *a 3.5 ábrán* látható karakterisztikus "poliéderre" illeszkedő, a *3.6 ábrán* látható spline-felület. Megjegyzendő, hogy az itt poliédernek nevezett alakzat lapjai általában nem síkidomok, hanem torz négyszögek.

```
{$N+}

program f_p_spin;
    {paraméteres előállítású nyílt felületek spline interpolációval)

uses
  graph, grafind, crt, dos, kozos, kozos_3d, kerdesek, rajz_f3d,
  konf_3d, komm_fel, eger_kez, gomb_kez;

const
  nu=4; nv=4; ax=80; ay=80; n=12;
  valasz: kars = ['E','F','Q','V'];
  valasz f: kars = ['D','T', fel, le, jobb, bal, char(27)];

type
  mu = array[0..nu] of single;
  mv = array[0..nv] of single;

var
  gx, gy, gz: mu;
  hu, ku: array[l..nu-1] of single;
  fx, fy, fz: mv;
  hv, kv: array[l..nv-1] of single;

procedure splineu(var g: mu); {spline együtthatók u irányban}
begin
  ku[1]:=1/4; for i:=2 to nu-1 do ku[i]:=1/(4-ku[i-1]);
  g[0] :=0; g[nu] :=0; hu[1] :=hu[1] /4;
  for i:=2 to nu-1 do hu[i]:=ku[i]*(hu[i]-hu[i-1]); g[nu-1]:=hu[nu-1];
  for i:=nu-2 downto 1 do g[i]:=hu[i]-ku[i]*g[i+l];
end;

procedure splinev(var f: mv); {spline együtthatók v irányban}
begin
  kv[1]:=1/4; for j:=2 to nv-1 do kv[j]:=1/(4-kv[j-l]);
  f[0]:=0;  f [nv] :=0; hv[1] :=hv[1] /4;
  for j:=2 to nv-1 do hv[j]:=kv[j]*(hv[j]-hv[j-1]); f[nv-1]:=hv[nv-1];
  for j:=nv-2 downto 1 do f[j]:=hv[j]-kv[j]*f[j+1];
end;

procedure rajz; {felületmegjelenítés}
var
  u, w, u3, w3, v, t, v3, t3: single;
```

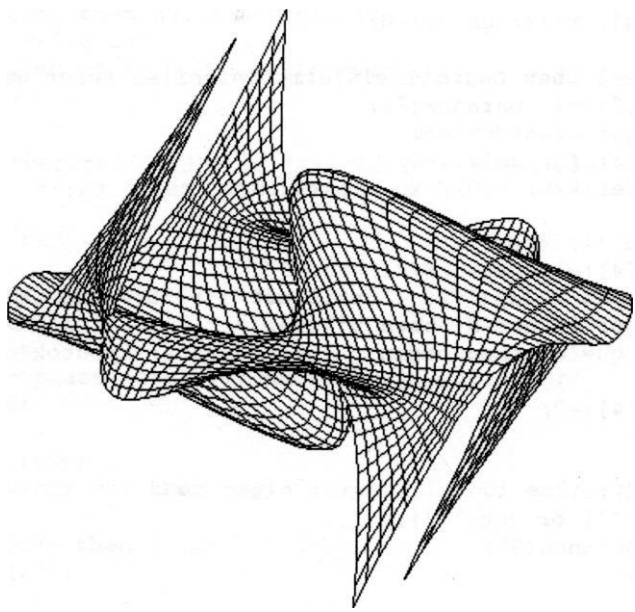
```
function xu(k,j,r: integer): single;
begin
  for is=1 to nu-1 do
    hu[i]:=rp[(i-1)*n,j*n+r]^.xp-2*rp[i*n,j*n+r]^.xp+
           rp[(i+1)*n,j*n+r]^.xp;
  splineu(gx);
  xu:=gx[k]*t3+gx[k+l]*v3+(rp[k*n,j*n+r]A.xp-gx[k])*t+
       (rp[(k+l)*n,j*n+r]^.xp-gx[k+l])*v;
end;
function xv(i,l: integer): single;
begin
  for j:=1 to nv-1 do hv[j]:=a[i,j-1].xp-2*a[i,j].xp+a[i,j+l].xp;
  splinev(fx);
  xv:=fx[1]*w3+fx[l+l]*u3+(a[i,l].xp-fx[1])*w+(a[i,l+l].xp-
       fx [ l+l ]) *u;
end;

function yu(k,j,r: integer): single;
begin
  for is=1 to nu-1 do
    hu[i]:=rp[(i-1)*n,j*n+r]A.yp-2*rp[i*n,j*n+r]^.yp+
           rp[(i+l)*n,j*n+r]A.yp;
  splineu(gy);
  yu:=gy[k]*t3+gy[k+l]*v3+(rp[k*n,j*n+r]A.yp-gy[k])*t+
       (rp[(k+l)*n,j*n+r]A.yp-gy[k+l])*v;
end;
function yv(i,l: integer): single;
begin
  for j:=1 to nv-1 do hv[j]:=a[i,j-1].yp-2*a[i,j].yp+a[i,j+l].yp;
  splinev (fy) ;
  yv:=fy[1]*w3+fy[l+l]*u3+(a[i,l].yp-fy[1])*w+(a[i,l+l].yp-
       fy[1+1])*u;
end;

function zu(k,j,r: integer): single;
begin
  for is=1 to nu-1 do
    hu[i]:=rp[(i-1)*n,j*n+r]^.zp-2*rp[i*n,j*n+r]^.zp+
           rp[(i+l)*n,j*n+r]^.zp;
  splineu(gz);
  zu:=gz[k]*t3+gz[k+l]*v3+(rp[k*n,j*n+r]^.zp-gz[k])*t+
       (rp[(k+l)*n,j*n+r]^.zp-gz[k+l])*v;
end;
function zv(i,l: integer): single;
begin
  for j:=1 to nv-1 do hv[j]:=a[i,j-1].zp-2*a[i,j].zp+a[i,j+l].zp;
  splinev (fz) ;
  zv:=fz[1]*w3+fz[1+1]*u3+(a[i,l].zp-fz[1])*w+(a[i,l+l].zp
       fz[1+1])*u;
end;
```

```
  procedure rajzracs; {felülethálo csomápontjainak kiszámítása}
  var rl, pl: integer;
  begin
     for k:=0 to nu do
       for 1:=0 to nv-1 do begin
        if 1=0 then r1:=0 else r1:=1;
        for r:=r1 to n do begin
           u:=r/n; w:=1-u; u3:=u*u*u; w3:=w*w*w;
           rp[k*n,l*n+r]^.uj (xv(k,1),yv(k,1),zv(k,1) );
        end;
       end;
     for 1:=0 to nv-1 do begin
       if 1=0 then r1:=0 else ri:=1;
       for r:=r1 to n do begin
        u:=r/n; w:=1-u;
        for k:=0 to nu-1 do begin
           if k=0 then pl:=0 else pi:=1;
           for p:=p1 to n do begin
             v:=p/n; t:=1-v; v3:=v*v*v; t3:=t*t*t;
             rp[k*n+p,1*n+r]^.uj (xu(k,l,r),yu(k,l,r),zu(k,l,r) );
           end;
        end;
       end;
     end;
     fl:=1; fp:=0;
  end;
begin
  racs; if gb[2]=1 then nyíl; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu,nv,n,1,0,0,1); eger_1;
  fr:=1;
end;
procedure fg; {vetítési irány változtatásának vezérlése}
begin
  repeat
     setcolor(7);
     case qj of
     'D':outtextxy(15,102,'T - tele');
     'T':outtextxy(15,102,'D - drótváz');
     end;
     repeat kr:=upcase(readkey); until kr in valasz_f;
     cleardevice;
     case kr of
     'D', ' T' : qj : =kr;
     else if kr<>chr(27) then begin forgat; fp:=0; end;
     end;
     if kr<>chr(27) then rajz;
  until kr=chr(27);
end;
begin
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to nun do
```

```
    for 1:=0 to nv*n do new(rp[k,l]);
  for k:=1 to nu*nv do new(hl[k]);
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nv1)+sqr(cp));
    for k:=0 to nu do for 1:=0 to nv do a[k,1].uj((k-nu/2)*ax,(1-
                                          nv/2) *ay, 0) ;
    qk:='N';
    racs; konfig('FPS'); eger_n; parancs(2); racs; kpontok(nu,nv,0,0);
    repeat
      eger_t(kx0,2,620,ky0); eger_l; qr:=' '; b:=0;
      repeat par_katt (2, qr) ; kp_r_katt (nu, nv) ; until qr in valasz;
      eger_n;
      case qr of
      'V':begin {kontrollpont kiválasztása és koordinátáinak
                 módosítása}
            gb[3]:=0; fe:=0; racs; kpontok(nu,nv,0,0);
            gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
            inform(szx,szy,' Hely: ');
            kp_kor(szx,szy,12); koordin3('Hely');
            if qr<>chr (27) then a [szx, szy] .uj (x, y, z) ;
            kp_kor(szx,szy,7); racs; kpontok(nu,nv,0,0);
            gb[1]:=0; lap(208,5); setfillstyle(1,11);
            bar(130,470,590,480);
            f1:=0;
          end;
      'F':begin {felületmegjelenítés}
            gb[3]:=1; parancs(2); fe:=1;
            megj; rajz; gb[3]:=0;
          end;
       'E':if fe=1 then begin {felületmegjelenítés keret nélkül}
            gb[2] :=1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb[2]:=0;
            keret (kx0, ky0) ; kp_racs (nu, nv) ; megj; rajz;
          end;
      'Q':begin (kilépés)
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('FPS') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs; kpontok(nu,nv,0,0);
                           end;
            gb[4]:=0;
          end;
      end;
      parancs(2);
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```



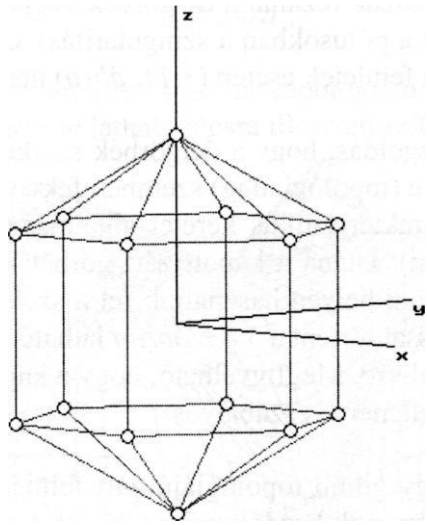
3.5. *ábra*



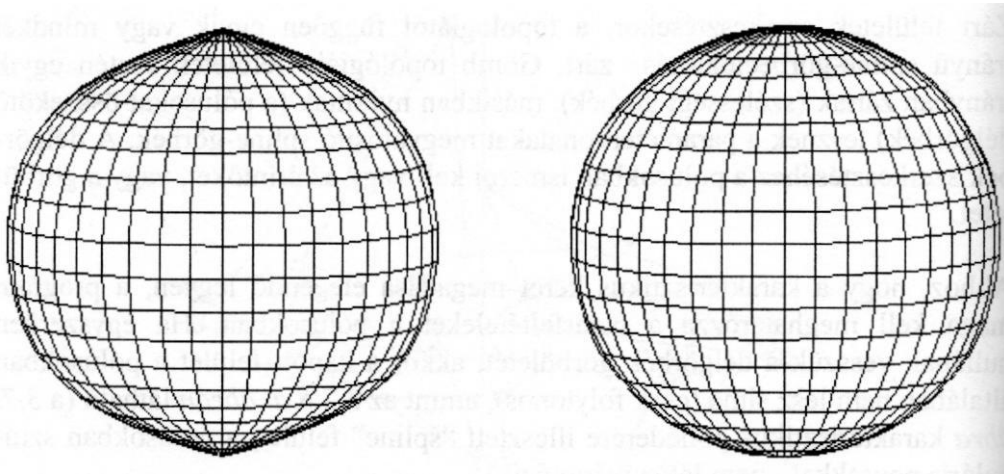
3.6. *ábra*

Zárt felületek szerkesztésekor, a topológiától függően egyik vagy mindkét irányú spline-görbeelőállítás zárt. Gömb topológiájú felületek esetén egyik irányban zártak (szélességi görbék), másikban nyitottak (a pólusokat összekötő délgörbék) lesznek a paramétervonalakat megvalósító spline-görbék. A délgörbék szerkesztéséhez a pólusokban ismerni kell vagy az érintőket, vagy a görbületet.

Ahhoz, hogy a karakterisztikus keret megadása elegendő legyen, a program maga kell meghatározza a határfeltételeket a pólusokban. Ha egyszerűen nullának vesszük a délgörbék görbületét, akkor a kapott felület a pólusokban általában nem lesz sima (csak folytonos), amint az a *3.8. a ábrán* látható (a 3.7. *ábra* karakterisztikus poliéderére illesztett "spline" felület, a pólusokban szinguláris pontokkal - nem létezik érintősík).



3.7. ábra



b

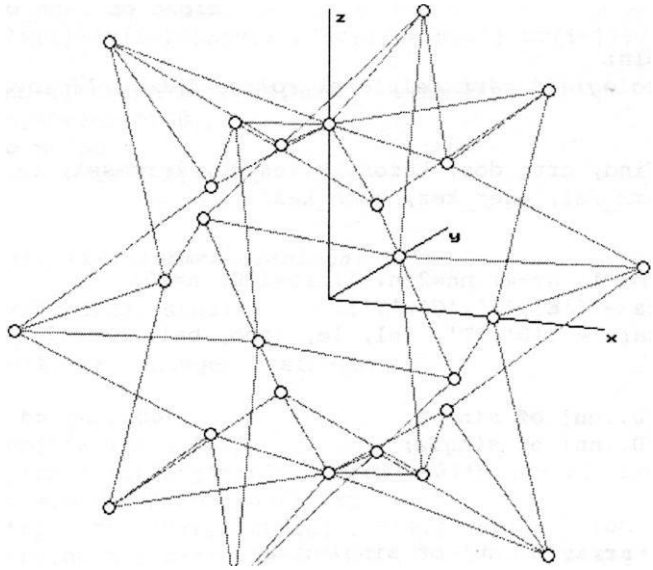
a

3.8. ábra

Egyik megoldás például az, hogy a pólusokban az érintősíkot eleve a *z* tengelyre merőlegesnek vesszük (ezáltal a délgörbék végpontjaiban az érintők egy síkba kerülnek - eltűnik a pólusokban a szingularitás), ez viszont bonyolultabb. nem gömbszimmetriájú felületek esetén *(3.11. ábra)* nem alkalmazható.

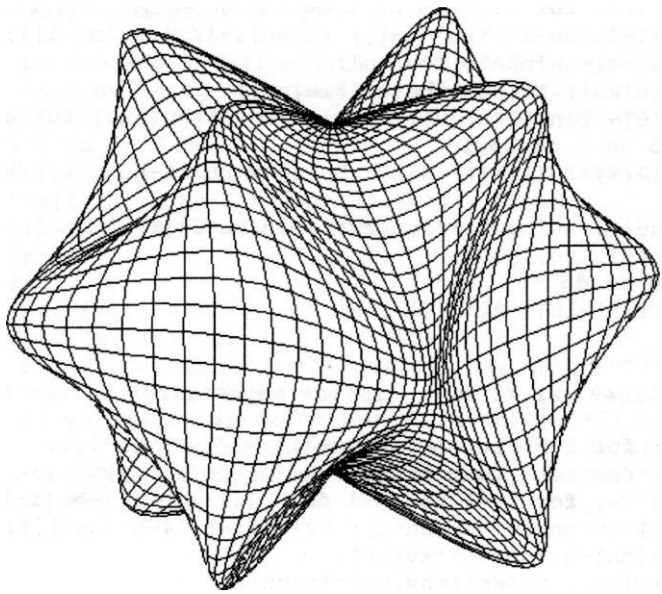
Másik, általánosabb megoldás, hogy a délgörbék szerkesztésekor az illető délgörbe saját keretét a vele (topológiailag) szemben fekvő délgörbe keretével kiegészítve kapott zárt karakterisztikus keretet alkalmazzuk (ehhez a délgörbék száma páros kell legyen). Utána a kapott zárt görbét kettéosztjuk és mindkét délgörbe pontjait a maguk helyén használjuk fel a szélességi görbék szerkesztéséhez. Ezzel az eljárással történt a *3.8.b ábrán* látható gömbfelület illesztése a 3.7. *ábrán* látható poliéderre. Megfigyelhető, hogy a kapott felület a viszonylag kisszámú kontrollpont ellenére is szabályos.

A *3.9. ábrán* látható egy gömb topológiájú zárt felület kontrollpontrácsa. Az interpolációt végző programok kezdőkonfigurációként egy gömb pontjait adják meg. Ezeket elmozdítva lehet a kívánt karakterisztikus poliédert megkapni.



3.9. ábra

A ZM_SPLN.PAS program illeszt gömb topológiájú kontrollpontrácsra zárt spline-felületet. A *3.9. ábrán* látható rácsra illesztett felület *(3.10. ábra):*



3.10. ábra

```
{$N+}

program zm_spin;
    {gömb topológiájú zárt felületek spline interpolációval)

uses
  graph, grafind, crt, dos, kozos, kozos_3d, kerdesek, rajz_f3d,
  konf_3d, komm_fel, eger_kez, gomb_kez;

const
  mm=4; nu=2*mm-1; nv=4; nn=2*nv-1; rg=120; n=10;
  v a l a s z :   k a r s   =
  valasz_f: kars = ['D','T', fel, le, jobb, bal, char(27)];

type
  mu = array[0..nu] of single;
  my = array[0..nn] of single;

var
  gx, gy, gz: mu;
  hu, ku, lu: array[0..nu] of
  single; fx, fy, fz: mv; hv, kv,
  lv: array [ 0 .. nn] of single;
  fi, psi: single;

procedure splineu(var g: mu); {spline együtthatók u irányban}
begin
  ku[0]:=1/4; for i:=1 to nu-1 do ku[i]:=1/(4-ku[i-1]);
  lu[0]:=1/4; for i:=1 to nu-2 do lu[i]:=-lu[i-1]*ku[i];
  hu[0]:=hu[0]/4; for i:=1 to nu-1 do hu[i]:=(hu[i]-hu[i-1])*ku[i];
  ku[nu-1]:=(1-lu[nu-1])*ku[nu-1]; ku[nu]:=1/(4-ku[nu-1]);
  hu[nu]:=(hu[nu]-hu[nu-1])*ku[nu];
  lu[nu]:=1/ku[nu]; hu[nu]:=hu[nu]*lu[nu];
  lu[0]:=(lu[0]-lu[nu])/ku[0]; hu[0]:=(hu[0]-hu[nu])/ku[nu];
  for i:=1 to nu-2 do begin
    lu[i]:=(lu[i]-lu[i-1])/ku[i]; hu[i]:=(hu[i]-hu[i-1])/ku[i];
  end;
  g[0]:=(hu[nu-1]-hu[nu-2])/(ku[nu-1]-lu[nu-2]);
  g[1] :=hu[nu]-lu[nu] *g[0];
  for i:=2 to nu do
    g[i]:=hu[i-2]-lu[i-2]*g[0];
end;

procedure splinev(var f: mv); {spline együtthatók v irányban)
begin
  kv[0]:=1/4; for i:=1 to nn-1 do kv[i]:=1/(4-kv[i-1]);
  lv[0]:=1/4; for i:=1 to nn-2 do lv[i]:=-lv[i-1]*kv[i];
  hv[0]:=hv[0]/4; for i:=1 to nn-1 do hv[i]:=(hv[i]-hv[i-
  1])*kv[i]; kv[nn-1]:=(1-lv[nn-1])*kv[nn-1]; kv[nn]:=1/(4-kv[nn-
  1]); hv[nn]:=(hv[nn]-hv[nn-1])*kv[nn];
  lv[nn]:=1/kv[nn]; hv[nn]:=hv[nn]*lv[nn];
  lv[0]:=(lv[0]-lv[nn])/kv[0]; hv[0]:=(hv[0]-hv[nn])/kv[nn];
```

```
  for i:=1 to nn-2 do begin
    lv[i]:=(lv[i]-lv[i-1])/kv[i]; hv[i]:=(hv[i]-hv[i-1])/kv[i];
  end;
  f[0]:=(hv[nn-1]-hv[nn-2])/(kv[nn-1'-lv[nn-2]);
  f[1] :=hv[nn]-lv[nn]*f[0];
  for i:=2 to nn do
    f[i]:=hv[i-2]-lv[i-2]*f[0];
end;

procedure rajz; {felületmegjelenítés}
var
  u,w,u3,w3,v,t,v3,t3: single;

  function xu(k,j,r: integer): single;
  begin
    for i:=0 to nu-2 do
      hu[i]:=rp[i*n,j*n+r]^.xp-
            2*rp[(i+l)*n,j*n+r]^.xp+rp[(i+2)*n,j*n+r]^.xp;
        hu[nu-1]:=rp[(nu-1)*n,j*n+r]^.xp-
        2*rp[nu*n,j*n+r]^.xp+rp[0,j*n+r]^.xp;
      hu[nu]:=rp[nu*n,j*n+r]^.xp-2*rp[0,j*n+r]^.xp+rp[n,j*n+r]^.xp;
      splineu(gx);
    if k=nu then
      xu:=gx[nu]*t3+gx[0]*v3+(rp[nu*n,j*n+r]^.xp-
          gx[nu])*t+ (rp[0,j*n+r]^.xp-gx[0])*v
    else
      xu:=gx[k]*t3+gx[k+l]*v3+(rp[k*n,j*n+r]^.xp-gx[k])*t+
          (rp[(k+l)*n,j*n+r]^.xp-gx[k+l])*v;
  end;
  function xv(i,l: integer): single;
  begin
    for j:=0 to nv-2 do hv[j]:=a[i,j].xp-
    2*a[i,j+l].xp+a[i,j+2].xp; hv[nv-1]:=a[i,nv-1].xp-
    2*a[i,nv].xp+a[i+mm,nv-1].xp; hv[nv]:=a[i,nv].xp-2*a[i+mm,nv-
    1].xp+a[i+mm,nv-2].xp; for j:=nv+l to nn-2 do
      hv[j]:=a[i+mm,nn-j+l].xp-2*a[i+mm,nn-j].xp+a[i+mm,nn-j-
    1].xp; hv[nn-1]:=a[i+mm,2].xp-2*a[i+mm,1].xp+a[i3O].xp;
    hv[nn]:=a[i+mm,l].xp-2*a[i3O].xp+a[i,l].xp;
    splinev(fx);
    case 1 of
    0..nv-1: xv:=fx[1]*w3+fx[1+1]*u3+(a[i,l].xp-fx[1])*w+(a[i,l+l].xp-
                  fx[1+1])*u;
    nv:      xv:=fx[nv]*w3+fx[nv+l]*u3+(a[i,nv].xp-fx[nv])*w+(a[i+mm,
                   nv-1].xp-fx[nv+l])*u;
    nv+l..nn-1: xv:=fx[1]*w3+fx[1+1]*u3+(a[i+mm,nn-l+1].xp-
                     fx[1])*w+(a[i+mm,nn-1].xp-fx[1+1])*u;
    nn:              xv:=fx[nn]*w3+fx[0]*u3+(a[i+mm,l].xp-
                  fx[nn])*w+(a[i3O].xp-
                  fx [0] ) *u;
   end;
  end;
```

```
function yu(k,j,r: integer): single;
begin
  for i:=0 to nu-2 do
    hu[i]:=rp[i*n,j*n+r]^.yp-
           2*rp[(i+1)*n,j*n+r]^.yp+rp[(i+2)*n,j*n+r]^.yp; hu[nu-
    1]:=rp[(nu-1)*n,j*n+r]^.yp-
    2*rp[nu*n,j*n+r]^.yp+rp[0,j*n+r)^.yp;
    hu[nu):=rp[nu*n,j*n+r]^.yp-2*rp[0,j*n+r]^.yp+rp[n,j*n+r]^.yp;
    splineu (gy) ;
  if k=nu then
    yu:=gy[nu)*t3+gy[0]*v3+(rp[nu*n,j*n+r)^.yp-
        gy[nu])*t+ (rp[0,j*n+r]^.yp-gy[0])*v
  else
    yu:=gy[k]*t3+gy[k+1]*v3+(rp[k*n,j*n+r)^.yp-gy[k])*t+
        (rp[(k+l)*n,j*n+r]^.yp-gy[k+1))*v;
end;
function yv(i,l: integer): single;
begin
  for j:=0 to nv-2 do hv[j]:=a[i,j).yp-
  2*a[i,j+1].yp+a[i,j+2].yp; hv[nv-1]:=a[i,nv-1].yp-
  2*a[i,nv).yp+a[i+mm,nv-1].yp; hv[nv]:=a[i,nv).yp-2*a[i+mm,nv-
  1].yp+a[i+mm,nv-2].yp; for j:=nv+1 to nn-2 do
    hv[j]:=a[i+mm,nn-j+l].yp-2*a[i+mm,nn-j].yp+a[i+mm,nn-j-
  1].yp; hv[nn-1]:=a[i+mm,2].yp-2*a[i+mm,l].yp+a[i3O].yp;
  hv[nn]:=a[i+mm,1].yp-2*a[i3O).yp+a[i,1].yp;
  splinev ( fy) ;
  case 1 of
  0..nv-1: yv:=fy[1]*w3+fy[1+1]*u3+(a[i,l].yp-fy[1])*w+(a[i,1+1].yp-
               fy[l+l])*u;
  nv: yv:=fy[nv]*w3+fy[nv+1]*u3+(a[i,nv].yp-
          fy[nv])*w+(a[i+mm,nv- 1].yp-fy[nv+l))*u;
  nv+l..nn-l: yv:=fy[1]*w3+fy[1+1]*u3+(a[i+mm,nn-1+1].yp-
      fy[1])*w+(a(i+mm,nn-1].yp-fy[1+1])*u; nn:
      yv:=fy[nn]*w3+fy[0]*u3+(a[i+mm,l].yp-fy[nn))*w+(a[i3O).yp-
      fy[0] ) *u;
  end;
end;
function zu(k,j,r: integer): single;
begin
  for i:=0 to nu-2 do
    hu[i]:=rp[i*n,j*n+r]^.zp-
           2*rp[(i+1)*n,j*n+r]^.zp+rp[(i+2)*n,j*n+r]^.zp;
    hu[nu-1]:=rp[(nu-1)*n,j*n+r]^.zp-
    2*rp[nu*n,j*n+r]^.zp+rp[0,j-n+r]^.zp;
    hu[nu]:=rp[nu*n,j*n+r]^.zp-
    2*rp[0,j*n+r)^.zp+rp[n,j*n+r]^.zp; splineu (gz) ;
  if k=nu then
    zu:=gz[nu]*t3+gz[0]*v3+(rp[nu*n,j*n+r)^.zp-
        gz[nu])*t+ (rp[0,j*n+r)^.zp-gz[0])*v
  else
    zu:=gz[k]*t3+gz[k+l]*v3+(rp[k*n,j*n+r]^.zp-gz[k])*t+
        (rp[(k+l)*n,j*n+r]^.zp-gz[k+l])*v;
end;
```

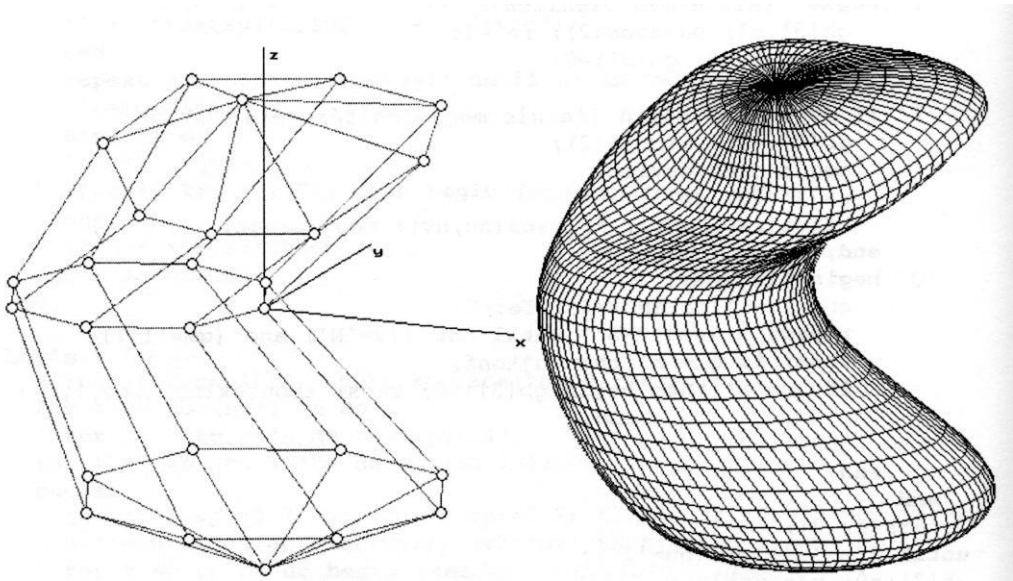
```
function zv(i,l: integer): single;
begin
  for j:=0 to nv-2 do hv[j]:=a[i,j].zp-2*a[i,j+l].zp+a[i,j+2].zp;
  hv[nv-1]:=a[i,nv-1].zp-2*a[i,nv].zp+a[i+mm,nv-1].zp;
  hv[nv]:=a[i,nv].zp-2*a[i+mm,nv-1].zp+a[i+mm,nv-2].zp;
  for j:=nv+1 to nn-2 do
    hv[j]:=a[i+mm,nn-j+1].zp-2*a[i+mm,nn-j].zp+a[i+mm,nn-j-1].zp;
  hv[nn-1]:=a[i+mm,2].zp-2*a[i+mm,1].zp+a[i3O].zp;
  hv[nn]:=a[i+mm,l].zp-2*a[i3O].zp+a[i,l].zp;
  splinev(fz);
  case 1 of
  0..nv-1: zv:=fz[1]*w3+fz[l+i]*u3+(a[i,l].zp-fz[1])*w+(a[i,1+1].zp-
           fz[1+1])*u;
  nv: zv:=fz[nv]*w3+fz[nv+1]*u3+(a[i,nv].zp-fz[nv])*w+(a[i+mm,nv-
           1].zp-fz[nv+l])*u;
  nv+l..nn-l: zv:=fz[1]*w3+fz[1+1]*u3+(a[i+mm,nn-l+l].zp-
           fz[1])*w+(a[i+mm,nn-1].zp-fz[l+l])*u;
  nn: zv:=fz[nn]*w3+fz[0]*u3+(a[i+mm,l].zp-fz[nn])*w+(a[i3O].zp-
           fz [0] ) *u;
 end;
end;

procedure rajzracs; {felületháló csomópontjainak kiszámítása}
var r1, p1: integer;
begin
  for k:=0 to mm-1 do begin
    for 1:=0 to nv-1 do begin
      if 1=0 then r1:=0 else r1:=1;
      for r:=r1 to n do begin
        u:=r/n; w:=1-u; u3:=u*u*u; w3:=w*w*w;
        rp[k*n,l*n+r]^.uj (xv(k,l),yv(k,l),zv(k,l) );
      end;
    end;
    for 1:=nv to nn do begin
      if 1=nv then r1:=0 else r1:=1;
      for r:=r1 to n do begin
        u:=r/n; w:=1-u; u3:=u*u*u; w3:=w*w*w;
        rp[(k+mm)*n, (nn-1)*n+n-r]A.uj(xv(k,1),yv(k,1),zv(k,1));
      end;
    end;
  end;
  for 1:=0 to nv-1 do begin
    if 1=0 then r1:=0 else r1:=1;
    for r:=r1 to n do begin
      u:=r/n; w:=1-u;
      for k:=0 to nu do begin
        if k=0 then pl:=0 else p1:=1;
        for p:=p1 to n do begin
          v:=p/n; t:=1-v; v3:=v*v*v; t3:=t*t*t;
          rp[k*n+p,l*n+r]A.uj (xu(k,l,r),yu(k,l,r),zu(k,l,r) );
        end;
      end;
    end;
```

```
      end;
      f1:=1; fp:=0;
    end;

begin
  racs; if gb[2]=1 then nyíl; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu+1,nv,n,1,0,0); eger_l;
  fr:=1;
end;

procedure fg; {vetítési irány változtatásának vezérlése}
begin
  repeat
     setcolor(7);
     case qj of
     'D':outtextxy(15,102,'T - tele');
     'T':outtextxy(15,102,'D - drótváz');
     end;
     repeat kr:=upcase(readkey); until kr in valasz_f;
     cleardevice;
     case kr of
     'D','T': qj:=kr;
     else if kr<>chr(27) then begin forgat; fp:=0; end;
     end;
     if kr<>chr(27) then rajz;
  until kr=chr(27);
end;

begin
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to (nu+1)*n do
     for l:=0 to nv*n do new(rp[k,l]);
  for k:=1 to (nu+1)*nv do new(hl[k]);
  repeat
     qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
     nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
     for k:=0 to nu do begin {kezdőkonfiguráció - gömb}
      psi:=2*pi*k/(nu+1);
      for l:=0 to nv do begin
         fi:=pi*l/nv;
         a [k, 1] .uj (rg*sin (fi) *cos (psi) , rg*sin (fi) *sin (psi) , rg*cos (fi)) ;
      end;
     end;
     qk:='N';
     racs; konfig('ZMS'); eger_n; parancs(2); racs;
     kpontok(nu+l,nv,1,0);

     repeat
      eger_t(kx0,2,620,ky0); eger_1; qr:=' '; b:=0;
      repeat par katt(2,qr); kp r katt(nu,nv); until qr in valasz;
      eger_n;
      case qr of
```

```
      'V':begin {kontrollpont kiválasztása és koordinátáinak
                 módosítása}
            gb[3]:=0; fe:=0; racs; kpontok(nu+l,nv,1,0);
            gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
            inform(szx,szy,'    Hely: '); kp_kor(szx,szy,12);
            koordin3('Hely');
            if qr<>char(27) then
              if (szy=0) or (szy=nv) then for k:=0 to nu do
                    a[k,szy] .uj (x,y,z)
            {a pólus mozgatásának összehangolása mindenik délgörbére}
                    else a [szx, szy] . uj (x, y, z) ;
            kp_kor(szx,szy,7); racs; kpontok(nu+1,nv,1,0);
            gb[1]:=0; lap(208,5); setfillstyle(1,11);
            bar(130,470,590,480);
            fl:=0;
          end;
      'F':begin {felületmegjelenítés}
            gb[3] :=1; parancs(2); fe:=1;
            megj; rajz; gb[3]:=0;
          end;
      'E':if fe=1 then begin {felületmegjelenítés keret nélkül)
            gb[2]:=1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb[2]:=0;
            keret(kx0, ky0); kp_racs(nu,nv); megj; rajz;
          end;
      'Q':begin {kilépés}
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('ZMS') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs; kpontok(nu+l,nv,1,0);
                           end;
            gb[4] :=0;
          end;
      end;
      parancs(2);
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```

A *3.11 ábra* egy másik, ugyancsak gömb topológiájú karakterisztikus poliédert és rá illesztett spline felületet ábrázol. Ezúttal a felső pólusban az érintősík nem merőleges a *z* tengelyre, maga a pólus sincs a *z* tengelyen (a pólus elnevezés itt kissé erőltetett, topológiai szempontból viszont helyes).

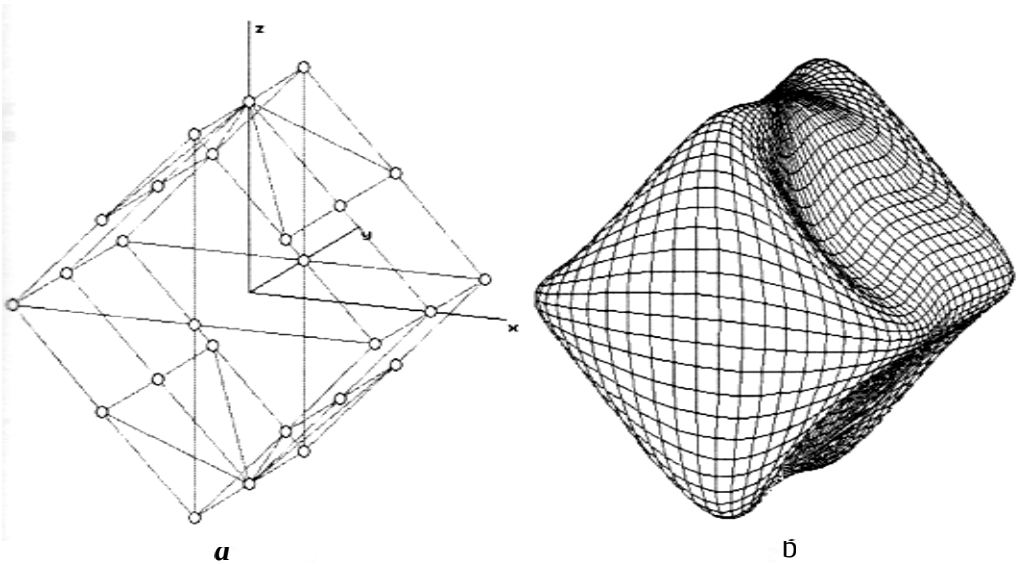
A *3.12* és *3.13. ábrákon* két, geometriailag azonos - csak különböző helyzetű - kocka egy-egy felosztása és a kapott karakterisztikus keretre illesztett spline-



a b

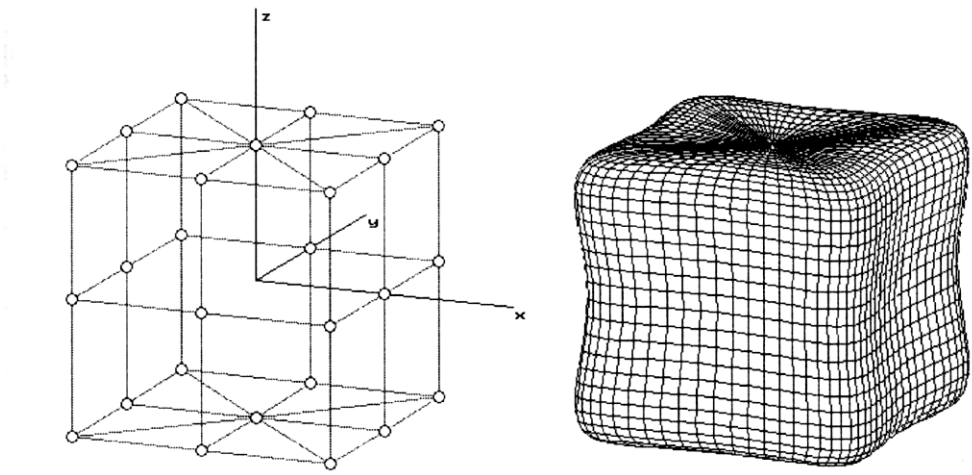
3.11. ábra

felület látható. Megfigyelhető, hogy a zárt felületek modellezésére bemutatott módszer - két pólus kiválasztása és a felület felosztása délgörbék és szélességi görbék hálójával - nem minden esetben kielégítő (bár az a lényeges előnye, hogy a paramétertartomány négyzetes felosztása nagymértékben megkönnyíti a spline felület előállítását). Ahhoz, hogy a kapott felület ne függjön a karak- terisztikus poliéder viszonyítási rendszerétől (a bemutatott példákban a pólusok helye a felületen meghatározó jellegű míg a *z* tengely körüli forgatás nem okoz változást), a gyakorlatban háromszögekből álló karakterisztikus poliéderre illesztenek spline, vagy inkább B-spline felületeket, ezeknek előállítása azonban lényegesen bonyolultabb.



a b

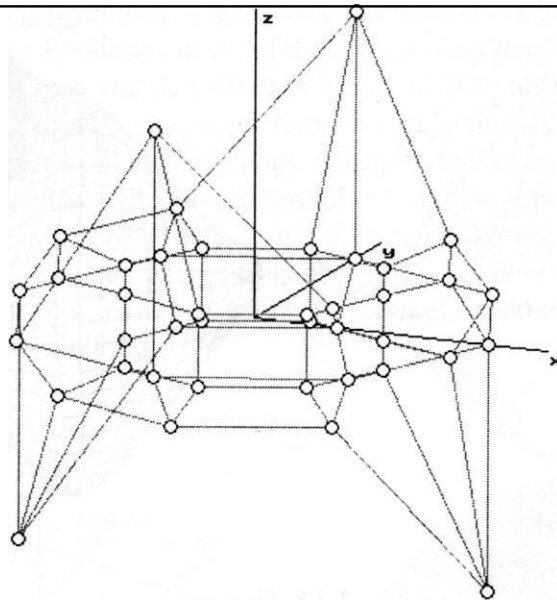
3.12.



a b

3.13. ábra

A *3.14. ábrán* látható egy gyűrű topológiájú zárt felület karakterisztikus poliédere. Az interpolációt végző programok kezdőkonfigurációként egy tórusz pontjait adják meg. Ezt módosítva lehet a kívánt felületalakot biztosító konfigurációt megkapni.



3.14. ábra

A ZY_SPLN.PAS program illeszt gyűrű topológiájú kontrollpontrácsra zárt spline-felületet. A fenti karakterisztikus poliéderre illesztett felület a 3.15. ábrán látható.

```
{$N+}

program zy spin;
    (gyűrű topológiájú zárt felületek spline interpolációval)

uses
 graph, grafind, crt, dos, kozos, kozos_3d, kerdesek, rajz_f3d,
 konf_3d, komm fel, eger_kez, gomb_kez;

const
 nu=7; nv=4; ru=120; ry=40; n=10;
 valasz: kars =
 valasz_f: kars = ['D','T', fel, le, jobb, bal, char(27)];

type
 mu = array[0..nu] of single;
 mv = array[0..nv] of single;

var
 gx, gy, gz: mu;
 hu, ku, lu: array[0..nu] of single;
 fx, fy, fz: mv;
```

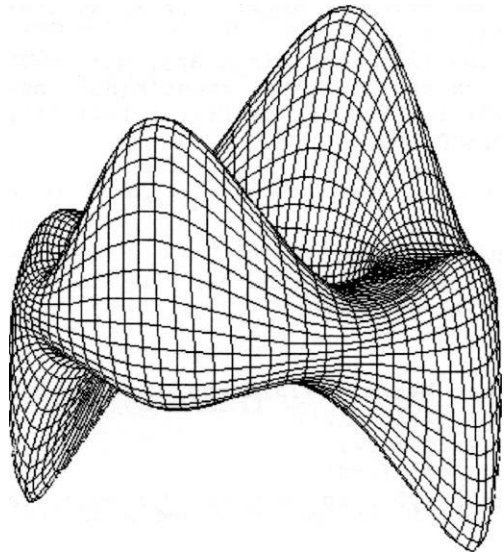
```
  hv, kv, lv: array[0..nv] of single;
  fi, psi: single;

procedure splineu(var g: mu); {spline együtthatók u irányban)
begin
  ku[0]:=1/4; for i:=1 to nu-1 do ku[i]:=1/(4-ku[i-1]); lu[0]:=1/4;
  for i:=1 to nu-2 do lu[i]:=-lu[i-1]*ku[i]; hu[0]:=hu[0]/4; for
  i:=1 to nu-1 do hu[i]:=(hu[i]-hu[i-1])*ku[i]; ku[nu-1]:=(1-lu[nu-
  1])*ku[nu-1]; ku[nu]:=1/(4-ku[nu-1]); hu[nu]:=(hu[nu]-hu[nu-
  1])*ku[nu];
  lu[nu]:=1/ku[nu]; hu[nu]:=hu[nu]*lu[nu];
  lu[0]:=(lu[O]-lu[nu])/ku[0]; hu[0]:=(hu[0]-hu[nu])/ku[nu];
  for i:=1 to nu-2 do begin
    lu[i]:=(lu[i]-lu[i-1])/ku[i]; hu[i]:=(hu[i]-hu[i-1])/ku[i];
  end;
  g[0]:=(hu[nu-1]-hu[nu-2))/(ku[nu-1]-lu[nu-2]);
  g[1]:=hu[nu]-lu[nu]*g[0];
  for i:=2 to nu do
    g[i]:=hu[i-2]-lu[i-2]*g[0];
end;

procedure splinev(var f: mv); {spline együtthatók v irányban}
begin
  kv[0]:=1/4; for i:=1 to nv-1 do kv[i]:=1/(4-kv[i-1]);
  lv[0]:=1/4; for i:=1 to nv-2 do lv[i]:=-lv[i-1]*kv[i];
  hv[0]:=hv[0]/4; for i:=1 to nv-1 do hv[i]:=(hv[i]-hv[i-1])*kv[i];
  kv[nv-1]:=(i-lv[nv-1])*kv[nv-i]; kv[nv]:=1/(4-kv[nv-i]);
  hv[nv]:=(hv[nv]-hv[nv-1])*kv[nv];
  lv[nv]:=1/kv[nv]; hv[nv]:=hv[nv]*lv[nv];
  lv[0]:=(lv[0]-lv[nv])/kv[0]; hv[0]:=(hv[0]-hv[nv])/kv[nv];
  for i:=1 to nv-2 do begin
    lv[i]:=(lv[i]-lv[i-i])/kv[i]; hv[i]:=(hv[i]-hv[i-i])/kv[i];
  end;
  f[0]:=(hv[nv-1]-hv[nv-2])/(kv[nv-1]-lv[nv-2]); f[1]:=hv[nv]-
        lv[nv] *f [0] ;
  for i:=2 to nv do
    f[i]:=hv[i-2]-lv[i-2]*f[0];
end;

procedure rajz; {felületmegjelenítés}
var
  u,w,u3,w3,v,t,v3,t3: single;

  function xu(k,j,r: integer): single;
  begin
    for i:=0 to nu-2 do
      hu[i]:=rp[i*n,j*n+r]^.xp-
             2*rp[(i+l)*n,j*n+r]^.xp+rp[(i+2)*n,j*n+r]^.xp;
       hu[nu-1]:=rp[(nu-i)*n,j*n+r]^.xp-
       2*rp[nu*n,j*n+r]^.xp+rp[O,j*n+r].xp;
       hu[nu]:=rp[nu*n,j*n+r]^.xp-
       2*rp[0,j*n+r]^.xp+rp[n,j*n+r]^.xp; splineu(gx);
    if k=nu then
```

```
    xu:=gx[nu]*t3+gx[0]*v3+(rp[nu*n,j*n+r]^.xp-gx[nu])*t+
        (rp[0,j*n+r]^.xp-gx[0])*v
  else
    xu:=gx[k]*t3+gx[k+l]*v3+(rp[k*n,j*n+r]^.xp-gx[k])*t+
        (rp[(k+l)*n,j*n+r]^.xp-gx[k+l])*v;
end;
function xv(i,l: integer): single;
begin
  for j:=0 to nv-2 do
    hv[j]:=a[i,j].xp-2*a[i,j+1].xp+a[i,j+2].xp;
  hv[nv-1]:=a[i,nv-1].xp-2*a[i,nv].xp+a[i30].xp;
  hv[nv]:=a[i,nv].xp-2*a[i30].xp+a[i,l].xp;
  splinev(fx);
  if 1=nv then
    xv:=fx[nv]*w3+fx[0]*u3+(a[i,nv].xp-fx[nv])*w+(a[i30].xp-fx[0])*u
  else
    xv:=fx[1]*w3+fx[1+1]*u3+(a[i,l].xp-fx[1])*w+(a[i,l+l].xp-
        fx[1+1])*u;
end;

function yu(k,j,r: integer): single;
begin
  for i:=0 to nu-2 do
    hu[i]:=rp[i*n,j*n+r]^.yp-
           2*rp[(i+l)*n,j*n+r]^.yp+rp[(i+2)*n,j*n+r]^.yp;
  hu[nu-1]:=rp[(nu-1)*n,j*n+r]^.yp-
            2*rp[nu*n,j*n+r]^.yp+rp[0,j*n+r]^.yp;
  hu[nu]:=rp[nu*n,j*n+r]^.yp-2*rp[0,j*n+r]^.yp+rp[n,j*n+r]^.yp;
  splineu(gy);
  if k=nu then
    yu:=gy[nu]*t3+gy[0]*v3+(rp[nu*n,j*n+r]^.yp-gy[nu])*t+
        (rp[0,j*n+r]^.yp-gy[0])*v
  else
    yu:=gy[k]*t3+gy[k+l]*v3+(rp[k*n,j*n+r]^.yp-gy[k])*t+
        (rp[(k+l)*n,j*n+rl^.yp-gy[k+l])*v;
end;
function yv(i,l: integer): single;
begin
  for j:=0 to nv-2 do
    hv[j]:=a[i,j].yp-2*a[i,j+l].yp+a[i,j+2].yp;
  hv[nv-1]:=a[i,nv-1].yp-2*a[i,nv].yp+a[i30].yp;
  hv[nv]:=a[i,nv].yp-2*a[i30].yp+a[i,l].yp;
  splinev ( fy) ;
  if 1=nv then
    yv:=fy[nv]*w3+fy[0]*u3+(a[i,nvl.yp-fy[nv])*w+(a[i30].yp-fy[0])*u
  else
    yv:=fy[1]*w3+fy[l+l]*u3+(a[i,l].yp-fy[1])*w+(a[i,l+l].yp-
        fy[l+l])*u;
end;

function zu(k,j,r: integer): single;
begin
```

```
    for i:=0 to nu-2 do
      hu[i]:=rp[i*n,j*n+r]^.zp-
            2*rp[(i+1)*n,j*n+r]^.zp+rp[(i+2)*n,j*n+r]^.zp; hu[nu-
      1]:=rp[(nu-1)*n,j*n+r]^.zp-
      2*rp[nu*n,j*n+r]^.zp+rp[0,j*n+r]^.zp;
      hu[nu]:=rp[nu*n,j*n+r]^.zp-
      2*rp[0,j*n+r]^.zp+rp[n,j*n+r]^.zp; splineu(gz);
    if k=nu then
      zu:=gz[nu]*t3+gz[0]*v3+(rp[nu*n,j*n+r]^.zp-gz[nu])*t+
          (rp[0,j*n+r]^.zp-gz[0])*v
    else
      zu:=gz[k]*t3+gz[k+1]*v3+(rp[k*n,j*n+r]^.zp-gz[k])*t+
          (rp[(k+1)*n,j*n+r]^.zp-gz[k+1])*v;
end;
function zv(i,l: integer): single;
begin
    for j:=0 to nv-2 do
      hv[j]:=a[i,j].zp-2*a[i,j+1].zp+a[i,j+2].zp;
    hv[nv-1]:=a[i,nv-1].zp-2*a[i,nv].zp+a[i3O].zp;
    hv[nv]:=a[i,nv].zp-2*a[i3O].zp+a[i,l].zp;
    splinev(fz);
    if 1=nv then
      zv:=fz[nv]*w3+fz[0]*u3+(a[i,nv].zp-fz[nv])*w+(a[i3O].zp-fz[0])*u
    else
      zv:=fz[1]*w3+fz[l+1]*u3+(a[i,ll.zp-fz[1])*w+(a[i,l+l].zp-
          fz[1+1])*u;
end;
procedure rajzracs; (felületháló csomópontjainak kiszámítása}
var rl, pl: integer;
begin
    for k:=0 to nu do
      for 1:=0 to nv do begin
        if 1=0 then r1:=0 else r1:=1;
        for r:=r1 to n do begin
          u:=r/n; w:=1-u; u3:=u*u*u; w3:=w*w*w;
          rp[k*n,l*n+r]^.uj (xv(k,l),yv(k,l),zv(k,l) );
        end;
      end;
    for 1:=0 to nv do begin
      if 1=0 then r1:=0 else r1:=1;
      for r:=r1 to n do begin
        u:=r/n; w:=1-u;
        for k:=0 to nu do begin
          if k=0 then pl:=0 else pl:=1;
          for p:=p1 to n do begin
            v:=p/n; t:=1-v; v3:=v*v*v; t3:=t*t*t;
            rp[k*n+p,1*n+r]^.uj (xu(k,l,r),yu(k,l,r),zu(k,l,r) );
          end;
        end;
      end;
    end;
    fl:=1; fp:=0;
end;
```

```
begin
  racs; if gb[2]=1 then nyíl; setcolor(15);
  if f1=0 then rajzracs;
  egern; megjelen(nu+1,nv+1,n,0,1,1,1); eger_1;
  fr:=1;
end;

procedure fg; {vetítési irány változtatásának vezérlése}
begin
  repeat
    setcolor(7);
    case qj of
    'D':outtextxy(15,102,'T - tele');
    'T':outtextxy(15,102,'D - drótváz');
    end;
    repeat kr:=upcase(readkey); until kr in valasz_f;
    cleardevice;
    case kr of
    'D','T': qj:=kr;
    else if kr<>chr(27) then begin forgat; fp:=0; end;
    end;
    if kr<>chr(27) then rajz;
  until kr=chr(27);
end;

begin
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to (nu+1)*n do
    for 1:=0 to (nv+1)*n do new(rp[k,l]);
  for k:=1 to (nu+1) * (nv+1) do new (hl [k]) ;
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nu do begin (kezdőkonfiguráció - gyűrű}
      psi:=2*pi*k/(nu+1);
      for 1:=0 to nv do begin
        fi:=2*pi*1/(nv+1)+pi/2;
        a[k,l].uj((ru+rv*sin(fi))*cos(psi),(ru+ry*sin(fi))*
        sin (psi) , ry*cos (fi)) ;
      end;
    end;
    qk:='N';
    racs; konfig('ZYS'); eger_n; parancs(2); racs;
    kpontok(nu+1,nv+1,1,1);
    repeat
      eger_t(kx0,2,620,ky0); eger_1; qr:=''; b:=0;
      repeat par katt(2,qr); kp r katt(nu,nv); until qr in valasz;
      eger_n;
      case qr of
         'V':begin {kontrollpont kiválasztása és koordinátáinak
                    módosítása}
           gb[3]:=0; fe:=0; racs; kpontok(nu+1,nv+1,1,1);
           gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
```

```
            inform(szx,szy,'   Hely: '); kp_kor(szx,szy,12);
            koordin3('Hely');
            if qr<>char(27) then a[szx,szy].uj(x,y,z);
            kp_kor(szx,szy,7); racs; kpontok(nu+1,nv+1,1,1);
            gb[1]:=0; lap(208,5); setfillstyle(1,11);
            bar(130,470,590,480);
            fl:=0;
          end;

      'F':begin (felületmegjelenítés)
           gb[3]:=1; parancs(2);
           fe:=1; megj; rajz; gb[3]
           :=0; end;

      'E':if fe=1 then begin (felületmegjelenítés keret nélkül)
            gb[2]:=1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb [ 2 ] : =0;
            keret(kx0, ky0) ; kp_racs (nu, nv) ; megj; rajz;
          end;

      'Q':begin (kilépés)
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('ZYS') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs;
                kpontok(nu+1,nv+1,1,1); end;
            gb[4] :=0;
          end;
      end;
      parancs(2);
   until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);

 until qu='N';
closegraph;
end.
```



3.15. ábra

Euler-Monge előállítású felületek esetén a paraméterrács éppen a kontrollpontok x és y koordinátáit adja meg. A Gauss előállítás (3.1 összefüggés) három függvényének megfelelő interpoláció helyett itt csak egyetlen koordináta *(z)* szerinti interpolációt kell elvégezni.

Másik különbség, hogy a felületnek meg lehet különböztetni egy felső és egy alsó oldalát. Hogy egy felületelemnek melyik oldalát látjuk, azt a vetítési irány és a felületelem normálisvektora közötti skaláris szorzat előjeléből lehet megállapítani. A *3.17. ábrán* látható felület alsó oldala sötétebb árnyalatú. A felületet a F_EM_SPL.PAS nevű program illesztette a *3.16. ábrán* látható kontrollpontrácsra.

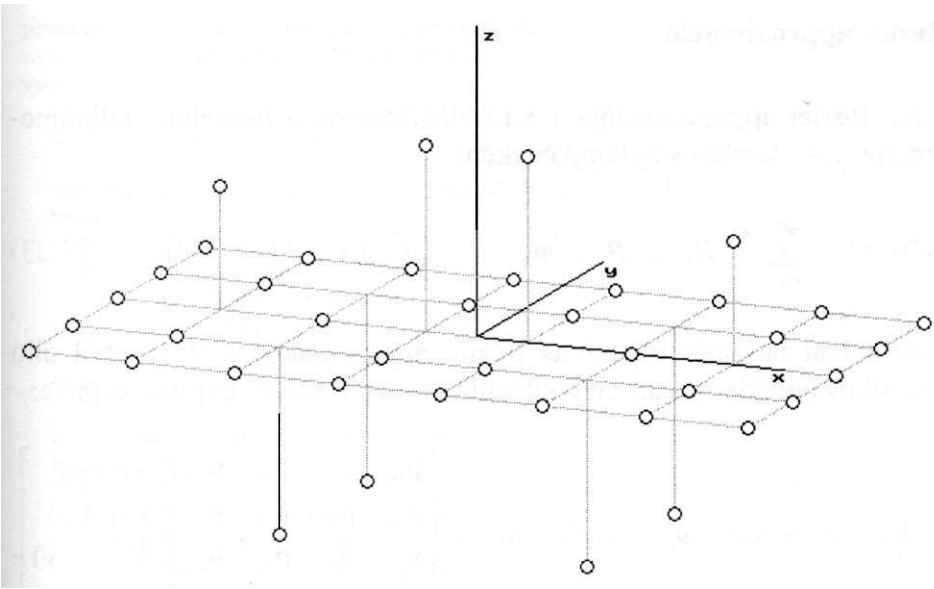
```
{$N+}

program f_em_spl; {Euler Monge előállítású felületek
                   spline interpolációval}

uses
   graph, grafind, crt, dos, kozos, kozos_3d, filekez, kerdesek,
   konf 3d, komm fel, eger kez, gomb kez;

const
  nx=7; ny=4; ax=70; ay=70; nr=12; ex=2*ax/nx/nr; ey=2*ay/ny/nr;
  valasz: kars = ['E','F','Q','V'];
```

```
type
  p5 = array[1..5] of pont;

var
  g: array[0..nx) of single;
  hx, qx: array[l..nx-1] of single;
  f: array[0..ny] of single;
  hy, qy: array[l..ny-1) of single;

procedure koordin; (ordináta (z) betáplálás)
begin
  lap (208, 5) ;
  setcolor(0); outtextxy(300,13,'Magasság:'); setcolor(13);
  outtextxy(275,30,'z = '); ablak(302,27,340,39);
  repeat kn:=''; 1:=1; iras(5,298,29); val(kn,z,hk);
    if hk<>0 then ablak(300,27,342,39); until (hk=0) or (qr=chr(27));
end;

procedure splinex(j,p: byte); (spline együtthatók x irányban)
begin
  qx[1]:=1/4; for i:=2 to nx-1 do qx[i]:=1/(4-qx[i-1]);
  for i:=1 to nx-1 do
    hx[i]:=rp[(i-1)*nr,j*nr+p]^.zp-
    2*rp[i*nr,j*nr+p)^.zp+rp[(i+l)*nr,j*nr+p]^.zp;
  g[0]:=0; g[nx]:=0; hx[1]:=hx[1]/4;
  for i:=2 to nx-1 do hx[i]:=qx[i]*(hx[i]-hx[i-1]); g[nx-1]:=hx[nx-1];
  for i:=nx-2 downto 1 do g[i]:=hx[i]-qx[i]*g[i+l];
end;

procedure spliney(i: byte); {spline együtthatók y irányban)
begin
  qy[1]:=1/4; for j:=2 to ny-1 do qy[j]:=1/(4-qy[j-1]);
  for j:=1 to ny-1 do hy[j]:=a[i,j-1].zp-2*a[i,j].zp+a[i,j+l].zp;
  f[0]:=0; f[ny]:=0; hy[1]:=hy[1]/4;
  for j:=2 to ny-1 do hy[j]:=qy[j]*(hy[j]-hy[j-1]); f[ny-1]:=hy[ny-1];
  for j:=ny-2 downto 1 do f[j]:=hy[j]-qy[j]*f[j+l];
end;

procedure inform; {ordináta (z) kiírás}
var ksl, ks2, kz: string;
begin
  setcolor(0); str(szx,ksl); str(szy,ks2); str(a[szx,szy].zp:3:0,kz);
  outtextxy(130,470,'Sorszám: '+ksl+','+ks2+'    Magasság:   z ='+kz);
end;

procedure rajz; {felületmegjelenítés}
var
  u,w,u3,w3,v,t,v3,t3: single;

  function xy(i,j: byte): single;
  begin xy:=a[i,j].xp*w+a[i,j+l].xp*u; end;
  function xx(i,j: byte): single;
  begin xx:=a[i,j].xp*t+a[i+l,j].xp*v; end;
```

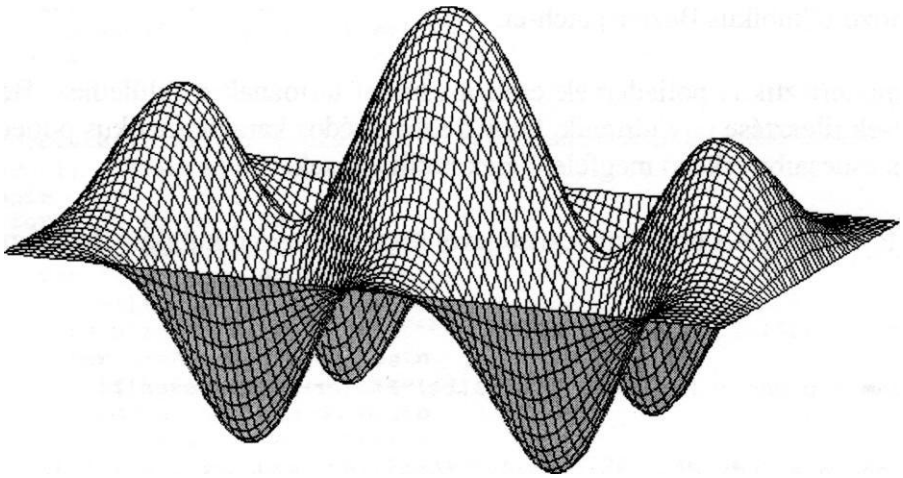
```
function yy(i,j: byte): single;
begin yy:=a[i,j].yp*w+a[i,j+l].yp*u; end;
function yx(i,j: byte): single;
begin yx:=yy(i,j)*t+yy(i+l,j)*v; end;
function zx(i,j,p: byte): single; {x irányú interpoláció}
begin
  splinex (j, p) ;
  zx:=g[i]*t3+g[i+l]*v3+(rp[i*nr,j*nr+p]^.zp-g[i])*t+
      (rp[(i+l)*nr,j*nr+p]A.zp-g[i+l])*v;
end;
function zy(i,j: byte): single; (y irányú interpoláció)
begin
  spliney(i);
  zy:=f [j) *w3+f [j+l] *u3+(a[i, j] .zp-f [j]) *w+(a[i, j+l] .zp-f [j+1]) *u;
end;

procedure rajzracs; {felületháló csomópontjainak kiszámítása)
begin
  for k:=0 to nx do begin
    for 1:=0 to ny-1 do
      for p:=0 to nr do begin
        u:=p/nr; w:=1-u; u3:=u*u*u; w3:=w*w*w;
        rp[k*nr,l*nr+p]^.uj(0,0,zy(k,l));
      end;
  end;
  for 1:=0 to ny-1 do
    for p:=0 to nr do begin
      u : =p/nr; w:=1-u;
       for k:=0 to nx-1 do
        for r:=0 to nr do begin
          v:=r/nr; t:=1-v; v3:=v*v*v; t3:=t*t*t;
          rp[k*nr+r,l*nr+p]^.uj(xx(k,l),yx(k,l),zx(k,l,p));
        end;
  end;
end;

procedure rajzol; {felületháló képsíkra vetítése és megjelenítése}
const cl=ex*ey;
var
  al, bl: single;
  kx, ky: array [ 1 .. 4 ] of integer;
  x,y,z: array[1..4] of single;
  sk: p5;
  procedure negyszog;
  var q: byte;
  begin
    x[1]:=rp[k,l]^.xp; y[1]:=rp[k,l)".yp; z[1]:=rp[k,l]^.zp;
    x[2]:=rp[k-1,1]^.xp; y[2]:=y[1]; z[2]:=rp[k-1,1)^.zp;
    x[3]:=x[2]; y[3]:=rp[k,l-1]^.yp; z[3]:=rp[k-1,1-1]^.zp;
    x[4]:=x[1]; y[4]:=y[3]; z[4]:=rp[k,1-1]^.zp;
    for q:=1 to 4 do begin
      kx[q]:=round((-bp*x[q]+ap*y[q])/nvl);
      ky[q]:=round((-cp*(ap*x[q]+bp*y[q])+sqr(nvl)*z[q])/nv2);
```

```
        sk[q].uj(kxl+kx[q],kyl-ky[q]);
      end;
      al:=ey*(z[2]-z(1]); bl:=(z[4]-z[1])*ex;
      if al*ap+bl*bp+cl*cp>0 then setfillstyle(1,0) else
         setfillstyle(1,7);
      fillpoly (4, sk) ;
    end;
    procedure sor;
    begin
      if by>0 then for 1:=1 to ny*nr do negyszog
              else for 1:=ny*nr downto 1 do negyszog;
    end;
  begin
    if ap>0 then for k:=1 to nx*nr do sor
            else for k:=nx*nr downto 1 do sor;
  end;

begin
  racs; if gb[2]=1 then nyíl; setcolor(15);
  rajzracs;
  eger_n; rajzol; eger_l;
end;

procedure fg; {vetítési irány változtatásának vezérlése)
begin
  repeat
    repeat kr:=readkey;
    until (kr=fel) or (kr=le) or (kr=jobb) or (kr=bal) or
    (ord(kr)=27);
    cleardevice;
    if kr<>chr(27) then begin forgat; rajz; end;
  until kr=chr(27);
end;

procedure kpontok; {kontrollpontok megjelenítése}
begin
  setcolor(8);
  for k:=0 to nx do for 1:=0 to ny do a[k,l].koord; if
  gb [ 2 ] =0 then begin
    for k:=0 to nx do begin
      vetit(a[k,0].xp,a(k,0].yp,0); moveto(xt,yt);
    vetit(a[k,ny].xp,a[k,ny].yp,0); lineto(xt,yt);
    end;
    for 1:=0 to ny do begin
      vetit(a[0,1].xp,a[0,1].yp,0); moveto(xt,yt);
    vetit(a[nx,l].xp,a[nx,l).yp,0); lineto(xt,yt);
    end;
    for k:=0 to nx do
      for 1:=0 to ny do begin
        vetit(a[k,l].xp,a[k,l].yp,0); moveto(xt,yt);
        lineto(a[k,l] .xr,a[k,l] .yr);
      end;
    setcolor(15);
```

```
    for 1:=0 to ny do
      for k:=0 to nx do a[k,l].megj(0);
  end;
end;
begin
  grindhi; keret(kx0, ky0); kp_racs(nx,ny); eger_k;
  for k:=0 to nx*nr do
    for 1:=0 to ny*nr do new(rp[k,l]);
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nx do for 1:=0 to ny do
      a[k,l].uj((k-nx/2)*ax,(l-ny/2)*ay,0);
    qk:='N'; racs; konfig('FEM'); eger_n; parancs(2); racs; kpontok;
    repeat
      eqer_t(kx0,2,620,ky0); eqer_1; qr:=' '; b:=0;
      repeat par_katt(2,qr); kp_r_katt(nx,ny); until qr in valasz;
      eger_n;
      case qr of
      'V':begin {kontrollpont kiválasztása és magasságának módosítása}
            gb[3]:=0; fe:=0; racs; kpontok;
            gb[1]:=1; parancs(2); if b=0 then sorszam(nx,ny); inform;
            kp_kor(szx,szy,12); koordin;
            if qr<>chr(27) then
            a[szx,szy].uj(a[szx,szy].xp,a[szx,szy].yp,z);
            kp_kor(szx,szy,7); racs; kpontok;
            gb(1):=0; lap(208,5); setfillstyle(1,11);
            bar(130,470,590,480);
          end;
      'F':begin {felületmegjelenítés}
            gb[3]:=1; parancs(2); fe:=1; rajz; gb[3]:=0;
          end;
      'E':if fe=1 then begin {felületmegjelenítés keret nélkül}
            gb[2]:=1; parancs(2); cleardevice;
            rajz; fg; gb[2]:=0;
            keret(kx0, ky0) ; kp_racs (nx, ny) ; rajz;
          end;
      'Q':begin {kilépés}
            gb [ 4 ] : =1; parancs(2);
            repeat kilep('FEM'); until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin fe:=0; racs; kpontok; end;
            gb[4]:=0;
          end;
      end;
      parancs(2);
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```



3.16. ábra



3.17. ábra

3.3.3. Bézier approximáció

A felületek Bézier approximációja a 3.13 előállításban a Bernstein-polinomokat használja (2.41 képlet) súlyfüggvényként:

$$\mathbf{r}(u,v) = \sum_{k=0}^{m}\sum_{l=0}^{n} B_{km}(u)B_{ln}(v)\mathbf{p}_{kl} \;; \qquad (u,v) \in [0,1]\times[0,1] \,. \qquad (3.23$$

A leggyakrabban használt bikubikus Bézier approximáció 4x4 pontból álló karakterisztikus poliéderhez állít elő közelítő felületet. Ennek explicit képlete:

$$\mathbf{r}(u,v) = \begin{bmatrix}(1-u)^3 & 3u(1-u)^2 & 3u^2(1-u) & u^3\end{bmatrix} \begin{bmatrix}\mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{02} & \mathbf{p}_{03}\\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13}\\ \mathbf{p}_{20} & \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23}\\ \mathbf{p}_{30} & \mathbf{p}_{31} & \mathbf{p}_{32} & \mathbf{p}_{33}\end{bmatrix} \begin{bmatrix}(1-v)^3\\ 3v(1-v)^2\\ 3v^2(1-v)\\ v^3\end{bmatrix},$$

$$(u,v) \in [0,1]\times[0,1] \qquad (3.24)$$

A *3.18. ábra* egy karakterisztikus poliédert ábrázol, a *3.19. ábra* pedig a hozzátartozó bikubikus Bézier-patch-et.

A karakterisztikus poliédernek csak a csúcsai tartoznak a felülethez. Bézier-patch-ek illesztése úgy történik, hogy a szomszédos karakterisztikus poliéderek közös csúcsaiba érkező megfelelő élek kollineárisak.

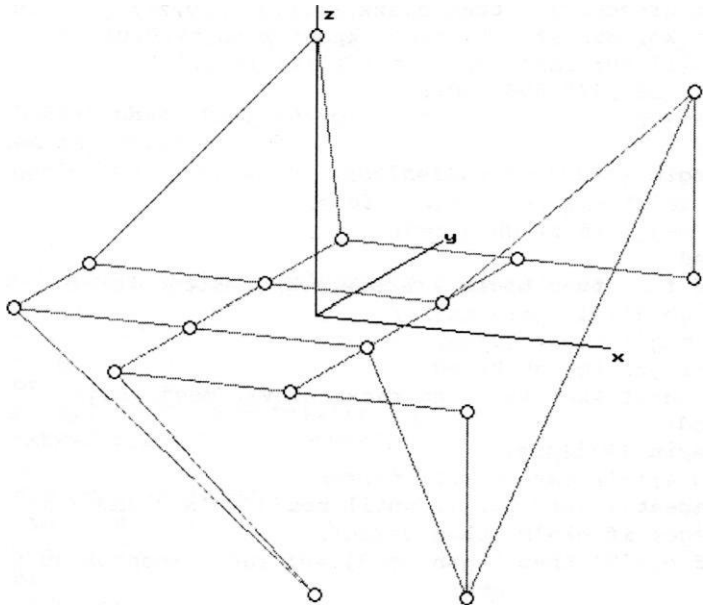
A F_P_BEZR.PAS program segítségével szerkeszthető bikubikus Bézier-patch.

```
{$N+}

program f p bezr; {felületfolt (patch) Bézier-közelítéssel}

uses
  graph, grafind, crt, dos, kozos, kozos_3d, kerdesek, rajz_f3d,
  konf_3d, komm_fel, eger_kez, gomb kez;

const
  nu=3; nv=3; ax=120; ay=120; n=16;
  valasz: kars =
  valasz f: kars = ['D','T', fel, le, jobb, bal, char(27)];
```

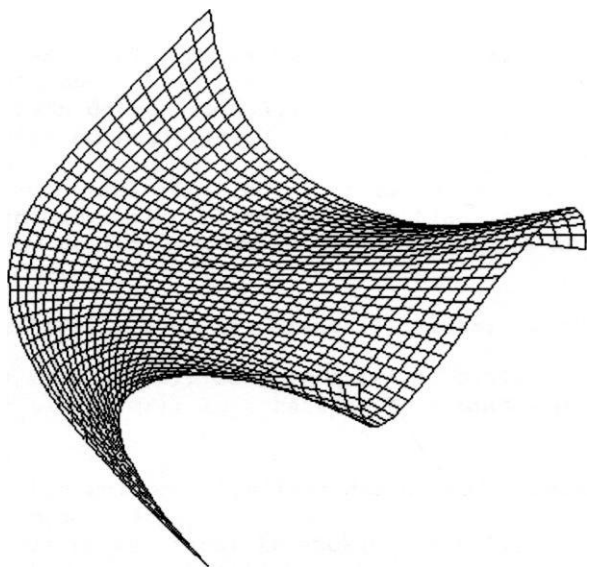
```
procedure rajz; (felületmegjelenítés}
type valt = array[0..3] of real;
var
  uu, vv: valt;
  u, v, w, t, s: real;

  function xx(uu,vv: valt): real;
  begin
    s:=0;
    for i:=0 to nu do
      for j:=0 to nv do
        s:=s+a[i,j].xp*uu[i]*vv[j];
    xx:=s;
  end;

  function yy(uu,vv: valt): real;
  begin
    s:=0;
    for i:=0 to nu do
      for j:=0 to nv do
        s:=s+a[i,j].yp*uu[i]*vv[j];
    yy:=s;
  end;

  function zz(uu,vv: valt): real;
  begin
    s:=0;
    for i:=0 to nu do
      for j:=0 to nv do
        s:=s+a[i,j].zp*uu[i]*vv[j];
    zz:=s;
  end;

  procedure rajzracs; (felületháló csomópontjainak kiszámítása)
  var rl, pl: integer;
  begin
    for 1:=0 to nv-1 do begin
      if 1=0 then r1:=0 else r1:=1;
      for r:=rl to n do begin
        v:=(l+r/n)/nv; t:=1-v;
        vv[0]:=t*t*t; vv[1]:=3*t*t*v; vv[2]:=3*t*v*v; vv[3]:=v*v*v;
        for k:=0 to nu-1 do begin
          if k=0 then pl:=0 else pl:=1;
          for p:=p1 to n do begin
            u:=(k+p/n)/nu; w:=1-u;
            uu[0]:=w*w*w; uu[1]:=3*w*w*u; uu[2]:=3*w*u*u;
                    uu[3]:=u*u*u;
            rp[k*n+p,1*n+r]^.uj(xx(uu,vv),yy(uu,vv),zz(uu,vv));
          end;
        end;
      end;
    end;
```

```
      fl:=1; fp:=0;
   end;

begin
   racs; if gb[2]=1 then nyíl; setcolor(15);
   if f1=0 then rajzracs;
   eger_n; megjelen(nu,nv,n,1,0,0,1); eger_1;
   fr:=1;
end;

procedure fg; (vetítési irány változtatásának vezérlése)
begin
   repeat
     setcolor(7);
     case qj of
     'D':outtextxy(15,102,'T - tele');
     'T':outtextxy(15,102,'D - drótváz');
     end;
     repeat kr:=upcase(readkey);
     until kr in valasz_f;
     cleardevice;
     case kr of
     'D','T': qj:=kr;
     else if kr<>chr(27) then begin forgat; fp:=0; end;
     end;
     if kr<>chr(27) then rajz;
   until kr=chr(27);
end;

begin
   grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
   for k:=0 to nu*n do
     for 1:=0 to nv*n do new(rp[k,1]);
   for k:=1 to nu*nv do new(hl[k]);
   repeat
     qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
     nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
     for k:=0 to nu do for 1:=0 to nv do a[k,1].uj((k-nu/2)*ax,(1-
       nv/2)*ay,0);
     qk:='N';
     racs; konfig('FPB'); eger_n; parancs(2); racs; kpontok(nu,nv,0,0);
     repeat
       eger_t(kx0,10,620,ky0); eger_1; qr:=' '; b:=0;
       repeat par katt(2,qr); kp r katt(nu,nv); until qr in valasz;
       eger_n;
       case qr of
       'V':begin (kontrollpont kiválasztása és koordinátáinak
                  módosítása}
           gb[3]:=0; fe:=0; racs; kpontok(nu,nv,0,0);
           gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
           inform(szx, szy, '    Hely:   ' ) ;
           kp kor(szx,szy,12); koordin3('Hely');
```

```
          if qr<>chr(27) then a[szx,szy].uj(x,y,z);
          kp_kor(szx,szy,7); racs; kpontok(nu,nv,0,0);
          gb[1]:=0; lap(208,5); setfillstyle(1,11);
          bar(130,470,590,480);
          fl:=0;
        end;
      'F':begin {felületmegjelenítés}
            gb[3]:=1; parancs(2); fe:=1;
            megj; rajz; gb[3] :=0;
          end;
       'E':if fe=1 then begin {felületmegjelenítés keret nélkül)
            gb[2]:=1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb[2] :=0;
            keret(kx0, ky0); kp_racs(nu,nv); megj; rajz;
          end;
      'Q':begin {kilépés}
          gb[4]:=1; parancs(2); fe:=0;
          repeat kilep('FPB'); until not ((v='N') and (qm='I'));
          vege; if qk='N' then ujkonf;
          if qu='N' then begin gb[3]:=0; racs; kpontok(nu,nv,0,0);
                         end;
          gb[4]:=0;
        end;
      end;
      parancs(2);
    until (qk='I') or (qu='I');
     gb [ 3 ] : =0; parancs(2);
  until qu='N';
  closegraph;
end.
```



3.18. ábra



3.19. ábra

3.3.4. B-spline approximáció

A B-spline közelítés esetében a 3.13 előállítás súlyfüggvényei egyváltozós B-spline polinomok. Periodikus B-spline-ok alkalmazásakor a kapott felület olyan foltokból áll, amelyeket a karakterisztikus keretnek egy-egy (m+1)x(n+1) kontrollpontból álló darabja határoz meg, ha *m* és *n* az *u* illetve v változójú B-spline blendingfüggvények foka. Hasonlóan a görbeszerkesztéshez, egy nyílt B-spline felület akkor éri el a határgörbéit, ha a karakterisztikus keret szélein álló kontrollpontokat annyiszor többszörösnek tekintjük, ahány az illető határvonalra merőleges irányú blendingfüggvény foka. A keret négy sarkán *mxn* szeres kontrollpontok állnak, ezek rajta lesznek a szerkesztett felületen.

A másodfokú B-spline felületek előállítása:

$$\mathbf{r}(u,v)=\begin{bmatrix}\frac{(1-u)^2}{2} & 1-\frac{(1-u)^2+u^2}{2} & \frac{u^2}{2}\end{bmatrix}\mathbf{P}\begin{bmatrix}\frac{(1-v)^2}{2} & 1-\frac{(1-v)^2+v^2}{2} & \frac{v^2}{2}\end{bmatrix}^T,$$
$$(u,v)\in[0,1]\times[0,1] \qquad (3.25)$$

ahol

$$\mathbf{P}=\begin{bmatrix}\mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{02}\\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{12}\\ \mathbf{p}_{20} & \mathbf{p}_{21} & \mathbf{p}_{22}\end{bmatrix} \qquad (3.26)$$

a karakterisztikus keretnek a szerkesztett folthoz tartozó darabja (az indexek átírva, a $\mathbf{p_{11}}$ az a kontrollpont, amely köré szerkesztjük a felületfoltot). A teljes felület a 3.25 képletet a karakterisztikus poliéder minden belső kontrollpontjára alkalmazva állítható elő, kiegészítve a szegélyek előállításával kétszeres szélső kontrollpontokra. A *3.20. ábra* azt az egyszerű esetet szemlélteti, amikor a patch karakterisztikus kerete az xy síkban fekvő egységnyi vonalközű négyzetrács:

$$\mathbf{P}=\begin{bmatrix}(0,0,0) & (0,1,0) & (0,2,0)\\ (1,0,0) & (1,1,0) & (1,2,0)\\ (2,0,0) & (2,1,0) & (2,2,0)\end{bmatrix}. \qquad (3.27$$

Ebben az esetben a felület egy sík négyzet lesz, amely a *3.20.a ábrán* látható, előállítása pedig:

$$\mathbf{r}:[0,1]\times[0,1]\rightarrow\left[\frac{1}{2},\frac{3}{2}\right]\times\left[\frac{1}{2},\frac{3}{2}\right]\times\{0\}$$
$$\mathbf{r}(u,v)=\left(u+\frac{1}{2},v+\frac{1}{2},0\right) \quad , \tag{3.28}$$

vagyis egyenletes megjelenítési paraméterfelosztás

$$u=\frac{i}{n_r} \quad , i=0,1,\ldots,n_r \quad ; \quad v=\frac{j}{n_r} \quad , j=0,1,\ldots,n_r \tag{3.29}$$

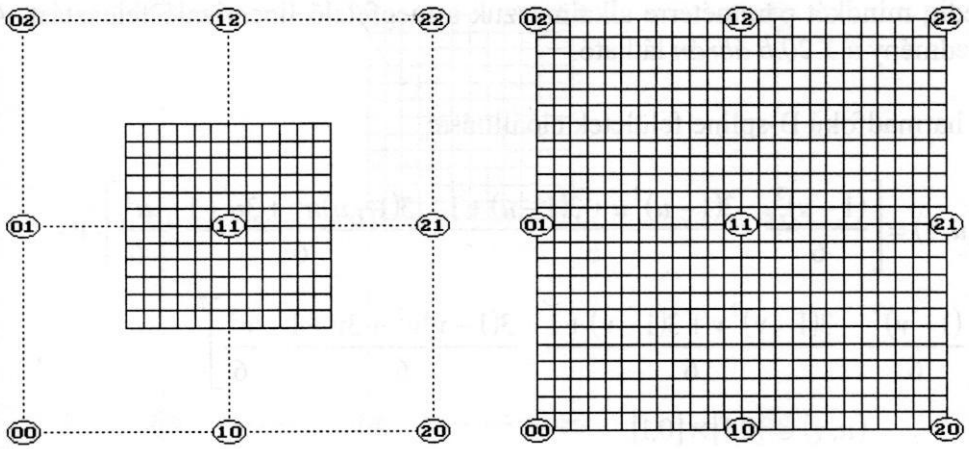
mellett a paramétervonalak egyenletes négyzetes hálót alkotnak. Ha most a felületet ki akarjuk terjeszteni a karakterisztikus keret széléig, akkor a szélső kontrollpontokat kétszeresnek tekintve állíthatók elő a szegélyező patch-ek. Például a

$$\mathbf{P}=\begin{bmatrix}(0,0,0) & (0,1,0) & (0,2,0)\\ (0,0,0) & (0,1,0) & (0,2,0)\\ (1,0,0) & (1,1,0) & (1,2,0)\end{bmatrix} \tag{3.30}$$

keret az

$$\mathbf{r}:[0,1]\times[0,1]\rightarrow\left[0,\frac{1}{2}\right]\times\left[\frac{1}{2},\frac{3}{2}\right]\times\{0\}$$
$$\mathbf{r}(u,v)=\left(\frac{u^2}{2},v+\frac{1}{2},0\right) \tag{3.31}$$

előállítást eredményezi, amely már *u* irányban nem lineáris, tehát a háló a széleken nem egyenletes felosztású. Ez bizonyos alkalmazások esetében (amikor a kontrollpontok térbeli eloszlása olyan, hogy a széleken ritkábbak) egyenesen előnyös lehet. Ha mégis azt akarjuk, hogy a megjelenítési háló egyenletes legyen, akkor a fenti esetben az



a *b*

3.20. ábra

A

$$\mathbf{P} = \begin{bmatrix} (1,0,0) & (1,1,0) & (1,2,0) \\ (2,0,0) & (2,1,0) & (2,2,0) \\ (2,0,0) & (2,1,0) & (2,2,0) \end{bmatrix} \tag{3.33}$$

kere

$$\mathbf{r}:[0,1]\times[0,1] \rightarrow \left[\frac{3}{2},2\right]\times\left[\frac{1}{2},\frac{3}{2}\right]\times\{0\}$$
$$\mathbf{r}(u,v) = \left(2-\frac{\left(1-u^2\right)}{2}, v+\frac{1}{2}, 0\right) \tag{3.34}$$

előállítású foltjának linearizáló felosztása pedig:

$$u = 1 - \sqrt{1 - \frac{2i}{n_r}} \quad , i = 0,1,\ldots,\frac{n_r}{2} \quad ; \quad v = \frac{j}{n_r} \quad , j = 0,1,\ldots,n_r \quad . \qquad (3.35$$

A v szerint szélső foltok előállítása hasonlóan történik, a sarokfoltok esetében pedig mindkét paraméterre alkalmazzuk a megfelelő linearizáló felosztást. Az eredmény a *3.20.b ábrán* látható.

A harmadfokú B-spline felületek előállítása:

A

$$\mathbf{r}(u,v) = \left[\frac{(1-u)^3}{6} \quad \frac{3(1-u)^2u + 3(1-u) + 1}{6} \quad \frac{3(1-u)u^2 + 3u + 1}{6} \quad \frac{u^3}{6}\right] \cdot \mathbf{P} \cdot$$

$$\cdot \left[\frac{(1-v)^3}{6} \quad \frac{3(1-v)^2v + 3(1-v) + 1}{6} \quad \frac{3(1-v)v^2 + 3v + 1}{6} \quad \frac{v^3}{6}\right]^T$$

$$(u,v) \in [0,1] \times [0,1]$$

(3.3

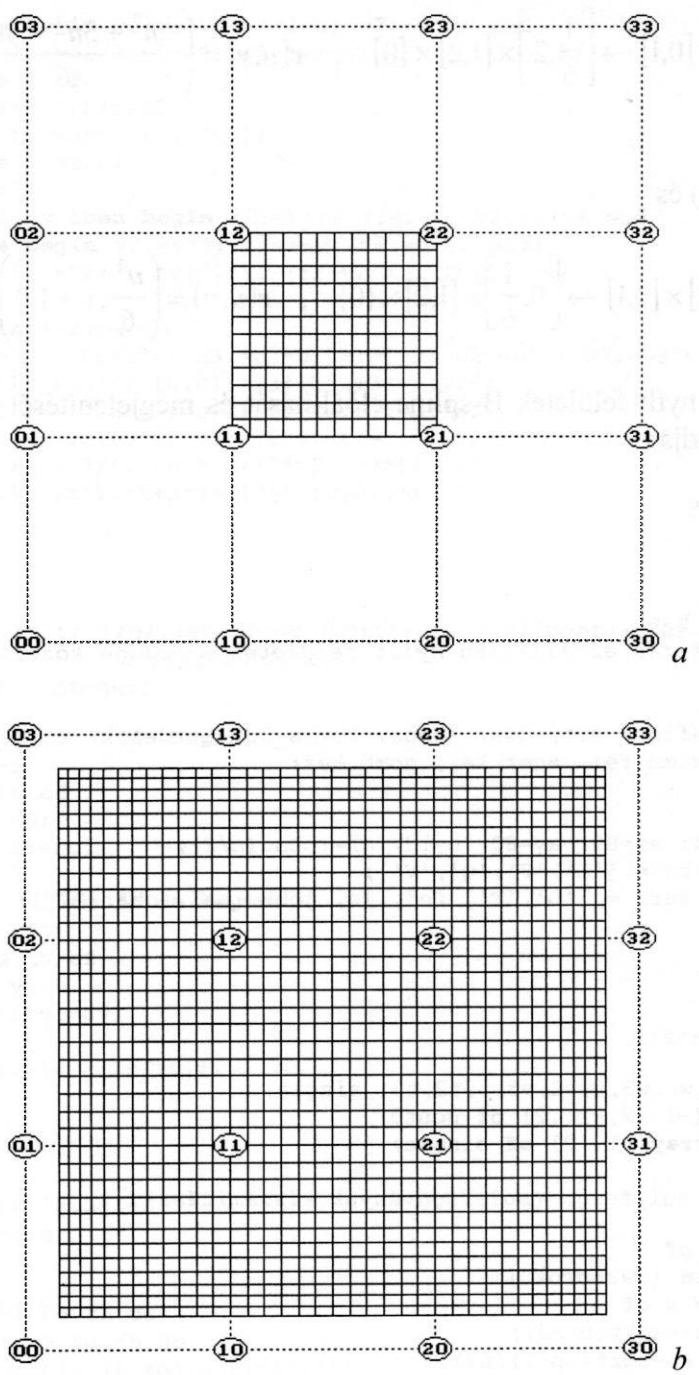
ahol

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}_{02} & \mathbf{p}_{03} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13} \\ \mathbf{p}_{20} & \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23} \\ \mathbf{p}_{30} & \mathbf{p}_{31} & \mathbf{p}_{32} & \mathbf{p}_{33} \end{bmatrix} . \qquad (3.3$$

3.21.a ábrán az xy síkban fekvő egységnyi vonalközű négyzetrácsra kapott

$$\mathbf{r}:[0,1]\times[0,1] \to [1,2]\times[1,2]\times\{0\} \quad , \qquad \mathbf{r}(u,v) = (u+1, v+1, 0) \qquad (3.38)$$

egyenletes felosztású foltot láthatjuk. A szegélyezéshez a megfelelő szélső pontokat előbb kétszeresnek, majd háromszorosnak tekintve az előállítási keret "elcsúsztatható" az illető irányba (miközben a széleken "torlódnak" az alappontok). Így kaphatók például a csökkenő *u* irányban szegélyező


03
13
23
33
02
12
22
32
01
11
21
31
00
10
20
30
a
03
13
23
33
02
12
22
32
01
11
21
31
00
10
20
30
b

3.21. ábra

$$\mathbf{r}:[0,1]\times[0,1]\to\left[\frac{1}{6},2\right]\times[1,2]\times\{0\} \quad , \quad \mathbf{r}(u,v)=\left(\frac{-u^3+3u^2+3u+1}{6},v+1,0\right) \tag{3.39}$$

(3.2.1.b ábra) és

$$\mathbf{r}:[0,1]\times[0,1]\to\left[0,\frac{1}{6}\right]\times[1,2]\times\{0\} \quad , \quad \mathbf{r}(u,v)=\left(\frac{u^3}{6},v+1,0\right) \tag{3.40}$$

patch-ek. A nyílt felületek B-spline előállítását és megjelenítését végző program forráskódja:

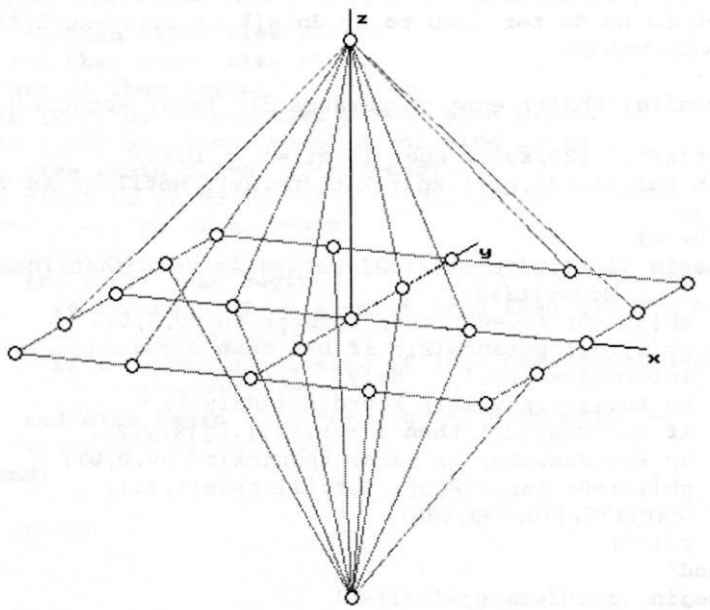
```
F_P_BSPL.PAS

{$N+}

program f_p_bspl;
    {paraméteres előállítású nyílt felületek B-spline közelítéssel}

uses
 graph, grafind, crt, dos, kozos, kozos_3d, kerdesek, rajz_f3d,
 konf_3d, komm_fel, eger_kez, gomb_kez;

const
 nu=4; nv=4; ax=80; ay=80; n=12; n2=round(n/2);
 valasz: kars =
 valasz_f: kars = ['D','T', fel, le, jobb, bal, char(27)];

var
 fh: byte;

procedure rajz;
var
 u,u2,u3,w,w2,w3,v,v2,v3,t,t2,t3: single;
 kp: array[-1..2,-1..2] of pont3;
 fu, fv: array[-1..2] of single;

 procedure sulyfugg; {súlyfüggvények kiszámítása}
 begin
   case qr of
   '2':begin {másodfokú}
       case k of
       0: u:=sqrt(p/n2);
       nu: w:=sqrt((p-1)/n2);
       else u:=p/n;
       end;
       if k=nu then begin w2:=w*w; u:=1-w; u2:=u*u; end
```

```
      else begin u2:=u*u; w:=1-u; w2:=w*w; end;
      fu[-1]:=w2/2; fu[0]:=1-(w2+u2)/2; fu[1]:=u2/2;
      case 1 of
      0: v:=sqrt(r/n2);
      nv: t:=sqrt((r-1)/n2);
      else v:=r/n;
      end;
      if 1=nv then begin t2:=t*t; v:=1-t; v2:=v*v; end
      else begin v2:=v*v; t:=1-v; t2:=t*t; end;
      fv[-1]:=t2/2; fv[0]:=1-(t2+v2)/2; fv[1]:=v2/2;
    end;
  '3':begin {harmadfokú}
      u:=p/n; u2:=u*u; u3:=u2*u; w:=1-u; w2:=w*w; w3:=w2*w;
      fu[-1]:=w3/6; fu[0]:=(3*w2*u+3*w+1)/6;
      fu[1]:=(3*u2*w+3*u+1)/6; fu[2]:=u3/6;
      v:=r/n; v2:=v*v; v3:=v2*v; t:=1-v; t2:=t*t; t3:=t2*t;
      fv[-1]:=t3/6; fv[0]:=(3*t2*v+3*t+1)/6;
      fv[1]:=(3*v2*t+3*v+1)/6; fv[2]:=v3/6;
    end;
  end;
end;

procedure folt; {felületfoltok (patch) kontrollpontjainak
                 összerendelése}
var tu, tv: integer;
begin
  for i:=-1 to fh do begin
    tu:=k+i;
    case tu of
    -1:tu:=0;
    nu+1:tu:=nu;
    end;
    for j:=-1 to fh do begin
      tv:=1+j;
      case tv of
      -1:tv:=0;
      nv+1:tv:=nv;
      end;
      kp[i,j]:=a[tu,tv];
    end;
  end;
end;

function xx: single;
var s: single;
begin
  s:=0;
  for i:=-1 to fh do
    for j:=-1 to fh do
      s:=s+kp[i,j].xp*fu[i]*fv[j];
  xx:=s
end;
```

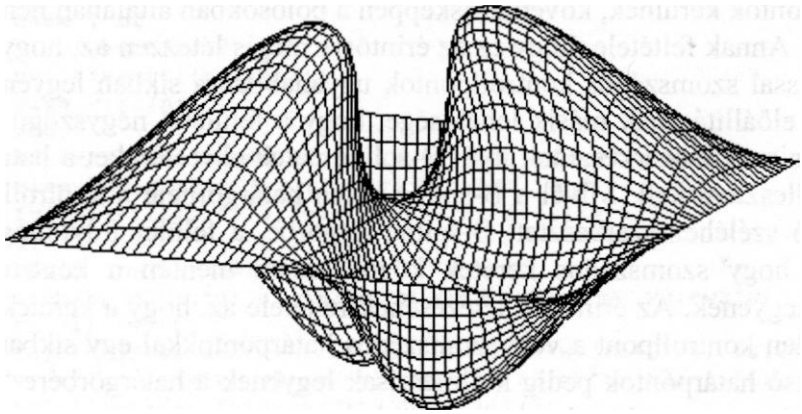
```
  function yy: single;
  var s: single;
  begin
    s:=0;
    for i:=-1 to fh do
      for j:=-1 to fh do
        s:=s+kp[i,j].yp*fu[i]*fv[j];
    yy:=s
  end;

  function zz: single;
  var s: single;
  begin
    s:=0;
    for i:=-1 to fh do
      for j:=-1 to fh do
        s:=s+kp[i,j].zp*fu[i]*fv[j];
    zz:=s
  end;

  procedure rajzracs; {felületháló csomópontjainak kiszámítása)
  var pl, rl, p2, r2, q1, q2: byte;
  begin
    for k:=0 to nu-fh+l do
      for 1:=0 to nv-fh+l do begin
        folt;
        if k=0 then p1:=0 else pl:=1;
        if 1=0 then r1:=0 else r1:=1;
        if qr='2' then begin
          if (k=0) or (k=nu) then p2:=n2 else p2:=n;
          if (1=0) or (1=nv) then r2:=n2 else r2:=n;
        end else begin p2:=n; r2:=n; end;
        for p:=pl to p2 do
          for r:=rl to r2 do begin
            sulyfugg;
            if qr='2' then begin
              if k=0 then ql:=p else if k=nu then ql:=k*n-p+1 else
                    q1:=k*n-n2+p;
              if 1=0 then q2:=r else if 1=nv then q2:=1*n-r+1 else
                    q2:=1*n-n2+r;
            end else begin ql:=k*n+p; q2:=1*n+r; end;
            rp[ql,q2]^.uj (xx,yy,zz);
          end;
      end;
    fl:=1; fp:=0;
  end;

begin
  racs; if gb[2]=1 then nyil; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu,nv,n,1,0,0,1); eger_1;
  fr:=1;
end;
```

```
procedure fg; {vetítési irány változtatásának vezérlése)
begin
  repeat
    setcolor(7);
    case qj of
    'D':outtextxy(15,102,'T - tele');
    'T':outtextxy(15,102,'D - drótváz');
    end;
    repeat kr:=upcase(readkey); until kr in valasz_f;
    cleardevice;
    case kr of
    'D','T': qj:=kr;
    else if kr<>chr(27) then begin forgat; fp:=0; end;
    end;
    if kr<>chr(27) then rajz;
  until kr=chr(27);
end;

begin
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to nu*n do
    for 1:=0 to nv*n do new(rp[k,l]);
  for k:=1 to nu*nv do new(hl[k]);
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nv1*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nu do for 1:=0 to nv do a[k,l].uj((k-nu/2)*ax,
      (1-nv/2) *ay, 0) ;
    qk:='N';
    racs; konfig('FBS'); eger_n; parancs(2); racs; kpontok(nu,nv,0,0);
    repeat
      eger_t(kx0,2,620,ky0); eger_1; qr:=' '; b:=0;
      repeat par katt(2,qr); kp r katt(nu,nv); until qr in valasz;
      eger_n;
      case qr of
         'V':begin {kontrollpont kiválasztása és koordinátáinak
                    módosítása}
            gb[3]:=0; fe:=0; racs; kpontok(nu,nv,0,0);
            gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
            inform(szx,szy,' Hely: ');
            kp_kor(szx,szy,12); koordin3('Hely');
            if qr<>char(27) then a[szx,szy].uj(x,y,z);
            kp_kor(szx,szy,7); racs; kpontok(nu,nv,0,0);
            gb[1]:=0; lap(208,5); setfillstyle(1,11);
            bar(130,470,590,480);
            f1:=0;
          end;
        'F':begin {felületmegjelenítés}
            gb[3]:=1; parancs(2); fe:=1;
            b_spline;
            if ((qr='2') and (fh=2)) or ((qr='3') and (fh=1)) then
            fl:=0;
            if qr='2' then fh:=1 else fh:=2;
```

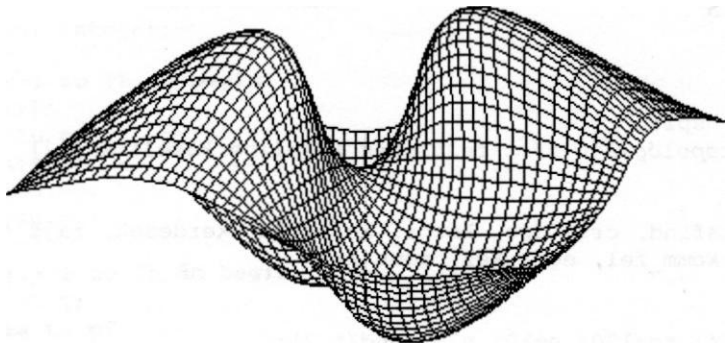
```
            megj; rajz; gb[3]:=0;
          end;
        'E':if fe=1 then begin {felületmegjelenítés keret nélkül}
              gb[2]:=1; parancs(2);
              megj; cleardevice;
              rajz; fg; gb[2] :=0;
              keret (kx0, ky0) ; kp_racs (nu, nv) ; megj; rajz;
            end;
        'Q':begin {kilépés}
              gb[4]:=1; parancs(2); fe:=0;
              repeat kilep('FBS') until not ((v='N') and (qm='I'));
              vege; if qk='N' then ujkonf;
              if qu='N' then begin gb[3]:=0; racs; kpontok(nu,nv,0,0);
                             end;
              gb[4]:=0;
            end;
        end;
        parancs(2);
      until (qk='I') or (qu='I');
      gb[3]:=0; parancs(2);
  until qu=' N' ;
  closegraph;
end.
```



3.22. ábra



3.23. ábra



3.24. ábra

A *3.22. ábrán* látható karakterisztikus keretre szerkesztett másod- és harmadfokú B-spline felületek a *3.23.* illetve a *3.24. ábrán* láthatóak. Megfigyelhető, hogy a harmadfokú felület a karakterisztikus keret "egyenetlenségeit" jobban kisimítj a.

A zárt felületek B-spline előállítása úgy történik, hogy a karakterisztikus poliédert egyik (gömb topológia) vagy mindkét (gyűrű topológia) irányban zártnak tekintjük és ennek függvényében rendeljük a megfelelő kontrollpontokat a foltokat meghatározó helyi rácshoz. Gömb topológiájú zárt felületek

esetén a délgörbék a pólusokban összefutnak, ezért a pólusokba sokszoros kontrollpontok kerülnek, következésképpen a pólosokban általában nem létezik érintősík. Annak feltétele, hogy ez az érintősík mégis létezzen az, hogy a pólus és a pólussal szomszédos kontrollpontok ugyanabban a síkban legyenek. Zárt felületek előállításának másik lehetősége, hogy a felületet négyszögű foltokra osztjuk és ezekre szerkesztünk nyílt B-spline patch-eket. Ezeket a határgörbék mentén illeszteni kell. Mivel a B-spline patch határgörbéje a kontrolipoliéder megfelelő széléhez szerkesztett B-spline térgörbe, a felület folytonosságának feltétele, hogy szomszédos keretek közös határa mentén a kontrollpontok közösek legyenek. Az érintősík létezésének feltétele az, hogy a keretek sarkain álló minden kontrollpont a vele szomszédos határpontokkal egy síkban feküdjön, a belső határpontok pedig kollineárisak legyenek a határgörbére "merőleges" irányban szomszédos kontrollpontokkal.

Az alábbi program gömb topológiájú karakterisztikus poliéderre szerkeszt és jelenít meg másod- vagy harmadfokú B-spline felületet.

ZM_BSPL.PAS

```
{$N+}

program zm_bspl;
        {gömb topológiájú zárt felületek B-spline közelítéssel}

uses
  graph, grafind, crt, dos, kozos, kozos_3d, kerdesek, rajz_f3d, konf
  3d, korom fel, eger kez, gomb kez;

const
  nu=7; nv=4; rg=120; n=10; n2=round(n/2);
  valasz: kars =
  valasz f: kars = ['D','T', fel, le, jobb, bal, char(27)];

var
  fh: byte;
  fi, psi: single;

procedure rajz;
var
  u,u2,u3,w,w2,w3,v,v2,v3,t,t2,t3: single;
  kp: array[-1..2,-1..2] of pont3;
  fu, fv: array[-1..2] of single;

  procedure sulyfugg; {súlyfüggvények kiszámítása} begin
    case qr of
     '2':begin {másodfokú}
```

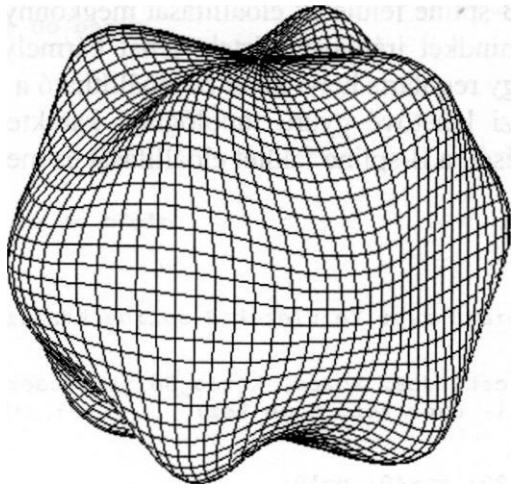
```
        u:=p/n; u2:=u*u; w:=1-u; w2:=w*w;
        fu[-1]:=w2/2; fu[0]:=1-(w2+u2)/2; fu[1]:=u2/2;
        case 1 of
        0: v:=sqrt(r/n2);
        nv: t:=sqrt((r-1)/n2);
        else v:=r/n;
        end;
        if 1=nv then begin t2:=t*t; v:=1-t; v2:=v*v; end
        else begin v2:=v*v; t:=1-v; t2:=t*t; end;
        fv[-1]:=t2/2; fv[0]:=1-(t2+v2)/2; fv[1]:=v2/2;
      end;
    '3':begin {harmadfokú}
        u:=p/n; u2:=u*u; u3:=u2*u; w:=1-u; w2:=w*w; w3:=w2*w;
        fu[-1]:=w3/6;
        fu[0]:=(3*w2*u+3*w+1)/6; fu[1]:=(3*u2*w+3*u+1)/6; fu[2]:=u3/6;
        v:=r/n; v2:=v*v; v3:=v2*v; t:=1-v; t2:=t*t; t3:=t2*t;
        fv[-1] :=t3/6; fv[0] :=(3*t2*v+3*t+1) /6;
        fv[1]:=(3*v2*t+3*v+1)/6; fv[2]:=v3/6;
      end;
    end;
end;

procedure folt; {felületfoltok (patch) kontrollpontjainak
                 összerendelése)
var tu, tv: integer;
begin
    for i:=-1 to fh do begin
      tu:=k+i;
      case tu of
      -1:tu:=nu;
      nu+1:tu:=0;
      nu+2:tu:=1;
      end;
      for j:=-1 to fh do begin
        tv:=1+j;
        case tv of
        -1:tv:=0;
        nv+1:tv:=nv;
        end;
        kp[i,j]:=a[tu,tv];
      end;
    end;
end;

function xx: single;
var s: single;
begin
    s:=0;
    for i:=-1 to fh do
      for j:=-1 to fh do
        s:=s+kp[i,j].xp*fu[i]*fv[j];
    xx:=s
end;
```

```
function yy: single;
  var s: single;
  begin
    s:=0;
    for i:=-1 to fh do
      for j:=-1 to fh do
        s:=s+kp[i,j].yp*fu[i]*fv[j];
    yy:=s
  end;

  function zz: single;
  var s: single;
  begin
    s:=0;
    for i:=-1 to fh do
      for j:=-1 to fh do
        s:=s+kp[i,j].zp*fu[i]*fv[j];
    zz:=s
  end;

  procedure rajzracs; {felületháló csomópontjainak kiszámítása)
  var p1, rl, r2, q2: byte;
  begin
    for k:=0 to nu do
      for 1:=0 to nv-fh+l do begin
        folt;
        if k=0 then pl:=0 else pl:=1;
        if 1=0 then r1:=0 else r1:=1;
        if qr='2' then begin
          if (1=0) or (1=nv) then r2:=n2 else r2:=n;
        end else r2:=n;
        for p:=p1 to n do
          for r:=r1 to r2 do begin
            sulyfugg;
            if qr='2' then begin
              if 1=0 then q2:=r
              else if 1=nv then q2:=1*n-r+1 else q2:=1*n-n2+r;
            end else q2:=1*n+r;
            rp [ k*n+p, q2 ] ^ . uj (xx, yy, z z) ;
          end;
      end;
    fl:=1; fp:=0;
  end;

begin
  racs; if gb[2]=1 then nyíl; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu+l,nv,n,0,1,0,1); eger_1;
  fr:=1;
end;
```

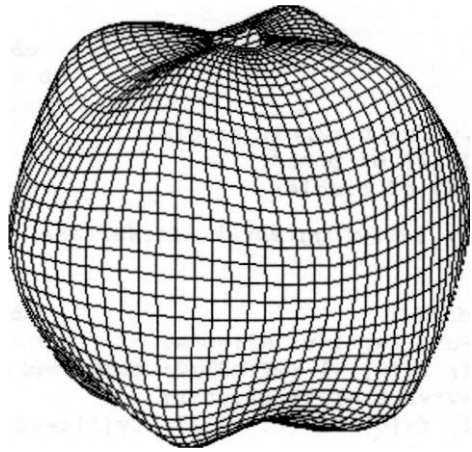
```
procedure fg; {vetítési irány változtatásának vezérlése}
begin
  repeat
    setcolor(7);
    case qj of
    'D':outtextxy(15,102,'T - tele');
    'T':outtextxy(15,102,'D - drótváz');
    end;
    repeat kr:=upcase(readkey); until kr in valasz_f;
    cleardevice;
    case kr of
    'D','T': qj:=kr;
    else if kr<>chr(27) then begin forgat; fp:=0; end;
    end;
    if kr<>chr(27) then rajz;
  until kr=chr(27);
end;

begin
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to (nu+l)*n do
    for 1:=0 to nv*n do new(rp[k,l]);
  for k:=1 to (nu+1) *nv do new (hl [k]) ;
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nu do begin (kezdőkonfiguráció - gömb)
      psi:=2*pi*k/(nu+1);
      for 1:=0 to nv do begin
        fi:=pi*1/nv;
        a[k,l] .uj (rg*sin(fi)*cos(psi),rg*sin(fi)*sin(psi),rg*cos(fi) );
      end;
    end;
    qk:='N';
    racs; konfig('ZMB'); eger_n; parancs(2); racs
    kpontok(nu+1,nv,1,0);
    repeat
      eger_t(kx0,2,620,ky0); eger_1; qr:=' '; b:=0;
      repeat par_katt(2,qr); kp_r_katt(nu,nv); until qr in valasz;
      eger_n;
      case qr of
      'V':begin {kontrollpont kiválasztása és koordinátáinak
                 módosítása}
            gb[3]:=0; fe:=0; racs; kpontok(nu+1,nv,1,0);
            gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
            inform(szx,szy,'    Hely: '); kp_kor(szx,szy,12);
            koordin3('Hely');
            if qr<>char(27) then
              if (szy=0) or (szy=nv) then for k:=0 to nu do
              a[k,szy] .uj (x,y,z)
       (a pólus mozgatásának összehangolása mindenik délgörbére)
                                    else a [szx, szy] . uj (x, y, z) ;
            kp kor(szx,szy,7); racs; kpontok(nu+1,nv,1,0);
```

```
            gb[1]:=0; lap(208,5); setfillstyle(1,11);
            bar(130,470,590,480);
            f1:=0;
          end;
      'F':begin (felületmegjelenítés)
            gb[3] :=1; parancs(2); fe:=1;
            b_spline;
            if ((qr='2') and (fh=2)) or ((qr='3') and (fh=1)) then
            f1:=0;
            if qr='2' then fh:=1 else fh:=2;
            megj; rajz; gb [ 3 ] : =0;
          end;
      'E':if fe=1 then begin {felületmegjelenítés keret nélkül)
            gb [ 2 ] : =1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb[2]:=0;
            keret(kx0, ky0); kp_racs(nu,nv); megj; rajz;
          end;
      'Q':begin {kilépés)
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('ZMB') until not ((v='N') and
            (qm='I')); vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs; kpontok(nu+i,nv,1,0);
                               end;
            gb[4] :=0;
          end;
      end;
      parancs(2);
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```

A *3.9. ábra* kontrolipoliéderét közelítő másod- illetve harmadfokú B-spline felületek a *3.25.* illetve *3.26. ábrákon* láthatók. A harmadfokú felületet nem terjesztettük ki egészen a pólusig, a paramétergörbéknek a pólus környezetében való torlódása miatt. Itt délgörbékhez a pólusokban többszörös alappontokat tekintettünk, ennek következtében a kapott felületnek a pólusai általában szinguláris pontok.



3.25. ábra



3.26. ábra

A következő - a β-spline felületekkel foglalkozó - alpontban bemutatunk egy, a segédkontrollpontok használatán alapuló módszert, amelynek segítségével bizonyos feltételek mellett a pólusokban is sima gömb topológiájú zárt felületek nyerhetők. Minthogy a harmadfokú uniform B-spline közelítés a β-spline sajátos esete, a szóban forgó módszer zárt harmadfokú B-spline felületek szerkesztéséhez is használható.

Gyűrű topológiájú B-spline felületek előállítását megkönnyíti az a tény, hogy a paramétervonalak mindkét irányban zártak, tehát bármely kontrollpont körül megszerkeszthető egy reguláris helyi keret és előállítható a megfelelő patch. Az alábbi program teszi lehetővé gyűrű topológiájú karakterisztikus poliéderek interaktív szerkesztését, a B-spline felület előállítását és megjelenítését.

ZY_BSPL.PAS

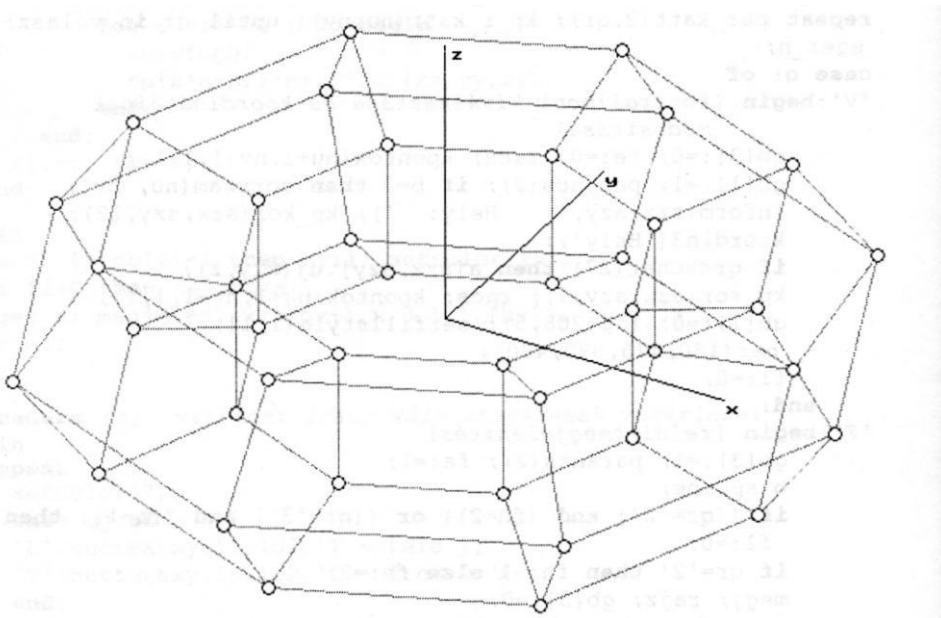
```
{$N+}
program zy bspl;
    {gyűrű topológiájú zárt felületek B-spline közelítéssel}
uses
  graph, grafind, crt, dos, kozos, kozos_3d, kerdesek,
  rajz_f3d, konf_3d, komm_fel, eger_kez, gomb_kez;

const
  nu=7; nv=4; ru=120; rv=40; a=10;
  valasz: kars = ['E','F','Q','V'];
  valasz_f: kars = ['D','T', fel, le, jobb, bal, char(27)];

var
  fh: byte;
  fi, psi: single;

procedure rajz;
var
  u, u2, u3, w, w2, w3, v, v2, v3, t, t2, t3: single;
  kp: array[-1..2,-1..2] of pont3;
  fu, fv: array[-1..2] of single;

  procedure sulyfugg; {súlyfüggvények kiszámítása}
  begin
    case qr of
    '2':begin {másodfokú}
        u:=p/n; u2:=u*u; w:=1-u; w2:=w*w;
        fu[-1]:=w2/2; fu[0]:=1-(w2+u2)/2; fu[1]:=u2/2;
        v:=r/n; v2:=v*v; t:=1-v; t2:=t*t;
        fv[-1]:=t2/2; fv[0]:=1-(t2+v2)/2; fv[1]:=v2/2;
      end;
    '3':begin {harmadfokú)
        u:=p/n; u2:=u*u; u3:=u2*u; w:=1-u; w2:=w*w; w3:=w2*w;
        fu[-1]:=w3/6; fu[0]:=(3*w2*u+3*w+1)/6;
        fu[1]:=(3*u2*w+3*u+1)/6; fu[2]:=u3/6;
        v:=r/n; v2:=v*v; v3:=v2*v; t:=1-v; t2:=t*t; t3:=t2*t;
        fv[-1]:=t3/6; fv[0]:=(3*t2*v+3*t+1)/6;
        fv[1]:=(3*v2*t+3*v+1)/6; fv[2]:=v3/6;
      end;
    end;
  end;
procedure folt; (felületfoltok (patch) kontrollpontjainak
                    összerendelése}
  var tu, tv: integer;
```

```
begin
  for i:=-1 to fh do begin
    tu:=k+i;
    case to of
    -1:tu:=nu;
    nu+1:tu:=0;
    nu+2:tu:=1;
    end;
    for j:=-1 to fh do begin
      tv:=1+j;
      case tv of
      -1:tv:=nv;
      nv+1:tv:=0;
      nv+2:tv:=1;
      end;
      kp[i,j]:=a[tu,tv];
    end;
  end;
end;

function xx: single;
var s: single;
begin
  s:=0;
  for i:=-1 to fh do
    for j:=-1 to fh do
      s:=s+kp[i,j].xp*fu[i]*fv[j];
  xx:=s
end;

function yy: single;
var s: single;
begin
  s:=0;
  for i:=-1 to fh do
    for j:=-1 to fh do
      s:=s+kp[i,j].yp*fu[i]*fv[j];
  YY:=s
end;

function zz: single;
var s: single;
begin
  s:=0;
  for i:=-1 to fh do
    for j:=-1 to fh do
      s:=s+kp[i,j].zp*fu[i]*fv[j];
  zz:=s
end;

procedure rajzracs; {felületháló csomópontjainak kiszámítása) var p1,
rl: byte;
begin
  for k:=0 to nu do
```

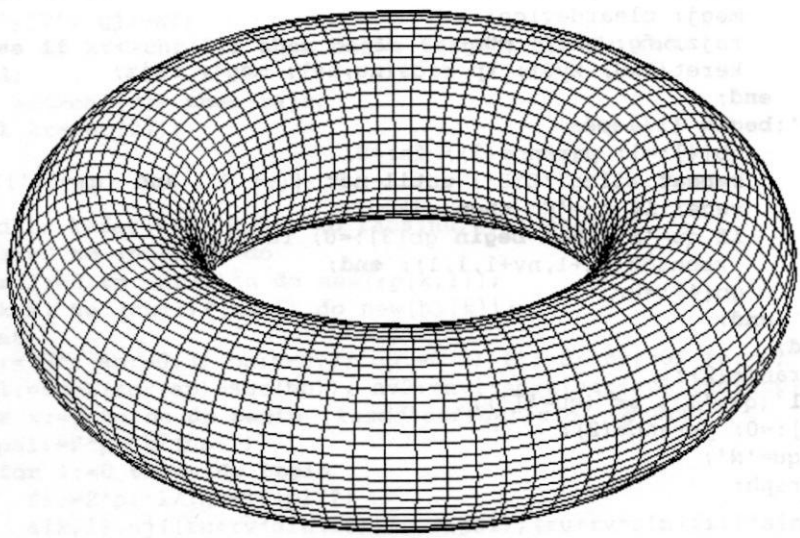
```
      for l:=0 to nv do begin
         folt;
         if k=0 then pl:=0 else p1:=1;
         if l=0 then r1:=0 else r1:=1;
         for p:=p1 to n do
           for r:=r1 to n do begin
             sulyfugg;
             rp[k*n+p,l*n+r]^.uj(xx,yy,zz);
           end;
      end;
    fl:=1; fp:=0;
  end;

begin
  racs; if gb[2]=1 then nyíl; setcolor(15);
  if f1=0 then rajzracs;
  egern; megjelen(nu+1,nv+1,n,0,1,1,fh-1); eger_1;
  fr:=1;
end;

procedure fg; {vetítési irány változtatásának vezérlése)
begin
  repeat
    setcolor(7);
    case qj of
    'D':outtextxy(15,102,'T - tele');
    'T':outtextxy(15,102,'D - drótváz');
    end;
    repeat kr:=upcase(readkey); until kr in valasz_f;
    cleardevice;
    case kr of
    'D','T':qj:=kr;
    else if kr<>chr(27) then begin forgat; fp:=0; end;
    end;
    if kr<>chr(27) then rajz;
  until kr=chr(27);
end;

begin
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to (nu+1)*n do
    for l:=0 to (nv+1)*n do new(rp[k,l]);
  for k:=1 to (nu+1)*(nv+l) do new(hl[k]);
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nu do begin {kezdőkonfiguráció - gyűrű)
      psi:=2*pi*k/(nu+l);
      for l:=0 to nv do begin
        fi:=2*pi*l/(nv+l)+pi/2;
        a[k,1].uj((ru+rv*sin(fi))*cos(psi),(ru+ry*sin(fi))*sin(psi),
        ry*cos(fi));
      end;
    end;
```

```
    qk:='N';
    racs; konfig('ZYB'); egern; parancs(2); racs;
    kpontok(nu+l,nv+l,l,l);
    repeat
      eger_t(kx0,2,620,ky0); eger_1; qr:=' '; b:=0;
      repeat par katt(2,qr); kp r katt(nu,nv); until qr in valasz;
      eger_n;
      case qr of
        'V':begin {kontrollpont kiválasztása és koordinátáinak
                   módosítása}
            gb[3):=0; fe:=0; racs; kpontok(nu+l,nv+1,1,1);
            gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
            inform(szx,szy,'    Hely: '); kp_kor(szx,szy,12);
            koordin3('Hely');
            if qr<>char(27) then a(szx,szy].uj(x,y,z);
            kp_kor(szx,szy,7); racs; kpontok(nu+l,nv+1,1,1);
            gb[1]:=0; lap(208,5); setfillstyle(1,11);
            bar(130,470,590,480);
            fl:=0;
          end;
      'F':begin (felületmegjelenítés}
            gb[3]:=1; parancs(2); fe:=1;
            bspline;
            if ((qr='2') and (fh=2)) or ((qr='3') and (fh=1)) then
              fl:=0;
            if qr='2' then fh:=1 else fh:=2;
            megj; rajz; gb[3]:=0;
          end;
       'E':if fe=1 then begin (felületmegjelenítés keret nélkül)
            gb[2]:=1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb [ 2 ] : =0;
            keret(kx0, ky0); kp_racs(nu,nv); megj; rajz;
          end;
      'Q':begin {kilépés}
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('ZYB') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs;
              kpontok(nu+l,nv+1,1,1); end;
            gb[4]:=0;
          end;
      end;
      parancs(2);
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```

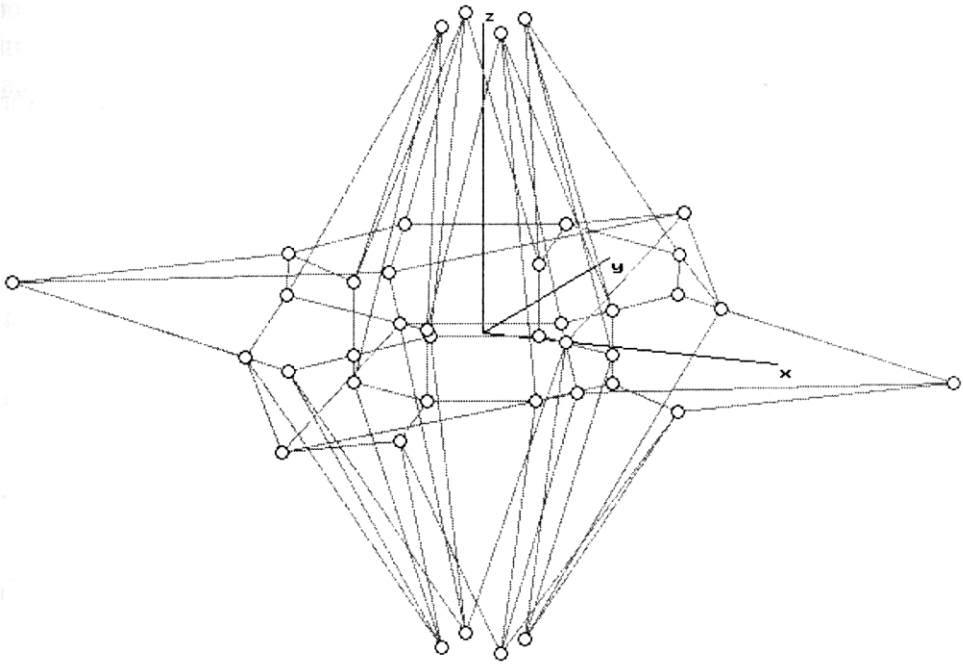
A *3.27. ábra* a program kiindulási kontrollpoliéderét mutatja, a *3.28. ábrán* pedig az erre szerkesztett másodfokú B-spline tórusz látható. Figyelemreméltó a kapott felület szabályossága.



3.27. ábra

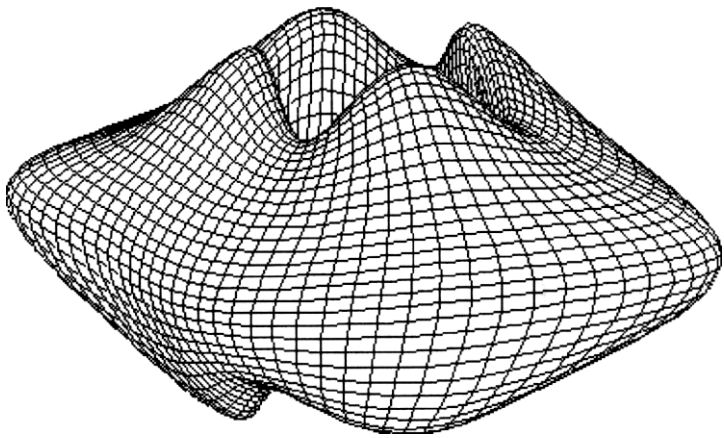


3.28. ábra



3.29. ábra

A 3.29. ábrán látható poliéderre illesztett harmadfokú B-spline felület:



3.30. ábra

3.3.5. β -spline közelítés

Felületek interaktív szerkesztése β -spline közelítéssel a harmadfokú uniform B-spline approximációhoz hasonló előállítással történik:

$$\mathbf{r}(u,v) = \begin{bmatrix} f_0(u) & f_1(u) & f_2(u) & f_3(u) \end{bmatrix} \cdot \mathbf{P} \cdot \begin{bmatrix} f_0(v) & f_1(v) & f_2(v) & f_3(v) \end{bmatrix}^T$$

$$(u, v) \in [0,1] \times [0,1] \qquad (3.41)$$

ahol az f súlyfüggvények alakját a 2.64 képletek adják, P (3.38) pedig a karakterisztikus keret 4x4 kontrollpontból álló darabját (patch) jellemző mátrix.

A paramétertartomány két irányában (a felület u = áll. illetve v = áll. paramétergörbéi mentén) az alakparaméterek - eltolás és feszültség - értékei (β_{1u} , β_{2u} illetve β_{1v} , β_{2v}) különbözhetnek. A β -spline görbékhez hasonlóan, a β -spline felületek esetében is bevezethetők (a paramétergörbék mentén) folytonosan változó alakparaméterek (2.68).

Nyílt felületek esetén a karakterisztikus keretet segédkontrollpontokkal szegélyezhetjük, amelyeket úgy helyezünk el, hogy a kapott patch határgörbéi a keret megfelelő szélén álló kontrollpontok által meghatározott β -spline görbék legyenek. Az

$$\mathbf{F}_u = \begin{bmatrix} f_{0u} & f_{1u} & f_{2u} & f_{3u} \end{bmatrix}$$
$$\mathbf{F}_v = \begin{bmatrix} f_{0v} & f_{1v} & f_{2v} & f_{3v} \end{bmatrix} \qquad (3.42)$$

jelölésekkel a vázolt feltételeket az

$$\mathbf{F}_u(0)\mathbf{P} = \begin{bmatrix} \mathbf{p}_{0,j-1} & \mathbf{p}_{0,j} & \mathbf{p}_{0,j+1} & \mathbf{p}_{0,j+2} \end{bmatrix}$$
$$\mathbf{F}_u(1)\mathbf{P} = \begin{bmatrix} \mathbf{p}_{m,j-1} & \mathbf{p}_{m,j} & \mathbf{p}_{m,j+1} & \mathbf{p}_{m,j+2} \end{bmatrix}; \quad j = 1,\ldots,n-2$$
$$\mathbf{P}\,\mathbf{F}_v(0)^T = \begin{bmatrix} \mathbf{p}_{i-1,0} & \mathbf{p}_{i,0} & \mathbf{p}_{i+1,0} & \mathbf{p}_{i+2,0} \end{bmatrix}^T$$
$$\mathbf{P}\,\mathbf{F}_v(1)^T = \begin{bmatrix} \mathbf{p}_{i-1,n} & \mathbf{p}_{i,n} & \mathbf{p}_{i+1,n} & \mathbf{p}_{i+2,n} \end{bmatrix}^T; \quad i = 1,\ldots,m-2 \qquad \textbf{(3.43)}$$

módon írhatjuk fel, ahol **P** a kontrollpoliéder megfelelő szélső darabjának mátrix reprezentációja. Felírva a 3.43 egyenletek baloldalán álló kifejezéseket és azonosítva a jobboldal megfelelő elemeivel a segédpontokra a

$$\begin{aligned}
\mathbf{p}_{-1,j} &= \mathbf{p}_{0,j} + \frac{\mathbf{p}_{0,j} - \mathbf{p}_{1,j}}{\beta_{1u}^{3}} \qquad ; \quad j = 0,\ldots,n \\
\mathbf{p}_{m+1,j} &= \mathbf{p}_{m,j} + \left(\mathbf{p}_{m,j} - \mathbf{p}_{m-1,j}\right)\beta_{1u}^{3} \\
\mathbf{p}_{i,-1} &= \mathbf{p}_{i,0} + \frac{\mathbf{p}_{i,0} - \mathbf{p}_{i,1}}{\beta_{1v}^{3}} \qquad ; \quad i = 0,\ldots,m \\
\mathbf{p}_{i,n+1} &= \mathbf{p}_{i,n} + \left(\mathbf{p}_{i,n} - \mathbf{p}_{i,n-1}\right)\beta_{1v}^{3}
\end{aligned} \tag{3.44}$$

képleteket kapjuk. Ahhoz, hogy a felület sarkai a kontrollpoliéder sarkain álló pontokba kerüljenek:

(3.46)

$$\begin{aligned}
\mathbf{F}_u(0)\mathbf{P}\mathbf{F}_v(0)^T &= \mathbf{p}_{0,0} \\
\mathbf{F}_u(0)\mathbf{P}\mathbf{F}_v(1)^T &= \mathbf{p}_{0,n} \\
\mathbf{F}_u(1)\mathbf{P}\mathbf{F}_v(0)^T &= \mathbf{p}_{m,0} \\
\mathbf{F}_u(1)\mathbf{P}\mathbf{F}_v(1)^T &= \mathbf{p}_{m,n}
\end{aligned} \tag{3.45}$$

a csúcsokban található segédpontok helyeit

$$\mathbf{p}_{-1,-1} = \frac{\left(\beta_{1u}^{3}+1\right)\left(\beta_{1v}^{3}+1\right)\mathbf{p}_{0,0} - \left(\beta_{1u}^{3}+1\right)\mathbf{p}_{0,1} - \left(\beta_{1v}^{3}+1\right)\mathbf{p}_{1,0} - \mathbf{p}_{1,1}}{\beta_{1u}^{3}\beta_{1v}^{3}}$$

$$\mathbf{p}_{-1,n+1} = \frac{\left(\beta_{1u}^{3}+1\right)\left(\beta_{1v}^{3}+1\right)\mathbf{p}_{0,n} - \left(\beta_{1u}^{3}+1\right)\beta_{1v}^{3}\mathbf{p}_{0,n-1} - \left(\beta_{1v}^{3}+1\right)\mathbf{p}_{1,n} + \beta_{1v}^{3}\mathbf{p}_{1,n-1}}{\beta_{1u}^{3}}$$

$$\mathbf{p}_{m+1,-1} = \frac{\left(\beta_{1u}^{3}+1\right)\left(\beta_{1v}^{3}+1\right)\mathbf{p}_{m,0} - \left(\beta_{1v}^{3}+1\right)\beta_{1u}^{3}\mathbf{p}_{m-1,0} - \left(\beta_{1u}^{3}+1\right)\mathbf{p}_{m,1} + \beta_{1u}^{3}\mathbf{p}_{m-1,1}}{\beta_{1u}^{3}\beta_{1v}^{3}}$$

$$\begin{aligned}
\mathbf{p}_{m+1,n+1} = {} & \left(\beta_{1u}^{3}+1\right)\left(\beta_{1v}^{3}+1\right)\mathbf{p}_{m,n} - \left(\beta_{1u}^{3}+1\right)\beta_{1v}^{3}\mathbf{p}_{m,n-1} - \left(\beta_{1v}^{3}+1\right)\beta_{1u}^{3}\mathbf{p}_{m-1,n} + \\
& + \beta_{1u}^{3}\beta_{1v}^{3}\mathbf{p}_{m-1,n-1}
\end{aligned}$$

képletek adják.

Az F_P_BETS.PAS program teszi lehetővé a karakterisztikus keret és az alakparaméterek interaktív módosítását, a felület megszerkesztését és megjelenítését.

```
{$N+}

program f_p_bets;
    {paraméteres előállítású nyílt felületek β-spline közelítéssel)

uses
 graph, grafind, crt, dos, kozos, kozos 3d, kerdesek, rajz_f3d,
 konf_3d, komm_fel, eger_kez, gomb_kez;

const
 nu=4; nv=4; ax=80; ay=80; n=10;
 valasz: kars = ['E','F','Q','V','A','C'];
 valasz_f: kars = ['D','T', fel, le, jobb, bal, char(27)];

var
 pu0, pu1: array[0..nv] of pont3;
 pv0, pv1: array[0..nu] of pont3;
 p00, p01, p10, p11: pont3;
 bl2u, bl3u, du, bl2v, bl3v, dv, dd: real;
 fc, fdu, fdv: byte;

procedure delta_u;
 begin
   b12u:=b1u*blu; b13u:=b12u*blu; du:=b13u+2*(b12u+blu)+b2u+1;
   fc:=0; fdu:=1;
 end;

procedure delta_v;
 begin
   b12v:=blv*blv; b13v:=b12v*blv; dv:=b13v+2*(b12v+blv)+b2v+1;
   fc:=0; fdv:=1;
 end;

procedure hatar; {segédhatárpontok kiszámítása}
begin
 for k:=0 to nu do begin
   pv0[k].uj((a[k,0].xp-a[k,1].xp)/b13v+a[k,0].xp,
               (a[k,0].yp-a[k,1].yp)/b13v+a[k,0].yp,
           (a[k,0].zp-a[k,1].zp)/b13v+a[k,0].zp);
          pv1[k].uj((a[k,nv].xp-a[k,nv-
          1].xp)*b13v+a[k,nv].xp, (a[k,nv].yp-a[k,nv-
          1].yp)*b13v+a[k,nv].yp, (a[k,nv].zp-a[k,nv-
          1].zp)*b13v+a[k,nv].zp); end;
 for 1:=0 to nv do begin
   pu0[1].uj((a[0,1].xp-a[1,1].xp)/b13u+a[0,1].xp, (a
           [0, 1] . yp-a [1, 1] . yp) /b13u+a [0, 1] .
           yp, (a[0,1].zp-a[1,1].zp)/b13u+a[0,1].zp);
           pu1[1].uj((a[nu,1].xp-a[nu-
```

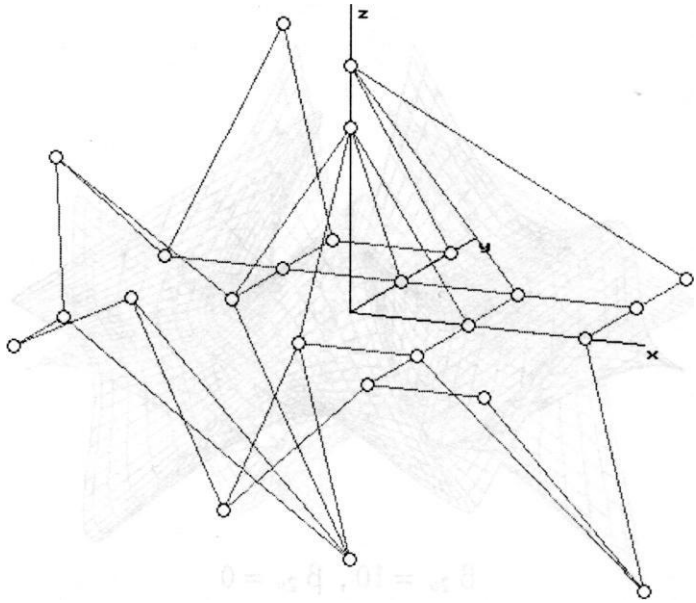
```
                (a[nu,1].yp-a[nu-1,1].yp)*b13u+a[nu,l].yp,
                (a[nu,1].zp-a[nu-1,1].zp)*b13u+a[nu,1].zp);
  end;
  p00.uj((pv0[0].xp-pv0[1].xp)/b13u+pv0[0].xp, {sarokpontok}
           (pv0[0].yp-pv0[1].yp)/b13u+pv0[0].yp, (pv0[0].zp-
          pv0[1].zp)/b13u+pv0[0].zp);
  pOl.uj((pu0[nv].xp-puO[nv-1].xp)*b13v+pu0[nv].xp,
           (pu0[nv].yp-pu0[nv-i].yp)*b13v+pu0[nv].yp,
           (pu0[nv].zp-puO[nv-1].zp)*b13v+puO[nv].zp);
  pl0.uj((pv0[nu].xp-pv0[nu-1].xp)*b13u+pv0[nu].xp,
           (pvO[nu].yp-pv0[nu-1].yp)*b13u+pv0[nu].yp,
           (pv0[nu].zp-pv0[nu-1].zp)*b13u+pv0[nu].zp);
  pll.uj((pul[nv].xp-pul[nv-1].xp)*b13v+pul[nv].xp,
           (pul[nv].yp-pul[nv-1].yp)*b13v+pul[nv].yp,
           (pul[nv].zp-pul[nv-1].zp)*b13v+pul[nv].zp);
  fc:=1;
end;

procedure rajz;
var
  u,u2,u3,v,v2,v3,f1,f2,f3,f4: single;
  kp: array[-1..2,-1..2) of pont3;
  fu, fv: array[-1..2] of single;

  procedure bl_funct_u; {súlyfüggvények u irányban)
  begin
    u:=p/n; u2:=u*u; u3:=u2*u;
    f1:=1-3*u+3*u2-u3; f2:=u2*(3-u); f3:=u*(3-u2); f4:=f2-u3;
    fu[-1]:=bi3u*fl; ( β-spline súlyfüggvények u irányban)
    fu[0]:=b13u*(1-f1)+b12u*(2-f2)+blu*(2-f3)+b2u*(1-f4);
    fu[1]:=b12u*f2+blu*f3+b2u*f4+1-u3;
    fu[2]:=u3;
  end;

  procedure bl_funct_v; {súlyfüggvények v irányban)
  begin
    v:=r/n; v2:=v*v; v3:=v2*v;
    fi:=1-3*v+3*v2-v3; f2:=v2*(3-v); f3:=v*(3-v2); f4:=f2-v3;
    fv[-1]:=b13v*f1; {β-spline súlyfüggvények v irányban)
    fv[0]:=b13v*(1-fl)+b12v*(2-f2)+blv*(2-f3)+b2v*(1-f4);
    fv[1]:=b12v*f2+blv*f3+b2v*f4+1-v3;
    fv[2]:=v3;
  end;

  procedure folt; {felületfoltok (patch) kontrollpontjainak
                    összerendelése}
  var tu, tv: integer;
  begin
    for i:=-1 to 2 do begin
      tu:=k+i;
      for j:=-1 to 2 do begin
        tv:=1+j;
        case tu of
```

```
        -l:case tv of
           -1: kp[i,j]:=p00;
           nv+l: kp[i,j]:=p01;
           else kp[i,j]:=pu0[tv];
           end;
        nu+l:case tv of
           -1: kp[i,j]:=p10;
           nv+l: kp[i,j]:=p11;
           else kp[i,j]:=pul[tv];
           end;
        else case tv of
             -1: kp[i,j]:=pv0[tu];

             nv+l: kp[i,j]:=pvl[tu];
             else kp[i,j]:=a[tu,tv];
             end;
        end;
      end;
    end;
  end;

  function xx: single;
  var s: single;
  begin
    s:=0;
    for i:=-1 to 2 do
      for j:=-1 to 2 do
        s:=s+kp[i,j].xp*fu[i]*fv[j];
    xx:=s/dd;
  end;

  function yy: single;
  var s: single;
  begin
    s:=0;
    for i:=-1 to 2 do
      for j:=-1 to 2 do
        s:=s+kp[i,j].yp*fu[i]*fv[j];
    yy:=s/dd;
  end;

  function zz: single;
  var s: single;
  begin
    s:=0;
    for i:=-1 to 2 do
      for j:=-1 to 2 do
        s:=s+kp[i,J].zp*fu[i]*fv[j];
    zz:=s/dd;
  end;
```

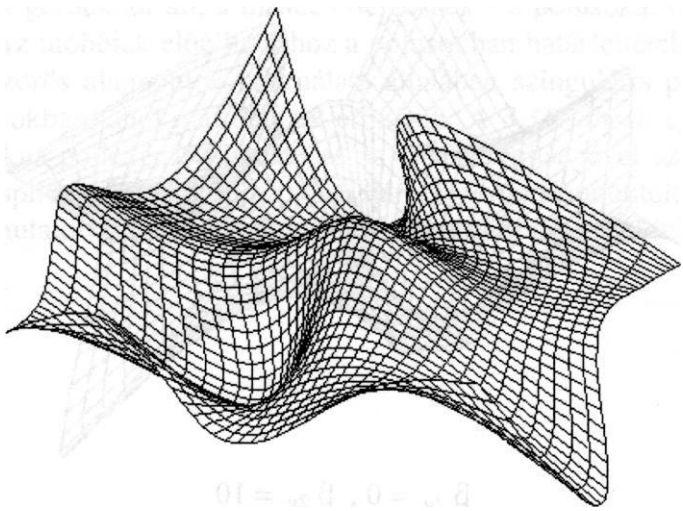
```
  procedure rajzracs; {felületháló csomópontjainak kiszámítása)
  begin
    if fdu=0 then begin
      delta_u;
      if abs(du)<0.1 then begin
        if du<0 then b2u:=b2u-1 else b2u:=b2u+1;
        delta_u; end; (nullaosztás elkerülése)
    end;
    if fdv=0 then begin
      delta_v;
      if abs(dv)<0.1 then begin
        if dv<0 then b2v:=b2v-1 else b2v:=b2v+1;
        delta_v; end; {nullaosztás elkerülése}
    end;
    dd:=du*dv;
    if fc=0 then hatar;
    for p:=0 to n do begin
      bl_funct_u;
      for r:=0 to n do begin
        bl_funct_v;
        for k:=0 to nu-1 do
          for 1:=0 to nv-1 do begin
            folt;
            rp[k*n+p,l*n+r]^.uj(xx,yy,zz);
          end;
      end;
    end;
    fl:=1; fp:=0;
  end;

begin
  racs; if gb[2)=1 then nyil; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu,nv,n,0,0,1); eger_l;
  fr:=1;
end;

procedure fg; (vetítési irány váltortatásának vezérlése)
   ..........

begin
  obj ='F';
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to nu*n do
    for 1:=0 to nv*n do new(rp[k,l]);
  for k:=1 to nu*nv do new(hl[k]);
  repeat
         ..........
    racs; konfig('FBS'); eger_n; parancs(2); bvalt; racs;
    kpontok(nu,nv,0,0);
    repeat
      eger_t(kx0,2,620,ky0); eger_l; qr:=' '; b:=0;
       repeat par_katt(2,qr); kp_r_katt(nu,nv); until qr in valasz;
```

```
      eger_n;
      case qr of
      'V':begin (kontrollpont kiválasztása és koordinátáinak
                 módosítása)
            ..........
      'F':begin {felületmegjelenítés}
            ........
     'E':if fe=1 then begin {felületmegjelenítés keret nélkül}
            ........
      'A':begin {eltolásmódosítás}
            gb[7]:=1; bvalt;
            irany; eltl; if bl<0.05 then bl:=0.05; if bl>20 then
            b1:=20;
            case qd of
            'U':begin blu:=b1; fdu:=0; end;
            'V':begin blv:=bl; fdv:=0; end;
            end;
            gb[7]:=0; lap(208,5); fl:=0;
         end;
      'C':begin {feszültségmódosítás}
            gb[8]:=1; bvalt; irany; fesz;
            case qd of
            'U':begin b2u:=b2; fdu:=0; end;
            'V':begin b2v:=b2; fdv:=0; end;
            end;
            gb[8]:=0; lap(208,5); fl:=0;
         end;
      'Q':begin (kilépés}
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('FBS') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs; kpontok(nu,nv,0,0);
                               end;
            gb[4]:=0;
         end;
      end;
      parancs(2); bvalt;
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```

A feszültség hatását a *3.31. ábrán* látható karakterisztikus kerettel szerkesztett felületek szemléltetik. A *3.32. ábrán* a harmadfokú B-spline felület, a *3.33.- 3.35. ábrákon* pedig különböző β_{2u} és β_{2v} értékekre kapott felületek láthatók $\beta_{1u}=\beta_{1v}=1$).

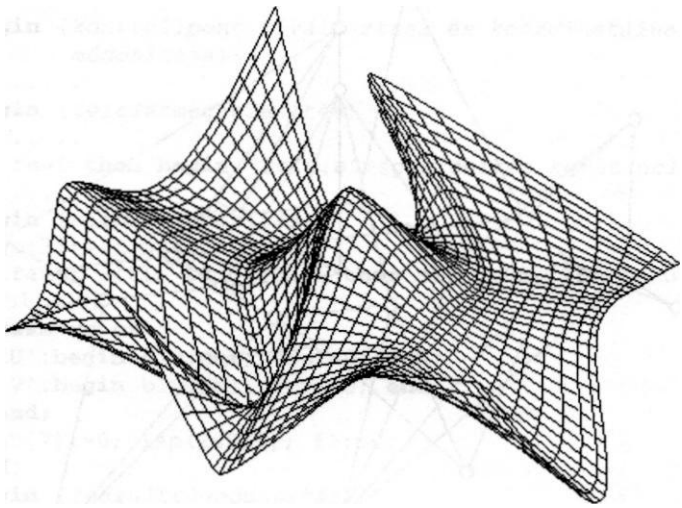


3.31. ábra



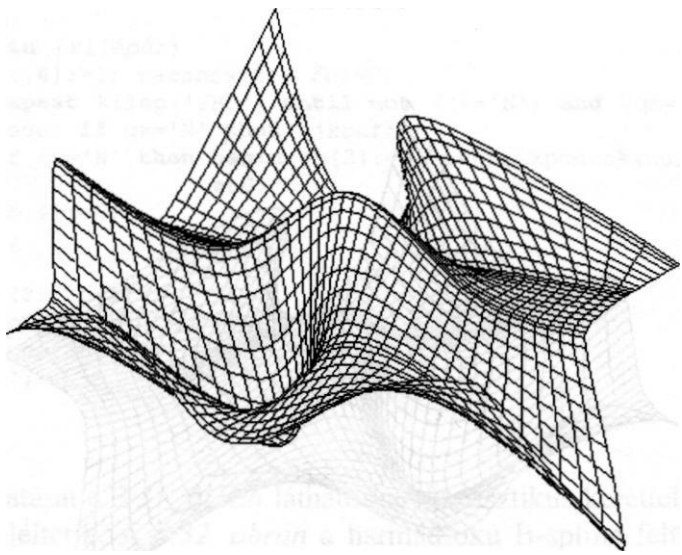
$\beta_{2u} = \beta_{2v} = 0$

3.32. ábra



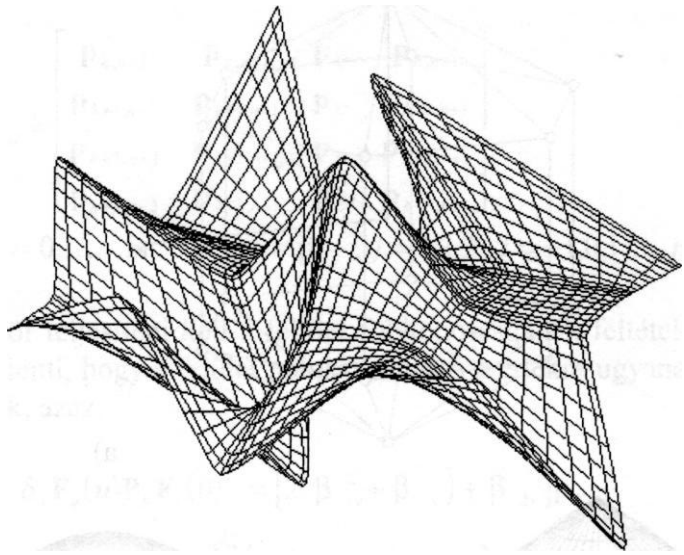
$\beta_{2u} = 10,\ \beta_{2v} = 0$

3.33. ábra



$\beta_{2u} = 0,\ \beta_{2v} = 10$

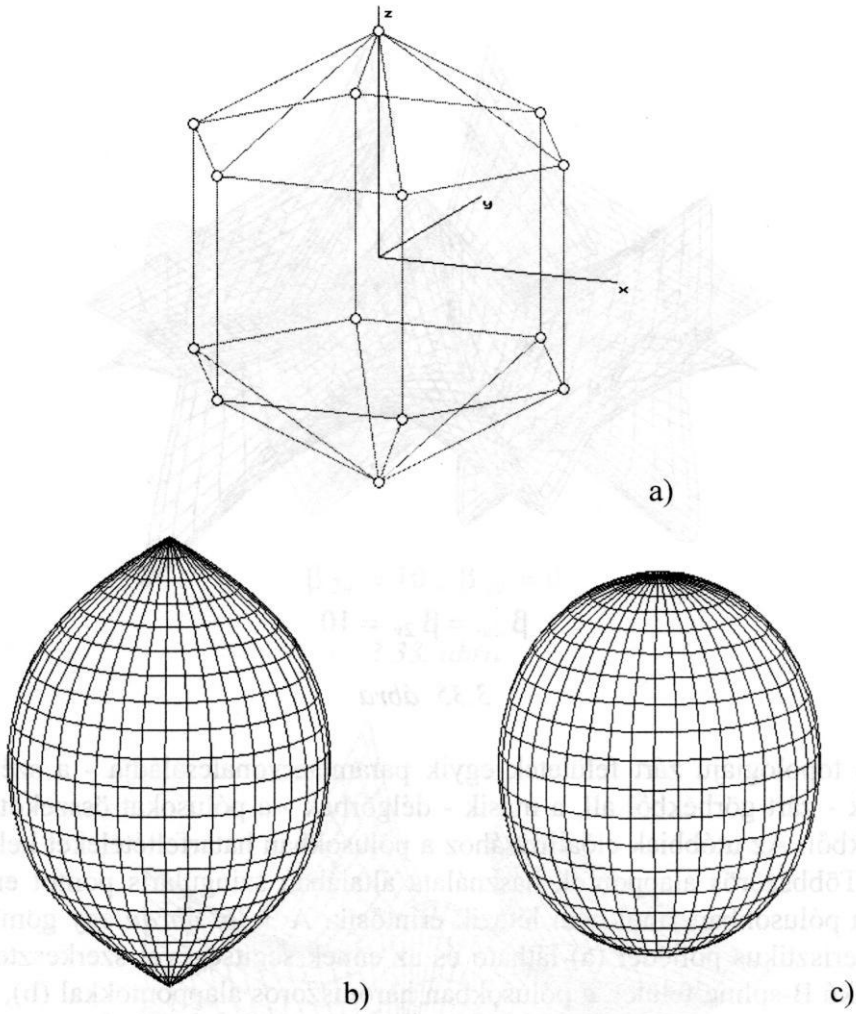
3.34. ábra



$\beta_{2u} = \beta_{2v} = 10$

3.35. ábra

Gömb topológiájú zárt felületek egyik paramétervonalcsaládja - a szélességi görbék - zárt görbékből áll, a másik - délgörbék - a pólusokat összekötő nyílt görbékből. Az utóbbiak előállításához a pólusokban határfeltételeket kell megadni. Többszörös alappontok használata általában szinguláris pontot eredményez a pólusokban, ahol nem létezik érintősík. A *3.36. ábrán* egy gömbbe írt karakterisztikus poliéder (a) látható és az ennek segítségével szerkesztett harmadfokú B-spline felület, a pólusokban háromszoros alappontokkal (b), illetve az alább bemutatott módszerrel meghatározott segédpontokkal (c).



3.36. ábra

Tekintsük a karakterisztikus poliédernek az északi, illetve déli pólus környezetében elhelyezkedő

$$\mathbf{P}_E = \begin{bmatrix} \mathbf{p}_{k,-1} & \mathbf{p}_E & \mathbf{p}_{k,1} & \mathbf{p}_{k,2} \\ \mathbf{p}_{k+1,-1} & \mathbf{p}_E & \mathbf{p}_{k+1,1} & \mathbf{p}_{k+1,2} \\ \mathbf{p}_{k+2,-1} & \mathbf{p}_E & \mathbf{p}_{k+2,1} & \mathbf{p}_{k+2,2} \\ \mathbf{p}_{k+3,-1} & \mathbf{p}_E & \mathbf{p}_{k+3,1} & \mathbf{p}_{k+3,2} \end{bmatrix} \qquad (3.4$$

illetv

$$\mathbf{P}_D = \begin{bmatrix} \mathbf{p}_{k,n-2} & \mathbf{p}_{k,n-1} & \mathbf{p}_D & \mathbf{p}_{k,n+1} \\ \mathbf{p}_{k+1,n-2} & \mathbf{p}_{k+1,n-1} & \mathbf{p}_D & \mathbf{p}_{k+1,n+1} \\ \mathbf{p}_{k+2,n-2} & \mathbf{p}_{k+2,n-1} & \mathbf{p}_D & \mathbf{p}_{k+2,n+1} \\ \mathbf{p}_{k+3,n-2} & \mathbf{p}_{k+3,n-1} & \mathbf{p}_D & \mathbf{p}_{k+3,n+1} \end{bmatrix} \qquad (3.48$$

$$k = 0,1,\ldots,m\,; \quad \mathbf{p}_{m+j,l} = \mathbf{p}_{j,l}\,,\ j = 1,2,3\,,\ l = -1,0,1,\ldots,n,n+1$$

foltjait. Először teljesíteni kell a felület folytonossá$_g$ának feltételét a pólusokban. Ez azt jelenti, hogy a felület pólusai minden k értékre u$_g$yanabba a pontba kell kerüljenek, azaz:

$$\delta_v \mathbf{F}_u(u)\mathbf{P}_E\mathbf{F}_v(0)^T = \left[2\left(\beta_{1v}^{\,2} + \beta_{1v}\right) + \beta_{2v}\right]\mathbf{p}_E + \\ + \sum_{i=0}^{3}\left(\beta_{1v}^{\,3}\mathbf{p}_{k+i,-1} + \mathbf{p}_{k+i,1}\right) f_i(u) \neq f(u) \qquad , \qquad (3.49)$$

illet

$$\delta_v \mathbf{F}_u(u)\mathbf{P}_D\mathbf{F}_v(1)^T = \left[2\left(\beta_{1v}^{\,2} + \beta_{1v}\right) + \beta_{2v}\right]\mathbf{p}_D + \\ + \sum_{i=0}^{3}\left(\beta_{1v}^{\,3}\mathbf{p}_{k+i,n-1} + \mathbf{p}_{k+i,n+1}\right) f_i(u) \neq f(u)\,, \qquad (3.50$$

azaz a pólus helye nem függhet az u paraméter értékétől, minden délgörbe azonos végpontokkal kell rendelkezzen. Mivel a β -spline súlyfüggvények az egység felosztását képezik (vagyis összegük mindig 1), belátható, hogy a 3.49 és 3.50 feltételek akkor teljesülnek, ha ezek együtthatói a megfelelő összegekben állandóak, ahonnan a segédpontokra:

$$\mathbf{p}_{k,-1} = \frac{\mathbf{c}_E - \mathbf{p}_{k,1}}{\beta_{1v}^{\,3}} \quad ; \quad k = 0,1,\ldots,m \quad . \qquad (3.51$$
$$\mathbf{p}_{k,n+1} = \mathbf{c}_D - \beta_{1v}^{\,3}\mathbf{p}_{k,1}$$

Az állandó vektorok értékeit úgy határozzuk meg, hogy - ha lehetséges - a délgörbék érintővektorai ezek végpontjaiban (azaz a pólusokban) egy síkban legyenek. A felület előállítása az északi pólus közelében:

$$\mathbf{r}_E(u,v) = \mathbf{p}_E f_1(v) + \sum_{\substack{j=0 \\ j\neq 1}}^{3} g_j(u) f_j(v)$$
$$g_j(u) = \sum_{i=0}^{3} \mathbf{p}_{k+i,j} f_i(u) \quad , \tag{3.52}$$

a déli pólus környezetében pedig:

$$\mathbf{r}_D(u,v) = \mathbf{p}_D f_2(v) + \sum_{\substack{j=0 \\ j\neq 2}}^{3} g_{n+j-2}(u) f_j(v)$$
$$g_{n+j-2}(u) = \sum_{i=0}^{3} \mathbf{p}_{k+i,n+j-2} f_i(u) \quad . \tag{3.53}$$

A 3.51 képletekből következik:

$$g_0(u) = \frac{\mathbf{c}_E - g_1(u)}{\beta_{1v}^{3}}$$
$$g_{n+1}(u) = \mathbf{c}_D - \beta_{1v}^{3} g_{n-1}(u) \quad ; \quad k = 0,1,\ldots,m \quad . \tag{3.54}$$

A 3.52, illetve 3.53 előállítások v szerinti parciális deriváltjai adják a délgörbék érintővektorait a pólusokban (v = 0 -ra). A délgörbék súlyfüggvényeit v szerint deriválva 3.52 és 3.53 -ból, 3.54-et figyelembe véve kapjuk:

$$\frac{\delta_v}{3}\frac{\partial}{\partial v}\mathbf{r}_E(u,v)\Big|_{v=0} = -\mathbf{c}_E + \left(\beta_{1v}+1\right) g_1(u) + \beta_{1v}\left(\beta_{1v}^{2}-1\right)\mathbf{p}_E$$
$$\frac{\delta_v}{3}\frac{\partial}{\partial v}\mathbf{r}_D(u,v)\Big|_{v=1} = \mathbf{c}_D - \beta_{1v}^{2}\left(\beta_{1v}+1\right) g_{n-1}(u) + \left(\beta_{1v}^{2}-1\right)\mathbf{p}_D \quad . \tag{3.55}$$

Annak - egyik, számunkra jól kezelhető - elégséges feltétele, hogy a fenti képletek jobboldala ugyanabban a síkban fekvő vektor legyen bármely *u* értékre az, hogy a pólusokkal szomszédos kontrollpontok egy síkban legyenek és a kere-

sett állandó vektoroknak a pólus helyvektorát tartalmazó szabadtaggal képezett összege is ebben a síkban legyen. Erre egy megoldás:

$$\mathbf{c}_E = \frac{\beta_{v1}+1}{m+1}\sum_{k=0}^{m}\mathbf{p}_{k,1} + \beta_{1v}\left(\beta_{1v}^2-1\right)\mathbf{p}_E$$
$$\mathbf{c}_D = \beta_{1v}^2\frac{\beta_{v1}+1}{m+1}\sum_{k=0}^{m}\mathbf{p}_{k,n-1} - \left(\beta_{1v}^2-1\right)\mathbf{p}_D \quad , \qquad (3.56$$

ahol a jobboldal első tagja az illető pólussal szomszédos kontrollpontok által alkotott sokszög súlypontja (szorozva egy, a v irányú eltolástól függő állandóval). A 3.51 képletek az állandóvektorok 3.56-tal számított értékeivel adják a keresett segédpontokat a délgörbék végein.

A bemutatott módszert valósítja meg a ZM_BETSP.PAS program.

```
{$N+}

program zm_betsp;
      {gömb topológiájú zárt felületek β-spline közelítéssel}

uses
  graph, grafind, crt, dos, kozos, kozos 3d, kerdesek, rajz_f3d,
  konf 3d, korom fel, eger kez, gomb kez;

const
  nu=7; nv=4; rg=160; n=10;
  valasz: kars = ['E','F','Q','V','A','C'];
  valasz f: kars = ['D','T', fel, le, jobb, bal, char(27)];

var
  fi, psi: single;
  pv0, pvl: array[0..nu] of pont3;
  pb0, pbl: pont3;
  bl2u, bl3u, du, bl2v, bl3v, dv, dd: real;
  fc, fdu, fdv: byte;
  bnl, bsl, bn2, bs2: real;

procedure delta_u;
  begin
    bl2u:=b1u*blu; bl3u:=b12u*blu; du:=b13u+2*(b12u+btu)+b2u+1;
    fc:=0; fdu:=1; end;

procedure delta_v;
  begin
    bl2v:=blv*blv; bl3v:=b12v*blv; dv:=b13v+2*(b12v+blv)+b2v+1;
    bnl:=(blv+l)/(nu+l); bs1:=b12v*bnl; bs2:=1-b12v; bn2:=-blv*bs2;
    fc:=0; fdv:=1; end;
```

```
procedure hatar; (segédhatárpontok kiszámítása a meridiángörbék
                  végein)
begin
 pb0.uj (0,0,0); pbl.uj
 (0,0,0); for k:=0 to nu do
 begin
   pb0.uj(pb0.xp+a[k,1].xp,pb0.yp+a[k,1].yp,pb0.zp+a[k,1].zp);
      pbl.uj(pbl.xp+a[k,nv-1].xp,pbl.yp+a[k,nv-1].yp,pbl.zp+
      a[k,nv-1].zp);
 end;
 pb0.uj(bnl*pb0.xp+bn2*a[0,0].xp,bnl*pb0.yp+bn2*a[0,0].yp,
        bnl*pb0.zp+bn2*a[0,0].zp);
 pbl.uj(bsl*pbl.xp+bs2*a[0,nv].xp,bsl*pbl.yp+bs2*a[0,nv].yp,
     bsl*pbl.zp+bs2*a[0,nv].zp); for k:=0 to nu do begin
   pv0[k].uj((pb0.xp-a[k,l].xp)/b13v,(pb0.yp-a[k,l].yp)/b13v,
             (pb0.zp-a[k,l].zp)/b13v);
   pvl[k].uj(pbl.xp-b13v*a[k,nv-1].xp,pbl.yp-b13v*a[k,nv-1].yp,
             pbl.zp-b13v*a[k,nv-1].zp);
 end; fc:=1;
end;

procedure rajz;
var
 u,u2,u3,v,v2,v3,fl,f2,f3,f4: single;
 kp: array[-1..2,-1..2] of pont3;
 fu, fv: array[-1..2] of single;

 procedure bl_funct_u; {súlyfüggvények u irányban - szélességi görbék
                        mentén)
 begin
   u:=p/n; u2:=u*u; u3:=u2*u;
   f1:=1-3*u+3*u2-u3; f2:=u2*(3-u); f3:=u*(3-u2); f4:=f2-u3;
   fu[-1]:=b13u*fl; (β-spline súlyfüggvények u irányban)
   fu[0]:=b13u*(1-fl)+b12u*(2-f2)+blu*(2-f3)+b2u*(1-f4);
   fu[1]:=b12u*f2+blu*f3+b2u*f4+1-u3; fu[2]:=u3;
 end;

 procedure bl_funct_v; (súlyfüggvények v irányban - délgörbék mentén]
 begin
   v:=r/n; v2:=v*v; v3:=v2*v;
   fl:=1-3*v+3*v2-v3; f2:=v2*(3-v); f3:=v*(3-v2); f4:=f2-v3;
   fv[-1]:=b13v*f1; (β-spline súlyfüggvények v irányban)
   fv[0]:=b13v*(1-fl)+b12v*(2-f2)+blv*(2-f3)+b2v*(1-f4);
   fv[1]:=b12v*f2+blv*f3+b2v*f4+1-v3; fv[2]:=v3;
 end;

 procedure folt; {felületfoltok (patch) kontrollpontjainak
                  összerendelése)
 var tu, tv: integer;
 begin
   for i:=-1 to 2 do begin
     tu:=k+i;
     for j:=-1 to 2 do begin
```

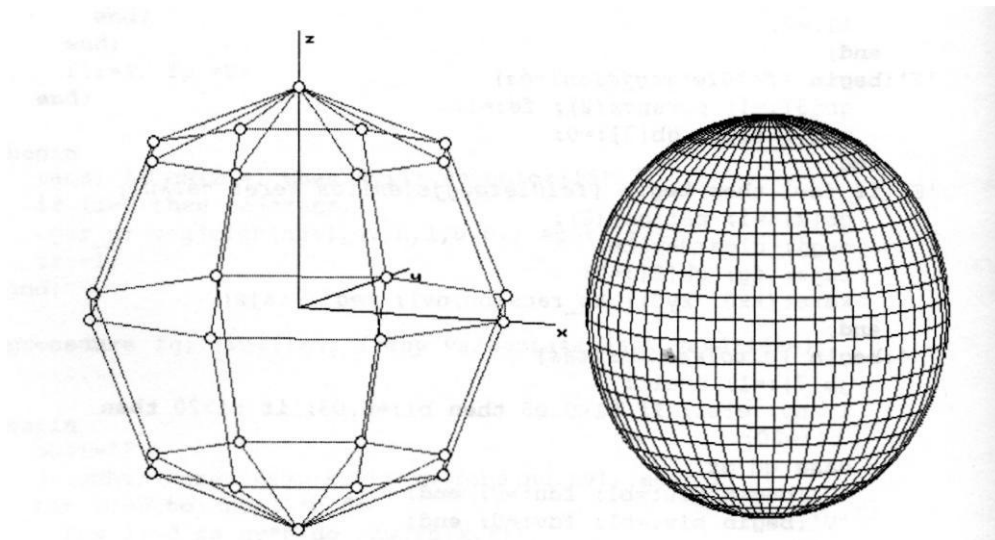
```
      tv:=1+j; case
      to of -
      l:case tv of
         -1: kp[i,j]:=pv0[nu];
         nv+l: kp[i,j]:=pvl[nu];
         else kp[i,j]:=a[nu,tv];
         end;
      nu+l:case tv of
         -1: kp[i,j]:=pv0[0];
         nv+l: kp[i,j]:=pv1[0];
         else kp[i,j]:=a[0,tv];
         end;
      nu+2:case tv of
         -1: kp[i,j):=pv0[1];
         nv+l: kp(i,j):=pv1[1];
         else
         kp[i,j]:=a[l,tv];
         end;
      else case tv of
           -1: kp[i,j]:=pv0[tu];
           nv+l: kp[i,j]:=pvl[tu];
           else kp[i,j]:=a[tu,tv);
           end;
      end;
    end;
  end;
end;

function xx: single;
var s: single;
begin
  s:=0;
   for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].xp*fu[i]*fv[j];
  xx:=s/dd;
end;

function yy: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].yp*fu[i]*fv[j];
  yy:=s/dd;
end;

function zz: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
```

```
        s:=s+kp[i,j].zp*fu[i]*fv[j];
    zz:=s/dd;
  end;

  procedure rajzracs; {felületháló csomópontjainak kiszámítása}
  begin
    if fdu=0 then begin
      delta_u;
      if abs(du)<0.1 then begin
        if du<0 then b2u:=b2u-1 else b2u:=b2u+i;
        delta_u; end; {nullaosztás elkerülése}
    end;
    if fdv=0 then begin
      delta_v;
      if abs(dv)<0.1 then begin
        if dv<0 then b2v:=b2v-1 else b2v:=b2v+1;
        delta_v; end; {nullaosztás elkerülése}
    end;
    dd:=du*dv;
    if fc=0 then hatar;
    for p:=0 to n do begin
      bl_funct_u;
      for r:=0 to n do begin
        bl_funct_v;
        for k:=0 to nu do
           for l:=0 to nv-1 do begin
            folt;
            rp[k*n+p,l*n+r]^.uj(xx,yy,zz);
          end;
      end;
    end;
    fl:=1; fp:=0;
  end;

begin
  racs; if gb[2]=1 then nyil; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu+l,nv,n,1,0,0); eger_l;
  fr:=1;
end;

procedure fg; (vetítési irány váltortatásának vezérlése)
   ........

begin
  obj ='F';
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to (nu+1)*n do
    for l:=0 to nv*n do new(rp[k,l]);
  for k:=1 to (nu+l) *nv do new (hl [k]) ;
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2;
    fl:=0; fr:=0; fp:=0; fe:=0; fc:=0; fdu:=0; fdv:=0;
```

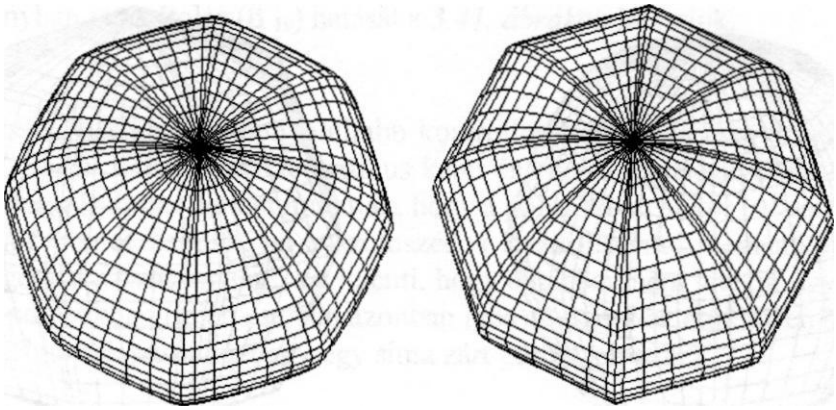
```
nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
for k:=0 to nu do begin {kezdőkonfiguráció - gömb}
  psi:=2*pi*k/(nu+1);
  for 1:=0 to nv do begin
    fi:=pi*1/nv;
    a [k, 1] .uj (rg*sin (fi) *cos (psi) , rg*sin (fi) *sin (psi) , rg*cos (fi)) ;
  end;
end;
qk:='N';
racs; konfig('ZMB'); eger_n; parancs(2); bvalt; racs;
kpontok(nu,nv,0,0);
repeat
  eger_t(kx0,2,620,ky0); eger_1; qr:=' '; b:=0;
  repeat par_katt(2,qr); kp_r_katt(nu,nv); until qr in valasz;
  eger_n;
  case qr of
  'V':begin {kontrollpont kiválasztása és koordinátáinak
             módosítása}
        gb[3]:=0; fe:=0; racs; kpontok(nu+1,nv,1,0);
        gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
        inform(szx,szy,'    Hely: '); kp_kor(szx,szy,12);
        koordin3('Hely');
        if qr<>char(27) then
          if (szy=0) or (szy=nv) then for k:=0 to nu do
                                    a[k,szy] .uj (x,y,z)
                                  else a [szx, szy] . uj (x, y, z) ;
        kp_kor(szx,szy,7); racs; kpontok(nu+1,nv,1,0);
        gb[1]:=0; lap(208,5); setfillstyle(1,11);
        bar(130,470,590,480);
        fl:=0;
      end;
  'F':begin {felületmegjelenítés}
        gb[3]:=1; parancs(2); fe:=1;
        megj; rajz; gb[3]:=0;
      end;
  'E':if fe=1 then begin {felületmegjelenítés keret nélkül}
        gb[2]:=1; parancs(2);
        megj; cleardevice;
        rajz; fg; gb[2]:=0;
        keret(kx0, ky0) ; kp_racs (nu, nv) ; megj; rajz;
      end;
  'A':begin {eltolásmódosítás}
        gb[7]:=1; bvalt;
        irany; eltl; if bl<0.05 then bl:=0.05; if bl>20 then
        bl:=20;
        case qd of
        'U':begin blu:=b1; fdu:=0; end;
        'V':begin blv:=b1; fdv:=0; end;
        end;
        gb[7]:=0; lap(208,5); fl:=0;
      end;
  'C':begin {feszültségmódosítás}
```

```
            gb[8]:=1; bvalt; irany; fesz;
            case qd of
            'U':begin b2u:=b2; fdu:=0; end;
            'V':begin b2v:=b2; fdv:=0; end;
            end;
            gb(8):=0; lap(208,5);
            fl:=0;
          end;
      'Q':begin {kilépés}
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('ZMB') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs; kpontok(nu,nv,0,0);
                               end;
            gb[4]:=0;
          end;
      end;
      parancs(2); bvalt;
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```

A különböző alakparamétereknek a felületre gyakorolt hatását a *3.37.* a) ábrán látható, gömbbe írt, karakterisztikus poliéderrel szerkesztett zárt felületek szemléltetik.



3.37. ábra

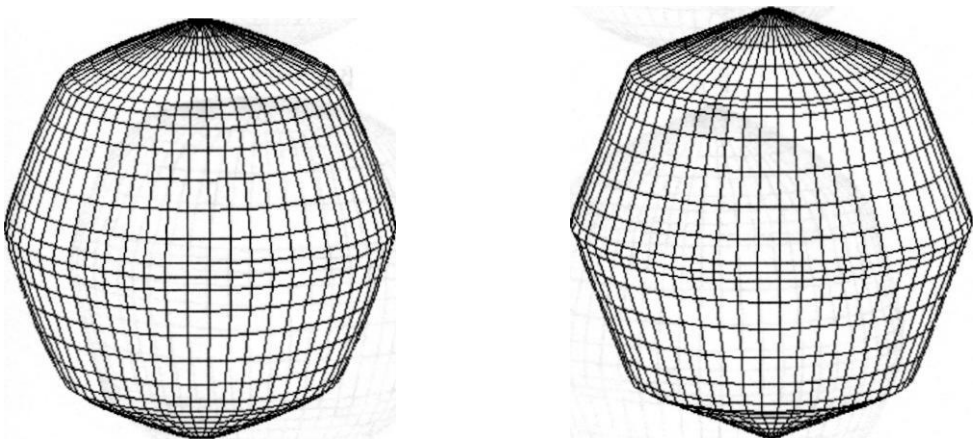


$\beta_{2u}=10$ $\beta_{2u} = 100$

3.38. ábra

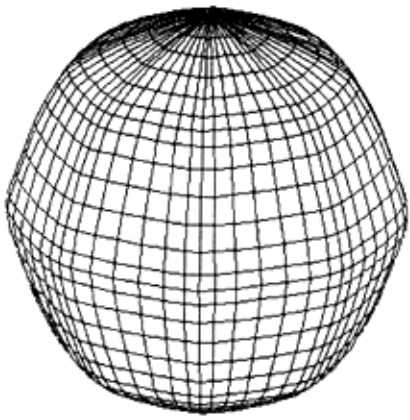
A *3.37.b) ábrán* a harmadfokú B-spline (β_i,' β_{lv}= 1, $\beta_{2u}=\beta_{2v}$=5) felület látható, a *3.38. ábrán* β -spline felületek *a* irányban pozitív feszültséggel, a *3.39. ábrán* β -spline felületek v irányban pozitív feszültséggel, a *3.40. ábrán* pedig β -spline felületek láthatók *u* és v irányban pozitív feszültséggel.

v

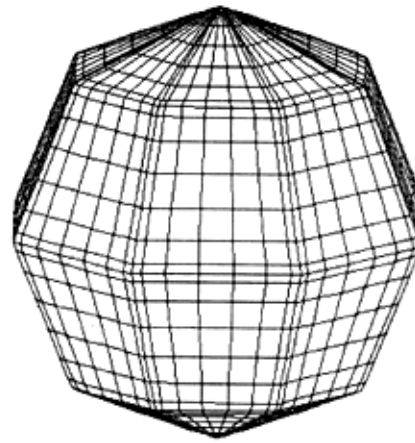


β_{2v}= 10 β_{2v} = 100

3.39. ábra

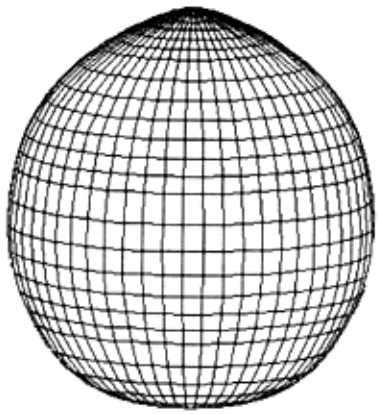


$\beta_{2u} = \beta_{2v} = 5$

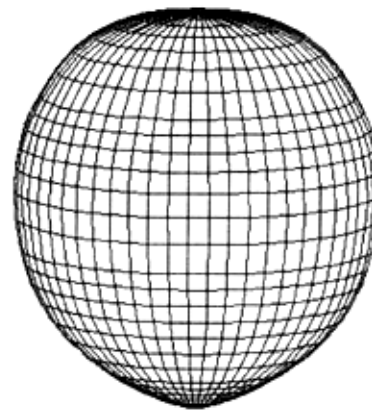


$\beta_{2u} = \beta_{2v} = 100$

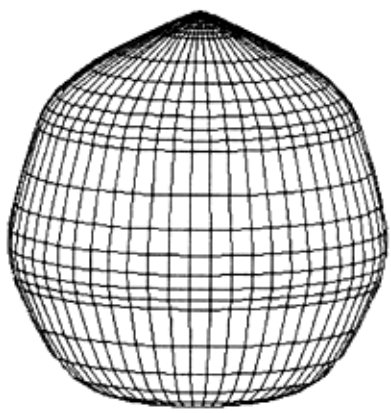
3.40. ábra



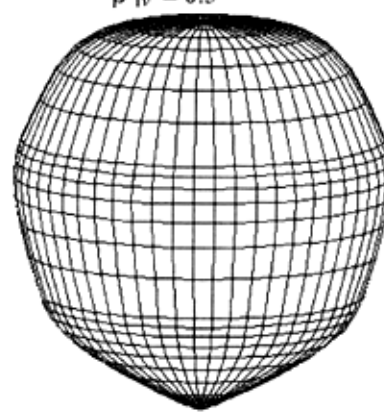
$\beta_{1v} = 2$



$\beta_{1v} = 0.5$



$\beta_{1v} = 5$

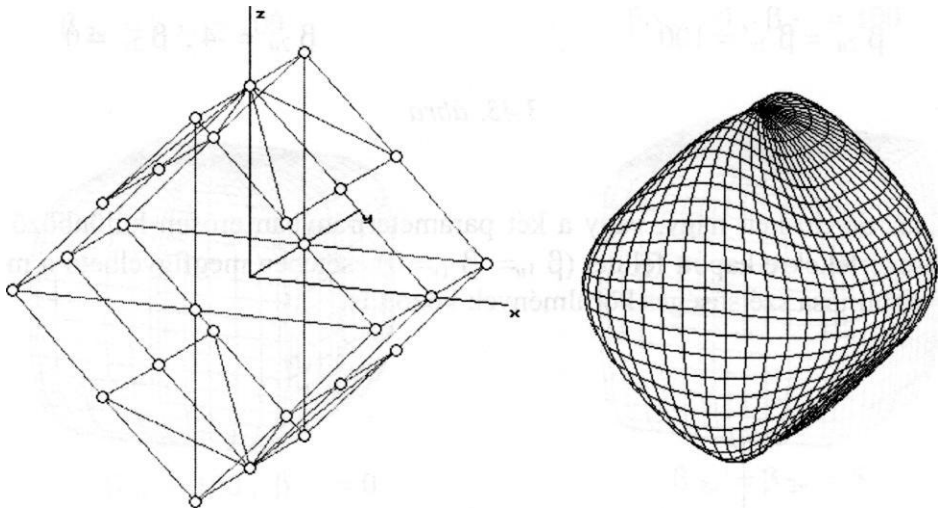


$\beta_{1v} = 0.2$

3.41. ábra

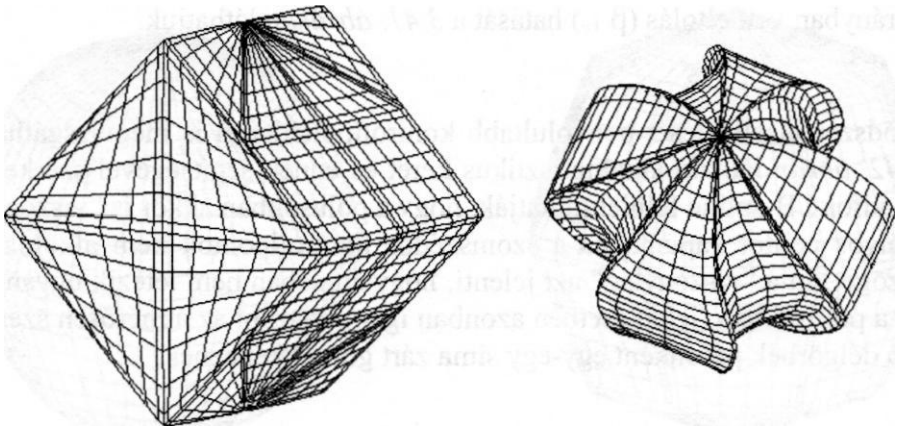
A v irányban vett eltolás (β_{lv}) hatását a *3.41. ábrákon* láthatjuk.

A módszer viselkedését bonyolultabb kontrollpoliéderrel is megvizsgálhatjuk. A *3.42. ábrán* látható karakterisztikus keret és ennek segítségével szerkesztett harmadfokú B-spline felület mutatják, hogy a pólusokban akkor is "viszonylag" szabályos pontot kapunk, ha a szomszédos kontrollpontok nem alkotnak sík sokszöget. Itt a "viszonylag" azt jelenti, hogy általában nem létezik ugyan érintősík a pólusokban (jelen esetben azonban igen), viszont az átmérősen szemben fekvő délgörbék páronként egy-egy sima zárt görbét képeznek.



a) b)

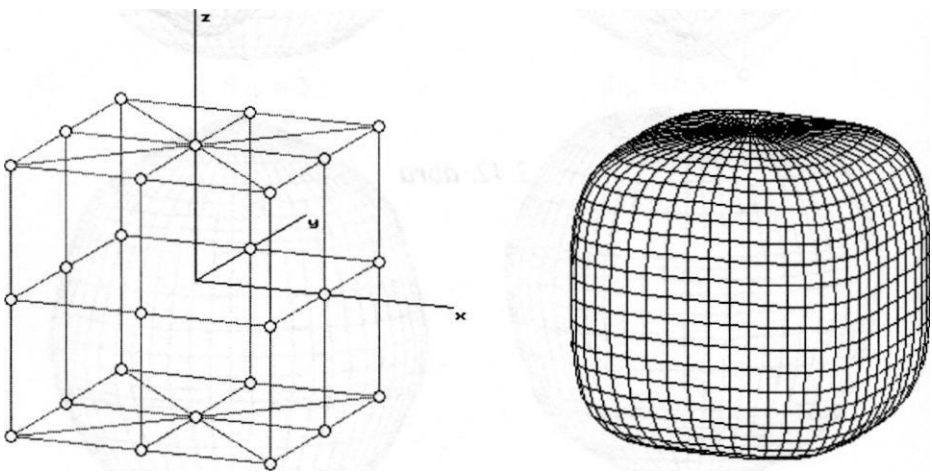
3.42. ábra



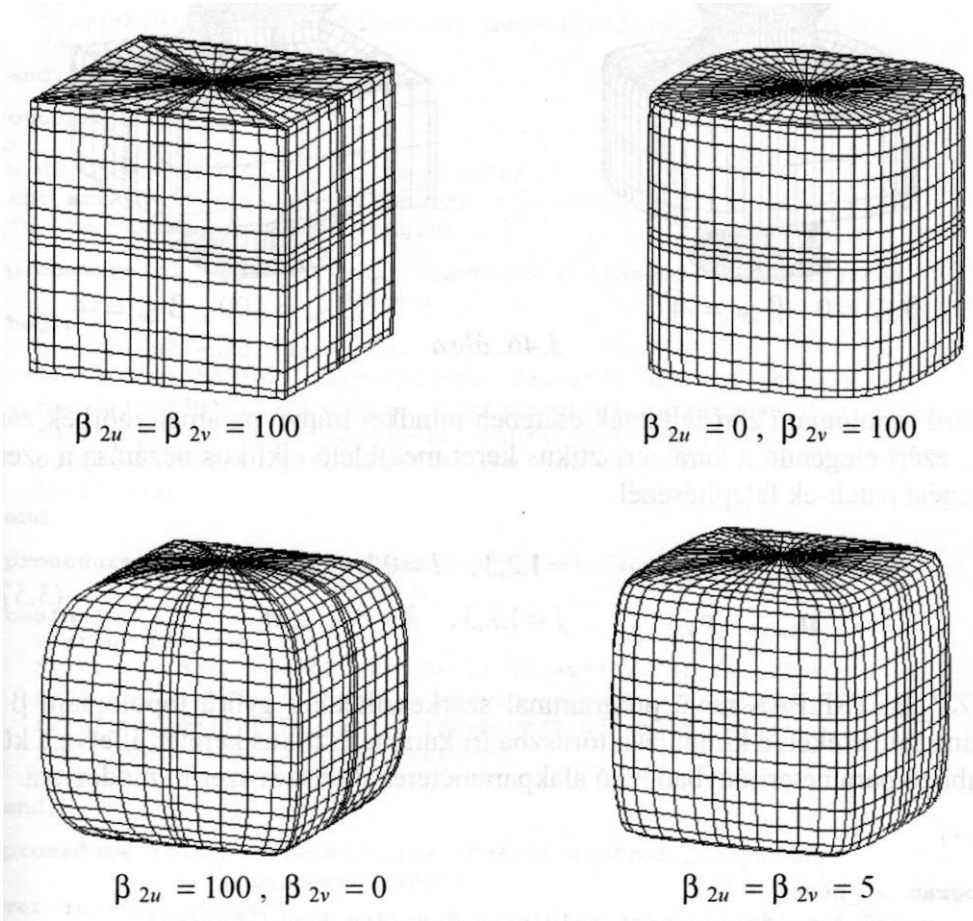
$\beta_{2u}=\beta_{2v}=100$ $\beta_{2u}=-4,\ \beta_{2v}=0$

3.43. ábra

A *3.43. ábrán* két, nagy, vagy a két paraméterirányban erősen különböző feszültségértékekre kapott felület ($\beta_{2u}=\beta_{2v}=0$) esetében megfigyelhető a módszer stabilitása szélsőséges körülmények között is.

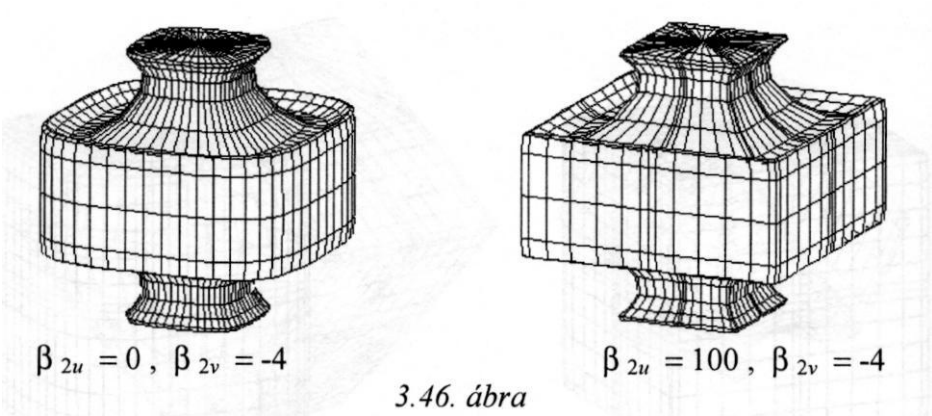


3.44. ábra



3.45. ábra

A *3.44.a) ábrán* látható karakterisztikus keret esetében a pólusokkal szomszédos kontrollpontok egy síkban fekszenek. A *3.44.b) ábrán* a megfelelő harmadfokú B -splinefelület, a *3.45. ábrán* pedig β -spline felületek láthatóak, pozitív feszültségértékekkel. Negatív feszültséggel a karakterisztikus poliédertől nagymértékben eltérő felületeket is kaphatunk *(3.46. ábra).* Arra is figyelni kell, hogy bizonyos (a kontrollpoliéder alakjától és az alakparaméterek értékétől füg- gő) körülmények között önátható felületet is kaphatunk, ami legtöbb esetben elkerülendő.



3.46. ábra

Gyűrű topológiájú zárt felületek esetében mindkét irányú paramétergörbék zártak, ezért elegendő a karakterisztikus keret megfelelő ciklikus bezárása a szerkesztési patch-ek felépítésénél:

$$\mathbf{p}_{m+i,l} = \mathbf{p}_{i,l} \quad ; \quad i = 1,2,3\,, \quad l = 0,1,\ldots,n$$
$$\mathbf{p}_{k,n+j} = \mathbf{p}_{k,j} \quad ; \quad j = 1,2,3\,, \quad k = 0,1,\ldots,m \qquad (3.5$$

A ZY_BETSP.PAS nevű programmal szerkeszthetünk gyűrű topológiájú β - spline felületeket, a kiindulási, tóruszba írt karakterisztikus keretet, illetve a különböző paraméterirányban ható alakparamétereket tetszés szerint módosítva.

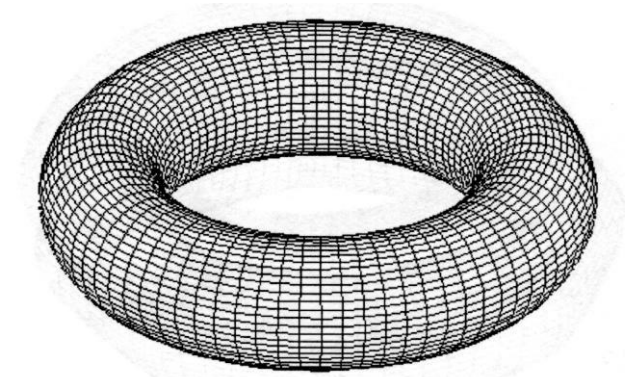
```
($N+)

program zy_betsp;
    (gyűrű topológiájú zárt felületek β-spline közelítéssel)
uses
  graph, grafind, crt, dos, kozos, kozos_3d, kerdesek,
  rajz_f3d, konf 3d, komm_fel, eger kez, gomb kez;
const
  nu=7; nv=4; ru=160; ry=60; n=10;
  v a l a s z :    k a r s    =
  valasz_f: kars = ['D','T', fel, le, jobb, bal, char(27)];
var
  fi, psi: single;
  bl2u, bl3u, du, bl2v, bl3v, dv, dd: real;
  fdu, fdv: byte;

procedure delta_u;
  begin
    bl2u:=blu*blu; bl3u:=b12u*blu; du:=b13u+2*(b12u+blu)+b2u+1;
    fdu:=1;
  end;
```

```
procedure delta_v;
  begin
    bl2v:=blv*blv; bl3v:=b12v*blv; dv:=b13v+2*(b12v+blv)+b2v+1;
    fdv:=1;
  end;

procedure rajz;
var
  u,u2,u3,v,v2,v3,f1,f2,f3,f4: single;
  kp: array[-1..2,-1..2] of pont3;
  fu, fv: array[-1..2] of single;

  procedure bl_funct_u; {súlyfüggvények u irányban - pályagörbék
                         mentén)
  begin
    u:=p/n; u2:=u*u; u3:=u2*u;
    f1:=1-3*u+3*u2-u3; f2:=u2*(3-u); f3:=u*(3-u2); f4:=f2-
    u3; fu[-1]:=b13u*f1; (β-spline súlyfüggvények u
    irányban) fu[0]:=b13u*(1-f1)+b12u*(2-f2)+blu*(2-
    f3)+b2u*(1-f4); fu[1]:=b12u*f2+blu*f3+b2u*f4+1-u3;
    fu[2]:=u3;
  end;

  procedure bl_funct_v; (súlyfüggvények v irányban - leírógörbék
                         mentén)
  begin
    v:=r/n; v2:=v*v; v3:=v2*v;
    f1:=1-3*v+3*v2-v3; f2:=v2*(3-v); f3:=v*(3-v2); f4:=f2-
    v3; fv[-1]:=b13v*f1; ( β-spline súlyfüggvények v
    irányban) fv[0]:=b13v*(1-f1)+b12v*(2-f2)+blv*(2-
    f3)+b2v*(1-f4); fv[1]:=b12v*f2+blv*f3+b2v*f4+1-v3;
    fv[2]:=v3;
  end;

  procedure folt; {felületfoltok (patch) kontrollpontjainak
                   összerendelése}
  var tu, tv: integer;
  begin
    for i:=-1 to 2 do begin
      tu:=k+i;
      case tu of
      -1:tu:=nu;
      nu+1:tu:=0;
      nu+2:tu:=1;
      end;
      for j:=-1 to 2 do begin
        tv:=1+j;
        case tv of
        -1:tv:=nv;
        nv+1:tv:=0;
        nv+2:tv:=1;
        end;
        kp[i,j]:=a[tu,tv];
      end;
```

```
 end;
end;

function xx: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].xp*fu[i]*fv[j];
  xx:=s/dd;
end;

function yy: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].yp*fu[i]*fv[j];
  yy:=s/dd;
end;

function zz: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].zp*fu[i]*fv[j];
  zz:=s/dd;
end;

procedure rajzracs; {felületháló csomópontjainak kiszámítása}
begin
  if fdu=0 then begin
    delta_u;
    if abs(du)<0.1 then begin
      if du<0 then b2u:=b2u-1 else b2u:=b2u+1;
      delta_u; end; {nullaosztás elkerülése}
  end;
  if fdv=0 then begin
    delta_v;
    if abs(dv)<0.1 then begin
      if dv<0 then b2v:=b2v-1 else b2v:=b2v+1;
      delta_v; end; {nullaosztás elkerülése}
  end;
  dd:=du*dv;
  for p:=0 to n do begin
    bl_funct_u;
    for r:=0 to n do begin
      bl_funct_v;
      for k:=0 to nu do
        for 1:=0 to nv do begin
          folt;
```

```
            rp [ k*n+p, l*n+r] ^ . uj (xx, yy, zz) ;
          end;
      end;
    end;
    fl:=1; fp:=0;
  end;

begin
  racs; if gb[2]=1 then nyíl; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu+l,nv+l,n,1,1,0); eger_1;
  fr:=1;
end;

procedure fg; {vetítési irány változtatásának vezérlése}
  ..........
begin
  obj:='F';
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to (nu+l)*n do
    for 1:=0 to (nv+l)*n do new(rp[k,l]);
  for k:=1 to (nu+l)*(nv+1) do new(hl[k]);
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.4; fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nu do begin (kezdőkonfiguráció - gyürü}
      psi:=2*pi*k/(nu+l);
      for 1:=0 to nv do begin
        fi:=2*pi*1/(nv+l)+pi/2;
        a[k,1] .uj ((ru+ry*sin(fi))*cos(psi), (ru+ry*sin(fi)) *sin(psi),
        ry*cos (fi)) ;
      end;
    end;
    qk:='N';
    racs; konfig('ZYB'); eger_n; parancs(2); bvalt;
    racs; kpontok(nu+l,nv+1,1,1);
    repeat
      eger_t(kx0,2,620,ky0); eger_l; qr:=' '; b:=0;
      repeat par_katt(2,qr); kp_r_katt(nu,nv); until qr in valasz;
      eger_n;
      case qr of
        'V':begin (kontrollpont kiválasztása és koordinátáinak
                   módosítása)
          gb[3]:=0; fe:=0; racs; kpontok(nu+l,nv+l,1,1);
          gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
          inform(szx, szy, '   Hely:  ');kp_kor (szx, szy, 12) ;
          koordin3('Hely');
          if qr<>char(27) then a[szx,szy].uj(x,y,z);
          kp_kor(szx,szy,7); racs; kpontok(nu+l,nv+l,l,l);
          gb[1]:=0; lap(208,5); setfillstyle(1,11);
          bar(130,470,590,480);
          fl:=0;
        end;
```

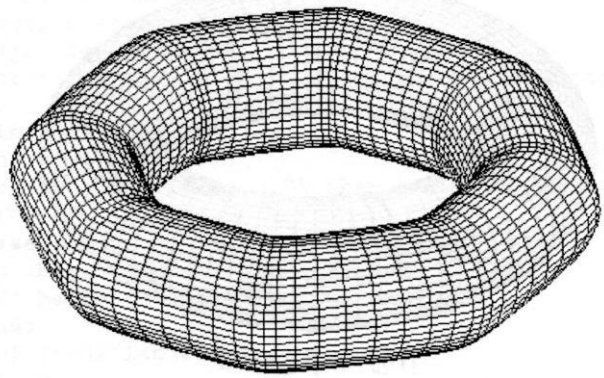
```
      'F':begin {felületmegjelenítés}
            gb[3]:=1; parancs(2); fe:=1;
            megj; rajz; gb[3] :=0;
          end;
      'E':if fe=1 then begin {felületmegjelenítés keret nélkül}
            gb[2]:=1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb[2]:=0;
            keret(kx0, ky0); kp_racs(nu,nv); megj; rajz;
          end;
      'A':begin (eltolásmódosítás)
            gb[7]:=1; bvalt;
            irany; eltl; if b1(0.05 then b1:=0.05; if bl>20 then
            bl : =20;
            case qd of
            'U':begin b1u:=b1; fdu:=0;
            end; 'V':begin blv:=b1;
            fdv:=0; end; end;
            gb[7]:=0; lap(208,5);
            f1:=0;
          end;
      'C':begin {feszültségmódosítás}
            gb[8]:=1; bvalt;
            irany; fesz;
            case qd of
            'U':begin b2u:=b2; fdu:=0; end;
            'V':begin b2v:=b2; fdv:=0; end;
            end;
            gb[8]:=0; lap(208,5);
            f1:=0;
          end;
      'Q':begin (kilépés)
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('ZYB') until not ((v='N') and
            (qm='I')); vege; if qk='N' then ujkonf; if
            qu='N' then begin gb[3]:=0; racs;
            kpontok(nu+1,nv+1,1,1); end;
            gb[4]:=0;
          end;
      end;
      parancs(2); bvalt;
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu=' N' ;
  ciosegraph;
end.
```



3.47. ábra

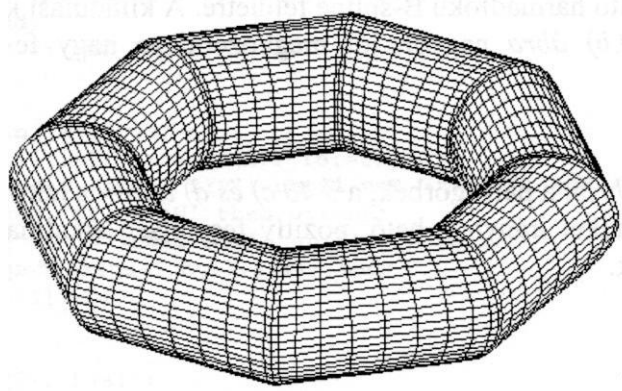
Az alábbi ábrákon a pozitív feszültség hatását szemléltetjük ((3 I u= β_{Iv}= 1) a *3.47. ábrán* látható harmadfokú B-spline felületre. A kiindulási karakterisztikus poliédert a *3.49.b) ábra* nagyon jól megközelíti (a nagy feszültségértékek miatt).

A *3.48 a) és b) ábrán* a pályagörbék, a *3.48 c) és d) ábrán* a leírógörbék, a *3.49. ábrán* pedig mindkét irányban ható, pozitív feszültség alkalmazásával kapott felületek láthatók.



a)

$\beta_{2u} = 10\,,\ \beta_{2v} = 0$



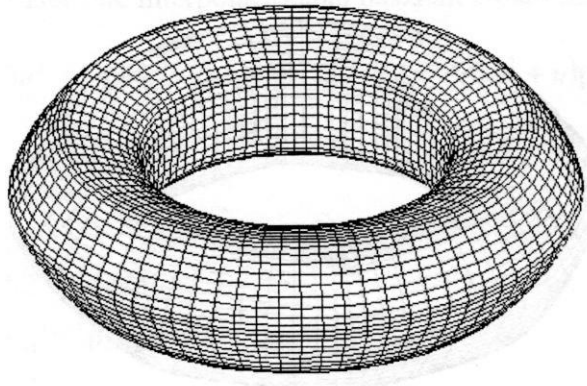
b)

$\beta_{2u} = 100\,,\ \beta_{2v} = 0$

3.48. ábra

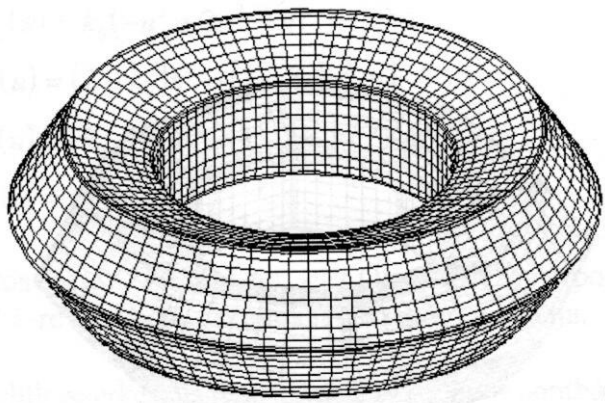
$\beta_{2u} = 0, \beta_{2v} = 100$

3. 48. ábra

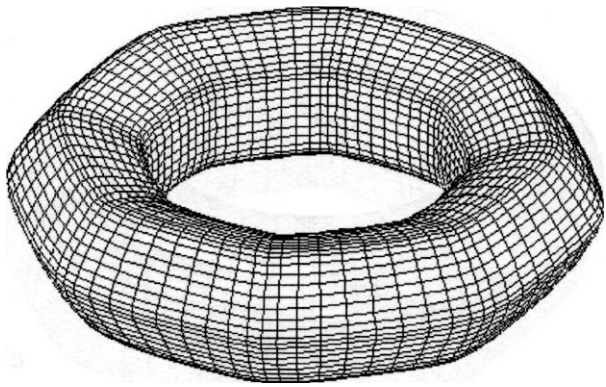


c)

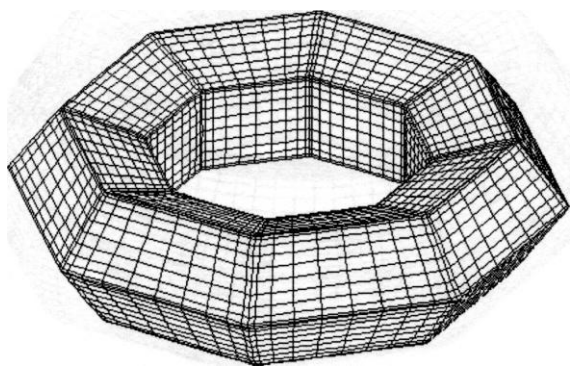
$\beta_{2u} = 0, \beta_{2v} = 5$



d)



$\beta_{2u} = \beta_{2v} = 5$



$\beta_{2u} = \beta_{2v} = 100$

3. 49. ábra

3.3.6. Helyi ellenőrzésű spline interpoláció

A görbék Coons-Hermite interpolációjánál használt előállításban:

$$\mathbf{r}(u) = (2u^3 - 3u^2 + 1)\mathbf{p}_i + (-2u^3 + 3u^2)\mathbf{p}_{i+1} + (u^3 - 2u^2 + u)\mathbf{p}_i' + (u^3 - u^2)\mathbf{p}_{i+1}' \tag{3.58}$$

az érintőket

$$\begin{aligned} \mathbf{p}_i' &= (\mathbf{p}_{i+1} - \mathbf{p}_{i-1})k_0 \\ \mathbf{p}_{i+1}' &= (\mathbf{p}_{i+2} - \mathbf{p}_i)k_0 \end{aligned} \tag{3.59}$$

képletekkel számolva a következő előállítást

$$\mathbf{r}(u) = \sum_{j=-1}^{2} f_j(u)\mathbf{p}_{i+j} \; ; \quad i = 0,1,\ldots,n-1 \, , \quad u \in [0,1] \quad , \tag{3.60}$$

ahol

$$\begin{aligned} f_{-1}(u) &= k_0(-u^3 + 2u^2 - u) \\ f_0(u) &= (2 - k_0)u^3 - (3 - k_0)u^2 + 1 \\ f_1(u) &= (k_0 - 2)u^3 + (3 - 2k_0)u^2 + k_0 u \\ f_2(u) &= k_0(u^3 - u^2) \end{aligned} \quad , \tag{3.61}$$

amelynek sajátos esete - $k_0 = 0.5$-tel - az Overhauser interpoláció. A 3.61 előállítás rögzített *i* -re a p_i és p_{i+1} pontokat összekötő ívet adja.

Ez az eljárás felületszerkesztésre is alkalmas. Az *mxn* pontból álló karakterisztikus kereten 4x4 -es patcheket tekintve:

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_{i-1,j-1} & \mathbf{p}_{i-1,j} & \mathbf{p}_{i-1,j+1} & \mathbf{p}_{i-1,j+2} \\ \mathbf{p}_{i,j-1} & \mathbf{p}_{i,j} & \mathbf{p}_{i,j+1} & \mathbf{p}_{i,j+2} \\ \mathbf{p}_{i+1,j-1} & \mathbf{p}_{i+1,j} & \mathbf{p}_{i+1,j+1} & \mathbf{p}_{i+1,j+2} \\ \mathbf{p}_{i+2,j-1} & \mathbf{p}_{i+2,j} & \mathbf{p}_{i+1,j+1} & \mathbf{p}_{i+2,j+2} \end{bmatrix} \; ; \; \begin{cases} i = 0,1,\ldots,m \\ j = 0,1,\ldots,n \end{cases} , \tag{3.62}$$

az

$$\mathbf{r}(u,v) = \begin{bmatrix} f_0(u) & f_1(u) & f_2(u) & f_3(u) \end{bmatrix} \cdot \mathbf{P} \cdot \begin{bmatrix} f_0(v) & f_1(v) & f_2(v) & f_3(v) \end{bmatrix}^T$$
$$(u,v) \in [0,1] \times [0,1] \qquad (3.63)$$

előállítású felületfolt a $\mathbf{p_{ij}}$, $\mathbf{p_{i+1j}}$, p_{i+1j+1} , p_{ij+1} pontokra illeszkedik. A 3.63 előállítás súlyfüggvényei formailag a 3.61 súlyfüggvényekkel megegyezőek, a k0 alakparaméter viszont a két paraméterirányban különböző lehet - k_{0u} és k_{0v}.

Nyílt felület szélein álló kontrollpontokat kétszeresnek tekintjük, így a felületet szegélyező görbék a karakterisztikus keret szélein álló pontokat interpoláló, a 3.61-hez hasonló előállítású görbék lesznek.

Ezt a módszert valósítja meg az F_P_HSPL.PAS program, amely megengedi az alakparaméterek interaktív megválasztását a [0,1] intervallumban.

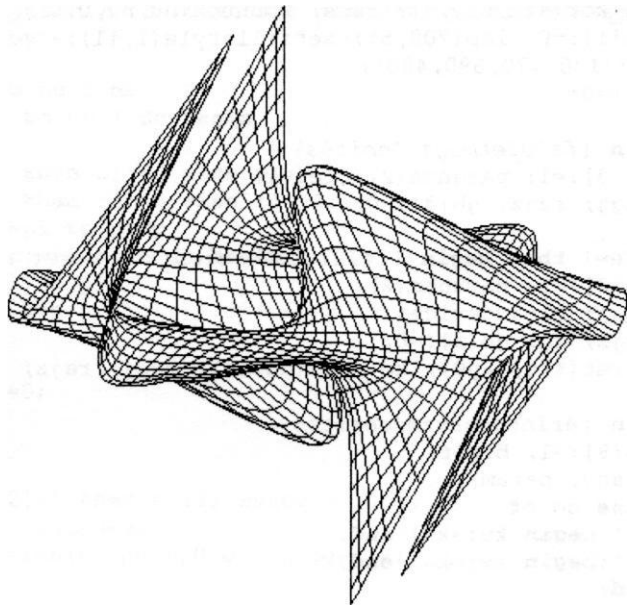
```
{$N+}

program f_p_hspl;
    {paraméteres előállítású nyílt felületek helyi ellenőrzésű spline 1
interpolációval)


uses
  graph, grafind, crt, dos, kozos, kozos 3d, kerdesek, rajz_f3d,
  konf 3d, komm_fel, eger_kez, gomb kez;

const
  nu=4; nv=4; ax=80; ay=80; n=10;
  valasz: kars =
  valasz f: kars = ['D','T', fel, le, jobb, bal, char(27)];

procedure rajz;
var
  u, u2, u3, v, v2, v3 : single;
  kp: array[-1..2,-1..2] of pont3;
  fu, fv: array[-1..2] of single;

  procedure sulyfugg; {súlyfüggvények kiszámítása}
  begin
    u:=p/n; u2:=u*u; u3:=u2*u;
    fu[-1]:=(-u3+2*u2-u)*ku; fu[2]:=(u3-u2)*ku;
    fu[0]:=2*u3-3*u2+1-fu[2]; fu[1]:=-2*u3+3*u2-fu[-1];
    v:=r/n; v2:=v*v; v3:=v2*v;
    fv[-1]:=(-v3+2*v2-v)*kv; fv[2]:=(v3-v2)*kv;
    fv[0]:=2*v3-3*v2+1-fv[2];  fv[1]:=-2*v3+3*v2-fv[-1];
  end;
```

```
procedure folt; (felületfoltok (patch) kontrollpontjainak
                  összerendelése)
var tu, tv: integer;
begin
  for i:=-1 to 2 do begin
    tu:=k+i;
    case tu of
    -1:tu:=0;
    nu+1:tu:=nu;
    end;
    for j:=-1 to 2 do begin

      case tv of
      -1:tv:=0;
      nv+1:tv:=nv;
      end;
      kp[i,j].=a[tu,tv];
    end;
  end;
end;

function xx: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].xp*fu[i]*fv[j];
  xx:=s
end;

function yy: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].yp*fu[i]*fv[j];
  yy:=s
end;

function zz: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].zp*fu[i]*fv[j];
  zz:=s
end;
```

```
  procedure rajzracs; (felületháló csomópontjainak kiszámítása)
  var p1, ri: byte;
  begin
    for k:=0 to nu-1 do
      for 1:=0 to nv-1 do begin
        folt;
        if k=0 then p1:=0 else p1:=1;
        if 1=0 then r1:=0 else r1:=1;
        for p:=p1 to n do
          for r:=r1 to n do begin
            sulyfugg;
            rp[k*n+p,1*n+r]A.uj(xx,yy,zz);
          end;
      end;
    f1:=1; fp:=0;
  end;

begin
  racs; if gb[2]=1 then nyíl; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu,nv,n,0,0,1); eger_1;
  fr:=1;
end;

procedure fg; {vetítési irány változtatásának vezérlése)
   ........
begin
  obj ='H';
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to nu*n do
    for 1:=0 to nv*n do new(rp[k,1]);
  for k:=1 to nu*nv do new(hl[k]);
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
    nvi:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvi*sqrt(sqr(nvi)+sqr(cp));
    for k:=0 to nu do for 1:=0 to nv do a[k,1].uj((k-nu/2)*ax,
                                          (1-nv/2)*ay,0);
    qk:='N';
    racs; konfig('FPS'); eger_n; parancs(2); bvalt; racs;
    kpontok(nu,nv,0,0);
    repeat
      eger_t(kx0,2,620,ky0); eger_l; qr:=' '; b:=0;
      repeat par katt(2,qr); kp_r_katt(nu,nv); until qr in valasz;
      eger_n;
      case qr of
      'V':begin {kontrollpont kiválasztása és koordinátáinak
                 módosítása}
            gb[3]:=0; fe:=0; racs; kpontok(nu,nv,0,0);
            gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
            inform(szx,szy,' Hely: ');
            kp_kor(szx,szy,12); koordin3('Hely');
            if qr<>char(27) then a[szx,szy].uj(x,y,z);
```

```
            kp_kor(szx,szy,7); racs; kpontok(nu,nv,0,0);
            gb[1] :=0; lap (208, 5) ; setfilistyle (1, ii) ;
            bar(130,470,590,480);
            f1:=0;
          end;
      'F':begin {felületmegjelenítés}
            gb[3]:=1; parancs(2); fe:=1;
            megj; rajz; gb[3]:=0;
          end;
      'E':if fe=1 then begin (felületmegjelenítés keret nélkül)
            gb[2]:=1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb[2] :=0;
            keret(kx0, ky0) ; kp_racs (nu, nv) ; megj; rajz;
          end;
      'C':begin (érintőhosszmódosítás)
            gb[8]:=1; bvalt;
            irany; param;
            case qd of
            'U':begin ku:=k0; end;
            'V':begin kv:=k0; end;
            end;
            gb[8]:=0; lap(208,5); f1:=0;
          end;
      'Q':begin {kílépés}
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('FPS') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs; kpontok(nu,nv,0,0);
                           end;
            gb[4]:=0;
          end;
      end;
      parancs(2); bvalt;
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```

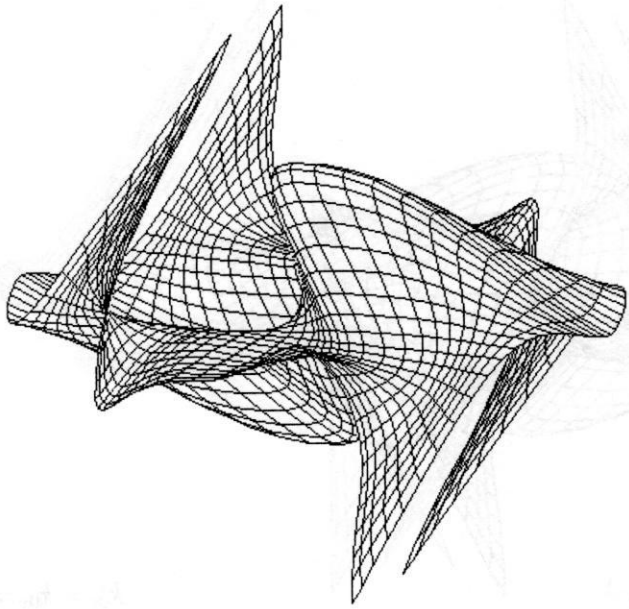
A *3.50. ábrán* a *3.5. ábra* karakterisztikus keretére illesztett Overhauser spline felület látható, a *3.51 - 3.54. ábrák* pedig az alakparaméter változtatásának eredményét szemléltetik.



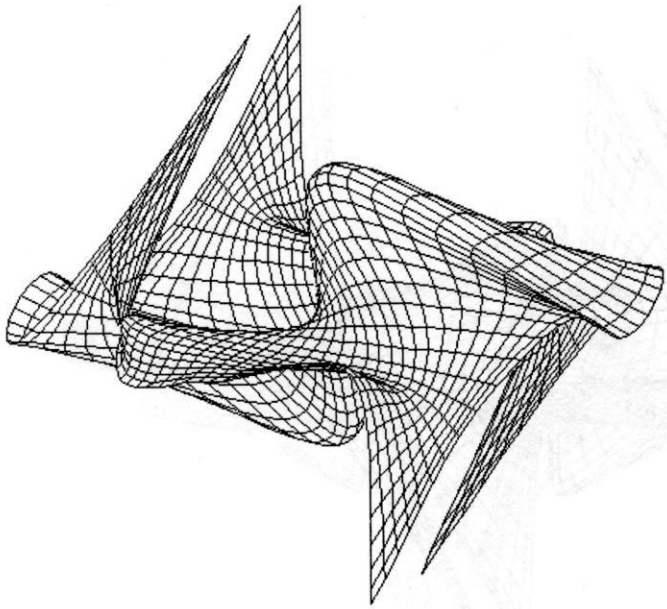
$k_{0u} = k_{0v} = 0.5$

3.50. ábra

Megfigyelhető, hogy ko fordított feszültségként működik, 0-hoz közeli értékekre a felület mintegy ráfeszül a karakterisztikus keretre, 0.5 -nél nagyobb értékekre viszont meglazul, fodrozódik. A k_o paraméter tulajdonképpen az illető irányú paramétervonalak kontrollpontokbeli érintőinek hosszát befolyásolja. Túl nagy, vagy negatív értékekre a paramétergörbéken hurkok keletkeznek, a felület önáthatóvá válik.

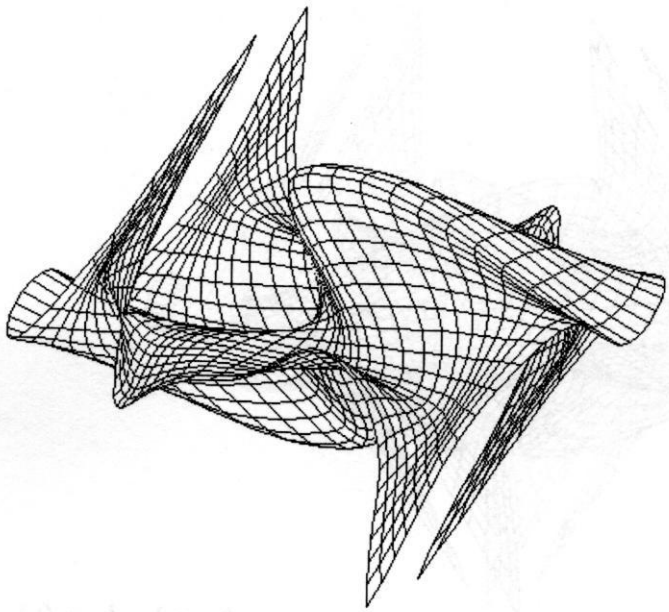


$k_{0u} = 1\ ,\ k_{0v} = 0.5$

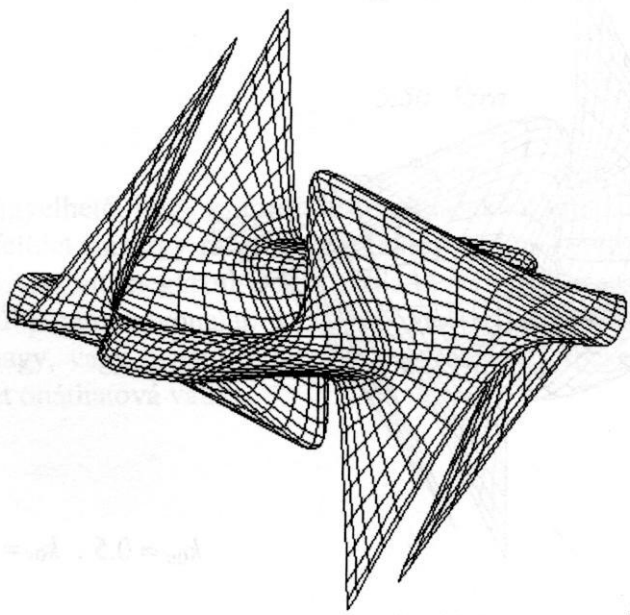


$k_{0u} = 0.5\ ,\ k_{0v} = 1$

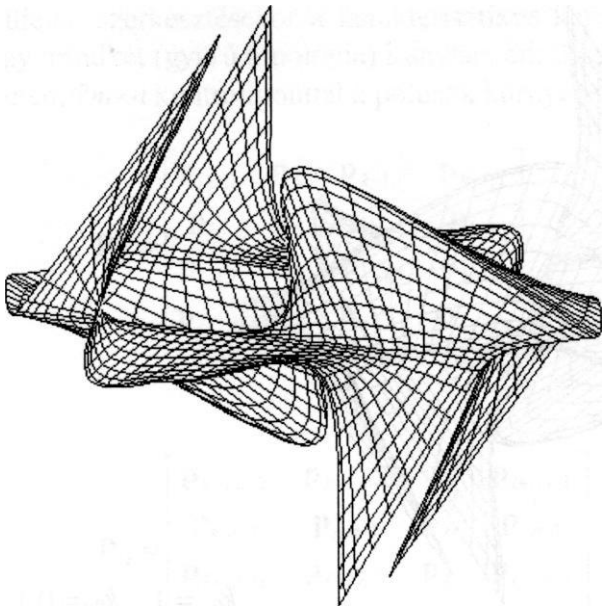
3.52. ábra



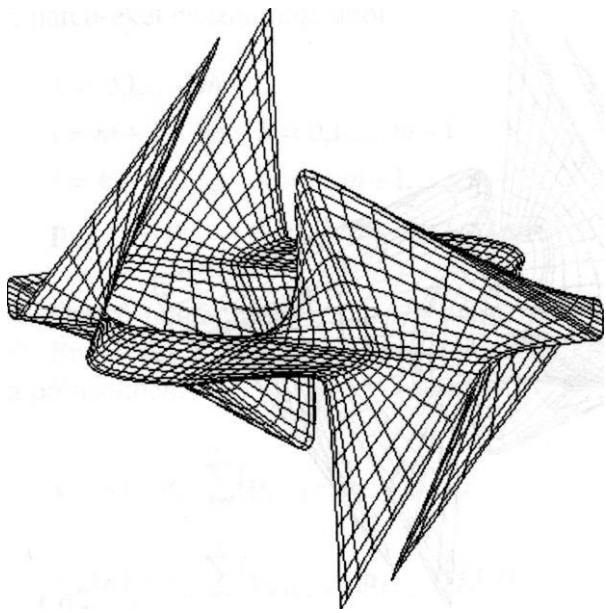
$k_{0u} = k_{0v} = 1$



$k_{0u} = 0.2\ ,\ k_{0v} = 0.5$



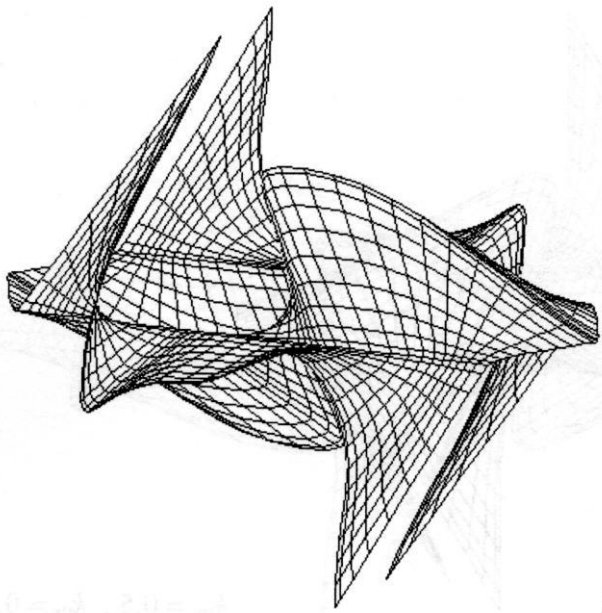
$ko_u=0.5$, $ko_v=0.2$



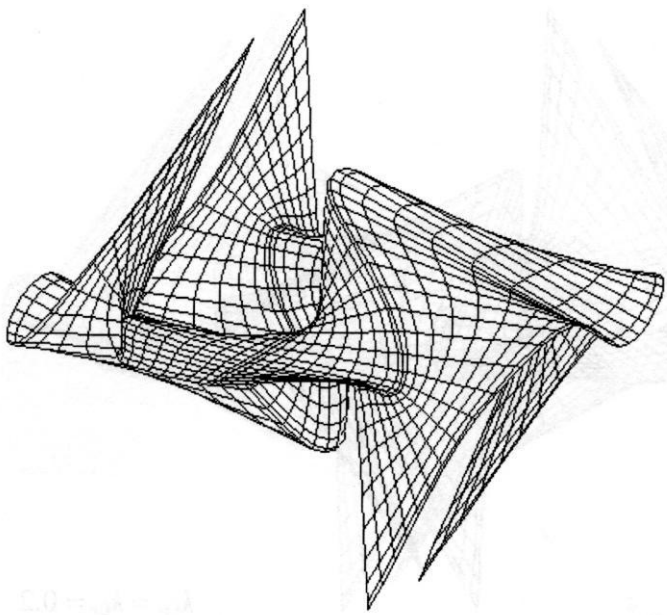
$ko_u=ko_v=0.2$

3.53. ábra

3.54. ábra



$k_{0u} = 1\ ,\ k_{0v} = 0.1$



$k_{0u} = 0.1\ ,\ k_{0v} = 1$

Zárt felületek szerkesztésekor a karakterisztikus keretet egyik (gömb topológia), vagy mindkét (gyűrű topológia) irányban ciklikusan bezárjuk. Gömb topológia esetén, *2m×n* kontrollponttal a pólusok környezetében

$$\mathbf{P}_E = \begin{bmatrix} \mathbf{p}_{l+1,1} & \mathbf{p}_E & \mathbf{p}_{k-1,1} & \mathbf{p}_{k-1,2} \\ \mathbf{p}_{l,1} & \mathbf{p}_E & \mathbf{p}_{k,1} & \mathbf{p}_{k,2} \\ \mathbf{p}_{l-1,1} & \mathbf{p}_E & \mathbf{p}_{k+1,1} & \mathbf{p}_{k+1,2} \\ \mathbf{p}_{l-2,1} & \mathbf{p}_E & \mathbf{p}_{k+2,1} & \mathbf{p}_{k+2,2} \end{bmatrix}, \tag{3.64}$$

illetve

$$\mathbf{P}_D = \begin{bmatrix} \mathbf{p}_{k-1,n-2} & \mathbf{p}_{k-1,n-1} & \mathbf{p}_D & \mathbf{p}_{l+1,n-1} \\ \mathbf{p}_{k,n-2} & \mathbf{p}_{k,n-1} & \mathbf{p}_D & \mathbf{p}_{l,n-1} \\ \mathbf{p}_{k+1,n-2} & \mathbf{p}_{k+1,n-1} & \mathbf{p}_D & \mathbf{p}_{l-1,n-1} \\ \mathbf{p}_{k+2,n-2} & \mathbf{p}_{k+2,n-1} & \mathbf{p}_D & \mathbf{p}_{l-2,n-1} \end{bmatrix} \tag{3.65}$$

összeállítású patch-eket használunk, ahol

$$\begin{aligned} &k = 0,1,\ldots,2m-1 \\ &l = m+k \quad \text{ha} \quad k = 0,1,\ldots,m-1 \\ &l = k-m \quad \text{ha} \quad k = m,m+1,\ldots,2m-1 \\ &\mathbf{p}_{2m+i,j} = \mathbf{p}_{i,j} \;,\; i = -1,0,1,2 \;,\; j = 1,\ldots,n-1 \end{aligned} \;, \tag{3.66}$$

vagyis a pólust belső pontként tartalmazó patch-ek átterjednek a "szembenfekvő féltekére". Ennek (és a súlyfüggvények alakjának) következtében a délgörhék érintői a pólusokban:

$$\begin{aligned} \mathbf{t}_{vE}(u) &= k_{0v} \sum_{i=-1}^{2} \left(\mathbf{p}_{k+i,1} - \mathbf{p}_{l-i,1} \right) f_i(u) \\ \mathbf{t}_{vD}(u) &= k_{0v} \sum_{i=-1}^{2} \left(\mathbf{p}_{k+i,n-1} - \mathbf{p}_{l-i,n-1} \right) f_i(u) \end{aligned} \;. \tag{3.67}$$

Ez az eredmény azt mutatja, hogy két-két, topológiailag egymással szemben fekvő délgörbe egy-egy síma zárt görbét alkot és, amennyiben a pólussal szom-

szédos, egymással szemben fekvők kontrollpontokat összekötő vektorok egy síkban, vagy legalább egymással párhuzamos síkokban fekszenek, akkor a pólusban létezik érintősík.

A ZM_HSPL.PAS program segítségével lehet gömb topológiájú zárt felületeket szerkeszteni helyi ellenőrzésű spline interpolációval, a kiindulási, gömbbe írt kontrollpoliéder és az alakparaméterek interaktív módosításával.

```
($N+)

program zm_hspl;
        (gömb topológiájú zárt felületek helyi ellenőrzésű spline
interpolációval}

uses
  graph, grafind, crt, dos, kozos, kozos 3d, kerdesek, rajz_f3d,
  konf 3d, komm_fel, eger_kez, gomb kez;

coast
  mm=4; nu=2*mm-1; nv=4; rg=160; n=10;
  valasz: kars = ['E','F','Q','V','C'];
  valasz f: kars = ['D','T', fel, le, jobb, bal, char(27)];

var
  fi, psi: single;

procedure rajz;
var
  u, u2, u3, v, v2, v3 : single;
  kp: array[-1..2,-1..2] of ponti;
  fu, fv: array[-1..2] of single;

  procedure sulyfugg; (súlyfüggvények kiszámítása}
  begin
    u:=p/n; u2:=u*u; u3:=u2*u;
    fu[-1]:=(-u3+2*u2-u)*ku; fu[2]:=(u3-u2)*ku;
    fu[0]:=2*u3-3*u2+1-fu[2]; fu[1]:=-2*u3+3*u2-fu[-1];
    v:=r/n; v2:=v*v; v3:=v2*v;
    fv[-1]:=(-v3+2*v2-v)*kv; fv[2]:=(v3-v2)*kv;
    fv[0]:=2*v3-3*v2+1-fv[2];  fv[1]:=-2*v3+3*v2-fv[-1];
  end;

  procedure folt; (felületfoltok (patch) kontrollpontjainak
                   összerendelése}
  var tu, tv, ts: integer;
  begin
    for i:=-1 to 2 do begin
      tu:=k+i;
      case tu of
      -1:tu:=nu;
```

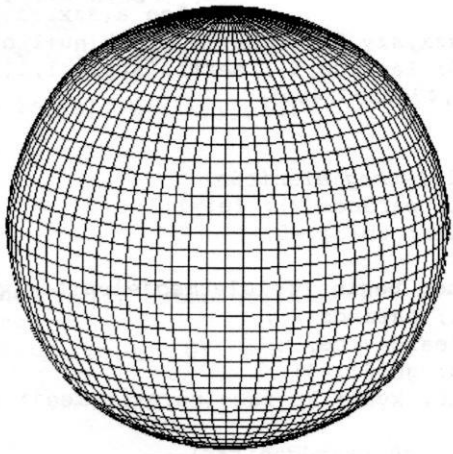
```
    nu+1:tu:=0;
    nu+2:tu:=1;
    end;
    for j:=-1 to 2 do begin
      tv:=1+j;
      case tv of
      -1:begin tv:=1; if tu<mm then ts:=tu+mm else ts:=to-mm; end;
      nv+1:begin tv:=nv-1; if tu<mm then ts:=tu+mm else ts:=to-mm;
           end;
      else ts:=tu;
      end;
      kp[i,j]:=a[ts,tv];
    end;
  end;
end;

function xx: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].xp*fu[i]*fv[j];
  xx:=s
end;

function yy: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].yp*fu[i]*fv[j];
  yy:=s
end;

function zz: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].zp*fu[i]*fv[j];
  zz:=s
end;

procedure rajzracs; {felületháló csomópontjainak kiszámítása}
var p1, rl: byte;
begin
  for k:=0 to nu do
    for 1:=0 to nv-1 do begin
      folt;
      if k=0 then pl:=0 else pl:=1;
```

```
        if 1=0 then r1:=0 else r1:=1;
        for p:=p1 to n do
          for r:=r1 to n do begin
            sulyfugg;
            rp[k*n+p,l*n+r]^.uj (xx,yy,zz);
          end;
       end;
    fl:=1; fp:=0;
  end;

begin
  racs; if gb[2]=1 then nyit; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu+1,nv,n,1,0,0); eger_1;
  fr:=1;
end;

procedure fg; {vetítési irány változtatásának vezérlése}
   ........
begin
  obj:='H';
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to (nu+1)*n do
    for 1:=0 to nv*n do new(rp[k,1]);
  for k:=1 to (nu+1)*nv do new(hl[k]);
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.2; fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nv1)+sqr(cp));
    for k:=0 to nu do begin (kezdőkonfiguráció - gömb)
       psi:=2*pi*k/(nu+1);
       for 1:=0 to nv do begin
         fi:=pi*1/nv;
         a [k, 1] .uj (rg*sin (fi) *cos (psi) , rg*sin (fi) *sin (psi) , rg*cos (fi)) ;
       end;
    end;
    qk:='N';
    racs; konfig('ZMS'); eger_n; parancs(2); bvalt; racs;
    kpontok(nu+1,nv,1,0);
    repeat
       eger_t(kx0,2,620,ky0); eger_1; qr:=''; b:=0;
       repeat par katt(2,qr); kp r katt(nu,nv); until qr in valasz;
       eger_n;
       case qr of
       'V':begin {kontrollpont kiválasztása és koordinátáinak
                  módosítása}
            gb[3]:=0; fe:=0; racs; kpontok(nu+1,nv,1,0);
            gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
            inform(szx,szy,'    Hely: '); kp_kor(szx,szy,12);
            koordin3('Hely');
            if qr<>char(27) then
              if (szy=0) or (szy=nv) then for k:=0 to nu do
                   a[k, szy] .uj (x,y, z)
```

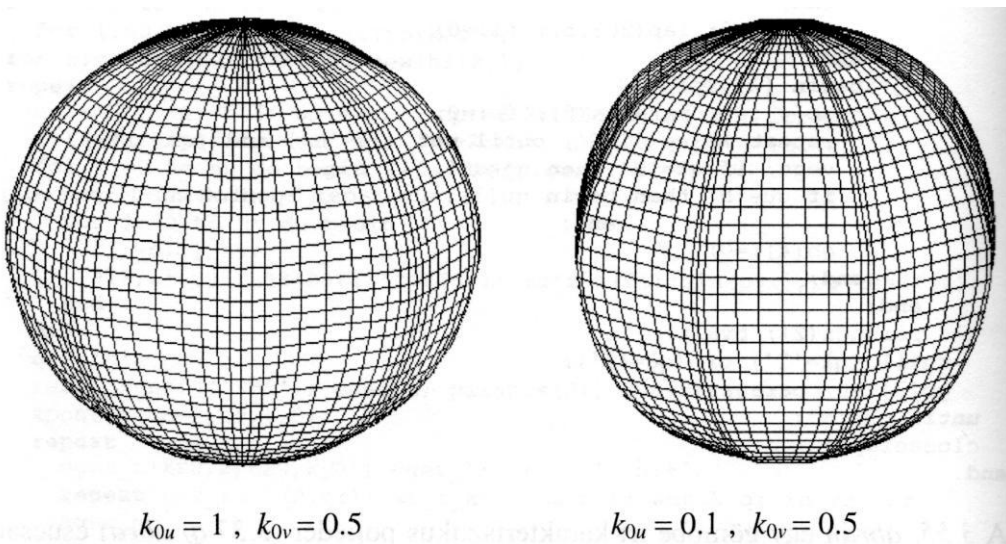
```
                                        else a [szx, szy] . uj (x, y, z) ;
            kp_kor(szx,szy,7); racs; kpontok(nu+1,nv,1,0);
            gb[1]:=0; lap(208,5); setfillstyle(1,11);
            bar(130,470,590,480);
            fl:=0;
          end;
      'F':begin (felületmegjelenítés)
            gb[3]:=1; parancs(2); fe:=1;
            megj; rajz; gb[3]:=0;
          end;
      'E':if fe=1 then begin (felületmegjelenítés keret nélkül)
            gb[2]:=1; parancs(2);
            megj; cleardevice;
            rajz; fg; gb[2] :=0;
            keret(kx0, ky0); kp_racs(nu,nv); megj; rajz;
          end;
      'C':begin (érintöhosszmódosítás)
            gb[8]:=1; bvalt;
            irany; param;
            case qd of
            'U':begin ku:=k0; end;
            'V':begin kv:=k0; end;
            end;
            gb[8]:=0; lap(208,5);
          end;
      'Q':begin (kilépés)
            gb[4]:=1; parancs(2); fe:=0;
            repeat kilep('ZMS') until not ((v='N') and (qm='I'));
            vege; if qk='N' then ujkonf;
            if qu='N' then begin gb[3]:=0; racs; kpontok(nu+1,nv,1,0);
                           end;
            gb[4]:=0;
          end;
      end;
      parancs(2); bvalt;
    until (qk='I') or (qu='I');
    gb[3]:=0; parancs(2);
  until qu='N';
  closegraph;
end.
```

A *3.55. ábrán* egy gömbbe írt karakterisztikus poliéder *(3.37.a) ábra)* csúcsait interpoláló Overhauser spline felület látható. A felület szabályossága arra enged következtetni, hogy a $_{kou}$ és $_{ko,}$ paraméterek természetes értéke 0.5.

3.55. ábra

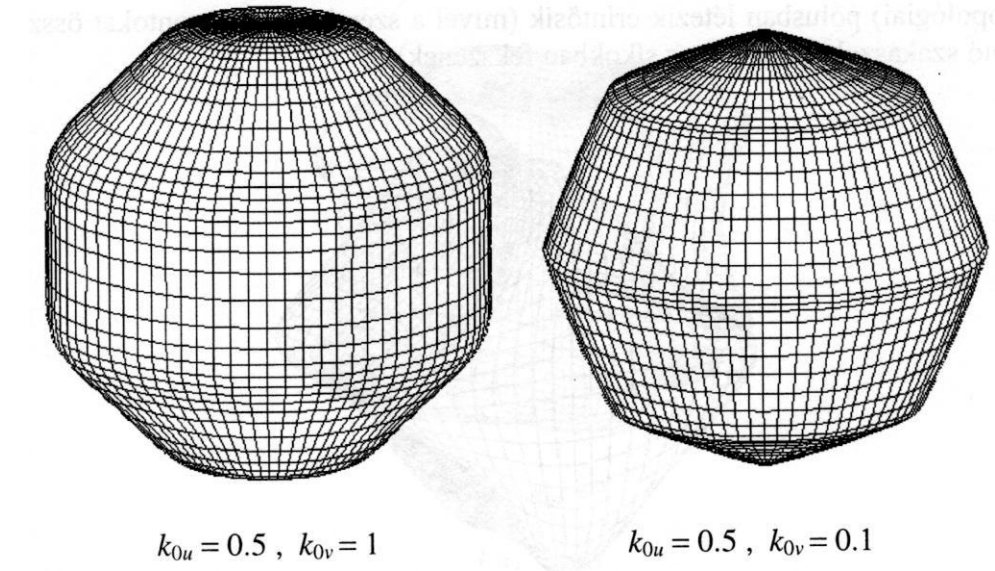


$k_{0u} = k_{0v} = 0.5$

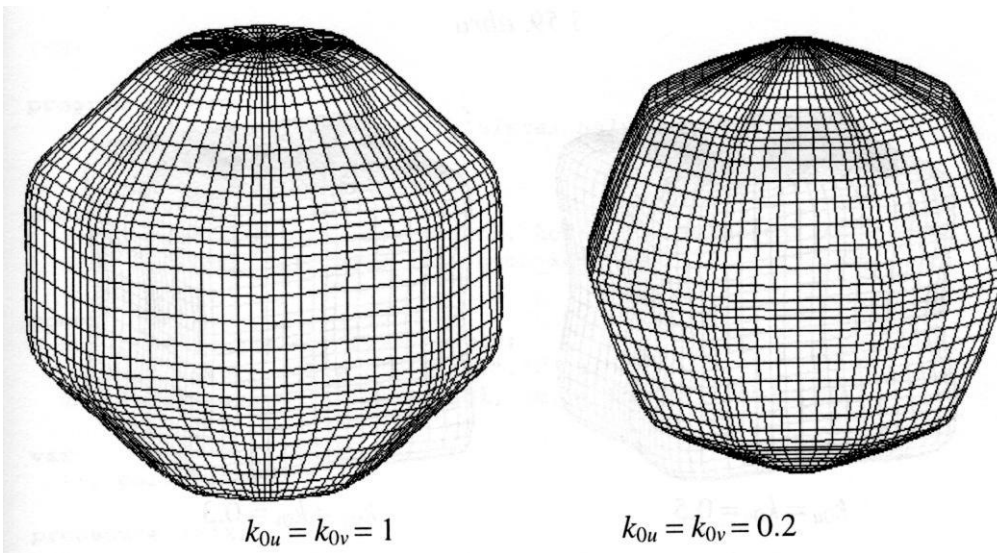


3.56. ábra

A *3.56. ábra* a szélességi-, a *3.57. ábra* a délgörbék menti alakparaméter hatását, míg a *3.58. ábra* a két alakparaméter együttes hatását szemlélteti.



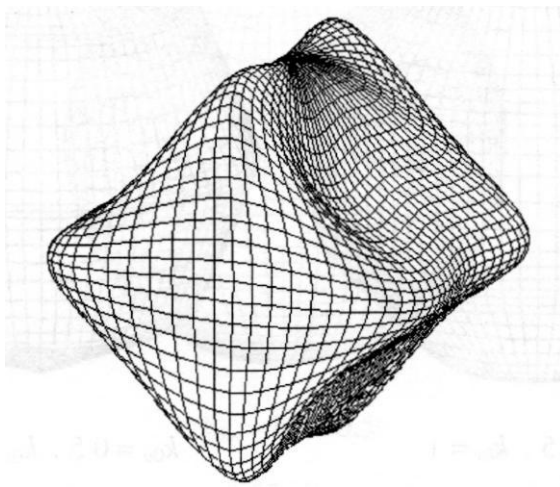
3.57. ábra



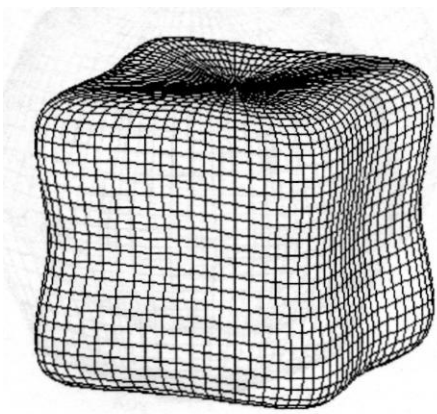
3.58. ábra

A *3.59. ábrán* egy élére állított kockán elhelyezkedő kontrollpontrácsot *(3.12. a) ábra)* interpoláló Overhauser spline látható. Megfigyelhető, hogy bár a pólusokkal szomszédos kontrollpontok nem fekszenek mind ugyanabban a síkban, a

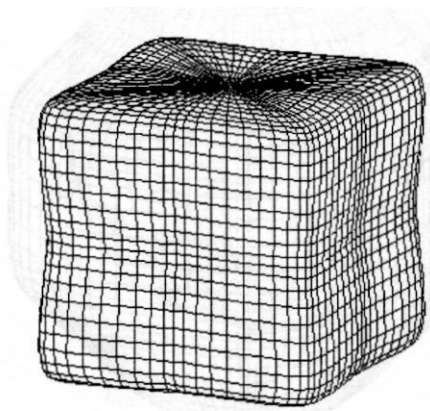
(topológiai) pólusban létezik érintősík (mivel a szembenfekvő pontokat összekötő szakaszok párhuzamos síkokban fekszenek).



$k_{0u} = k_{0v} = 0.5$

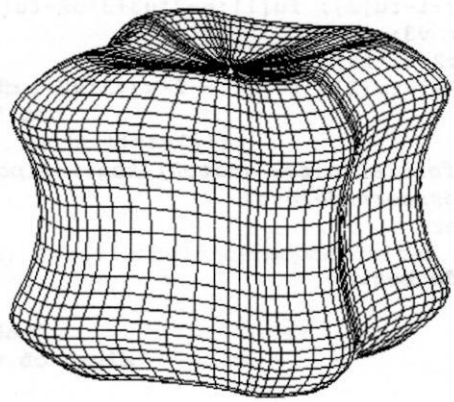


$k_{0u} = k_{0v} = 0.5$



$k_{0u} = k_{0v} = 0.3$

A *3.60. ábrán* a *3.13. a) ábra* karakterisztikus keretére illesztett Overhauser felület és egy feszültség alkalmazásával kapott felület látható. A 3.61. *ábrán* ugyanarra a keretre illesztett "laza" felületet láthatjuk.



$k_{0u} = k_{0v} = 1$

3.61. ábra

Gyűrű topológiájú felületek szerkesztésére szolgál az alábbi, ZY_HSPL.PAS nevű program.

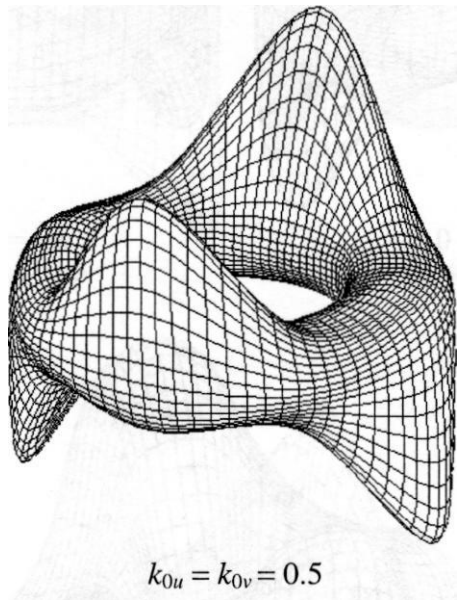
```
{$N+}

program zy_hspl;
    {gyűrű topológiájú zárt felületek helyi ellenőrzésű spline
     interpolációval}

uses
  graph, grafind, crt, dos, kozos, kozos_3d, kerdesek, rajz_f3d,
  konf 3d, komm_fel, eger kez, gomb kez;

const
  nu=7; nv=4; ru=160; ry=60; n=10;
  valasz: kars =
  valasz f: kars = ['D','T', fel, le, job, bal, char(27));

var
  fi, psi: single;

procedure rajz;
var
  u, u2, u3, w, w2, w3, v, v2, v3, t, t2, t3 : single;
  kp: array[-1..2,-1..2] of pont3;
  fu, fv: array(-1..2] of single;

  procedure sulyfugg; {súlyfüggvények kiszámítása}
  begin
    u:=p/n; u2:=u*u; u3:=u2*u;
```

```
  fu[-1]:=(-u3+2*u2-u)*ku; fu[2]:=(u3-u2)*ku;
  fu[0]:=2*u3-3*u2+1-fu[2]; fu[1]:=-2*u3+3*u2-fu[-1];
  v:=r/n; v2:=v*v; v3:=v2*v;
  fv[-1]:=(-v3+2*v2-v)*kv; fv[2]:=(v3-v2)*kv;
  fv[0]:=2*v3-3*v2+1-fv[2]; fv[1]:=-2*v3+3*v2-fv[-1];
end;

procedure folt; {felületfoltok (patch) kontrollpontjainak
                 összerendelése)
var tu, tv: integer;
begin
  for i:=-1 to 2 do begin
    tu:=k+i;
    case tu of
    -1:tu:=nu;
    nu+1:tu:=0;
    nu+2:tu:=1;
    end;
    for j:=-1 to 2 do begin
      tv:=1+j;
      case tv of
      -1:tv:=nv;
      nv+1:tv:=0;
      nv+2:tv:=1;
      end;
      kp[i,j]:=a[tu,tv];
    end;
  end;
end;

function xx: single;
var s: single;
begin
  s:=0;
  for i:=-1 to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j).xp*fu[i]*fv[j];
  xx:=s
end;

function yy: single;
var s: single;
begin
  s:=0;
  for i:=-i to 2 do
    for j:=-1 to 2 do
      s:=s+kp[i,j].yp*fu[i]*fv[j];
  yy:=s
end;
```

```
  function zz: single;
  var s: single;
  begin
    s:=0;
    for i:=-1 to 2 do
      for j:=-1 to 2 do
        s:=s+kp[i,j].zp*fu[i]*fv[j];
    zz:=s
  end;

  procedure rajzracs; (felületháló csomópontjainak kiszámítása}
  var pl, r1: byte;
  begin
    for k:=0 to nu do
      for 1:=0 to nv do begin
        folt;
        if k=0 then pl:=0 else pl:=1;
        if 1=0 then r1:=0 else r1:=1;
        for p:=p1 to n do
          for r:=r1 to n do begin
            sulyfugg;
            rp[k*n+p,l*n+r]^.uj (xx,yy,zz);
          end;
      end;
    fl:=1; fp:=0;
  end;

begin
  racs; if gb[2]=1 then nyil; setcolor(15);
  if f1=0 then rajzracs;
  eger_n; megjelen(nu+1,nv+1,n,1,1,0); eger_1;
  fr:=1;
end;

procedure fg; (vetítési irány váltortatásának vezérlése)
   ..........

begin
  obj ='H';
  grindhi; keret(kx0, ky0); kp_racs(nu,nv); eger_k;
  for k:=0 to (nu+1)*n do
    for 1:=0 to (nv+1)*n do new(rp[k,l]);
  for k:=1 to (nu+1)*(nv+1) do new(hl[k]);
  repeat
    qu:='N'; ap:=0.3; bp:=-0.7; cp:=0.4; fl:=0; fr:=0; fp:=0; fe:=0;
    nvl:=sqrt(sqr(ap)+sqr(bp)); nv2:=nvl*sqrt(sqr(nvl)+sqr(cp));
    for k:=0 to nu do begin (kezdőkonfiguráció - gyűrű)
      psi:=2*pi*k/(nu+1);
      for 1:=0 to nv do begin
        fi:=2*pi*1/(nv+1)+pi/2;
        a[k,1].uj((ru+rv*sin(fi))*cos(psi),(ru+ry*sin(fi))*sin(psi),
        ry*cos (fi)) ;
      end;
```

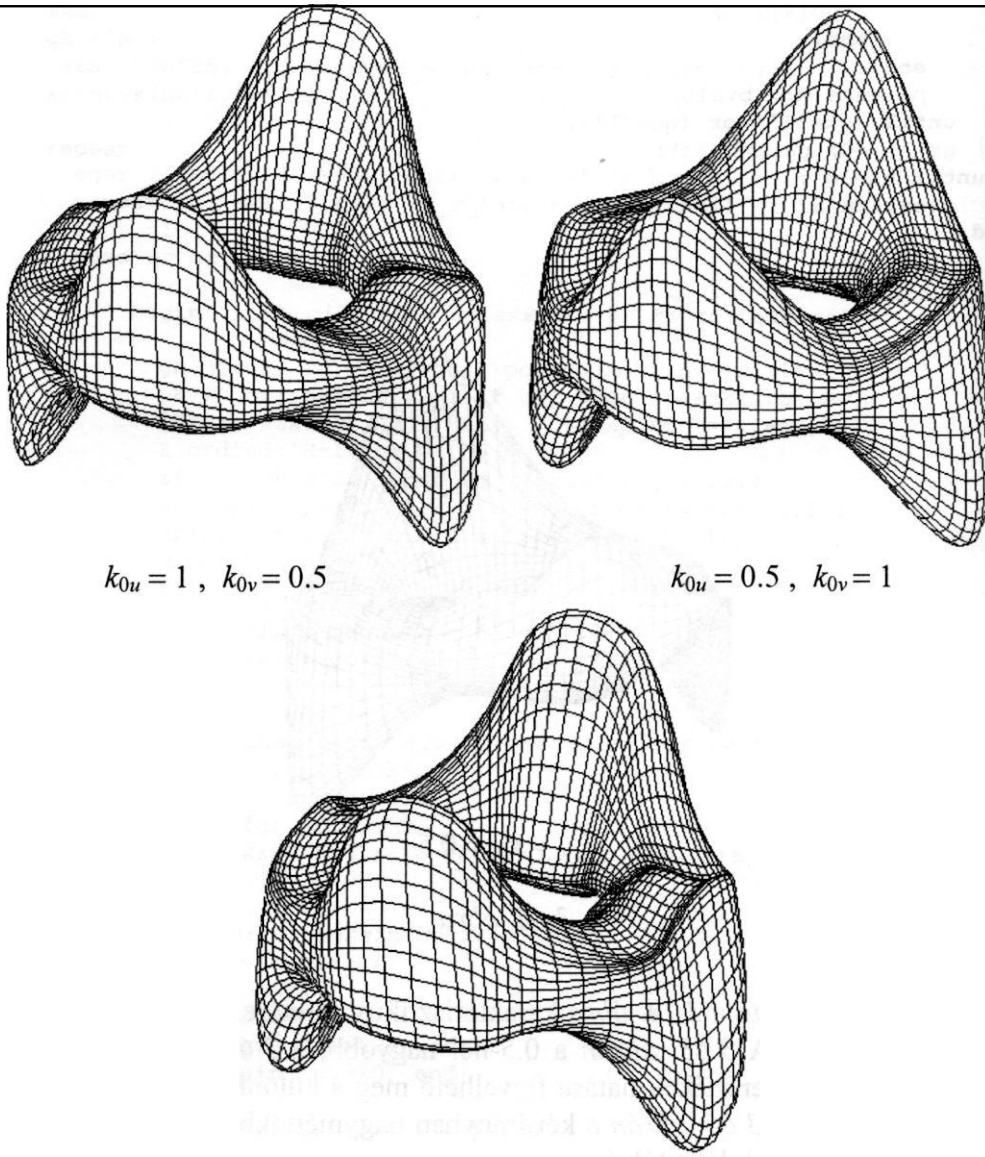
```
end;
qk:='N';
racs; konfig('ZYS'); egern; parancs(2); bvalt; racs;
kpontok(nu+1,nv+1,1,1);

repeat
   eger_t(kx0,2,620,kyo); eger_1; qr:=' '; b:=0;
   repeat par_katt(2,qr); kp_r_katt(nu,nv); until qr in valasz;
   eger_n;
   case qr of

   'V':begin {kontrollpont kiválasztása és koordinátáinak
              módosítása}
         gb[3]:=0; fe:=0; racs; kpontok(nu+l,nv+1,1,1);
         gb[1]:=1; parancs(2); if b=0 then sorszam(nu, nv);
         inform(szx,szy,'   Hely: '); kp_kor(szx,szy,12);
         koordin3('Hely');
         if qr<>char(27) then a[szx,szy].uj(x,y,z);
         kp_kor(szx,szy,7); racs; kpontok(nu+l,nv+1,1,1);
         gb[1]:=0; lap(208,5); setfi1lstyle(1,11);
         bar(130,470,590,480);
         fl :=0;
      end;

   'F':begin {felületmegjelenítés}
        gb[3]:=1; parancs(2);
        fe:=1; megj; rajz;
        gb[3]:=0; end;

   'E':if fe=1 then begin {felületmegjelenítés keret nélkül}
         gb[2]:=1; parancs(2);
         megj; cleardevice;
         rajz; fg; gb[2] :=0;
         keret(kx0, ky0); kp_racs(nu,nv); megj; rajz;
      end;

   'C':begin férintöhosszmódositás}
         gb[8]:=1; bvalt;
         irany; param;
         case qd of
         'U':begin ku:=k0; end;
         'V':begin kv:=k0; end;
         end;
         gb[8]:=0; lap(208,5); fl:=0;
      end;

   'Q':begin (kilépés}
         gb[4]:=1; parancs(2); fe:=0;
         repeat kilep('ZYS') until not ((v='N') and
         (qm='I')); vege; if qk='N' then ujkonf; if qu='N'
         then begin gb[3]:=0; racs;
         kpontok(nu+1,nv+1,1,1); end;
```

```
                gb[4] :=0;
              end;
        end;
        parancs(2); bvalt;
      until (qk='I') or (qu='I');
      gb [ 3 ] : =0; parancs (2) ;
    until qu=' N' ;
    closegraph;
end.
```



3.62. ábra

A *3.62. ábrán* látható a *3.14. ábra* karakterisztikus keretére illesztett Overhauser spline felület. A *3.63. ábrán* a 0.5-nél nagyobb, a *3.64. ábrán* a 0.5-nél kisebb alakparaméterértékek hatása figyelhető meg a különböző paramétergörbék mentén, míg a *3.65. ábrán* a két irányban nagymértékben különböző értékekre kapott felületek láthatók.



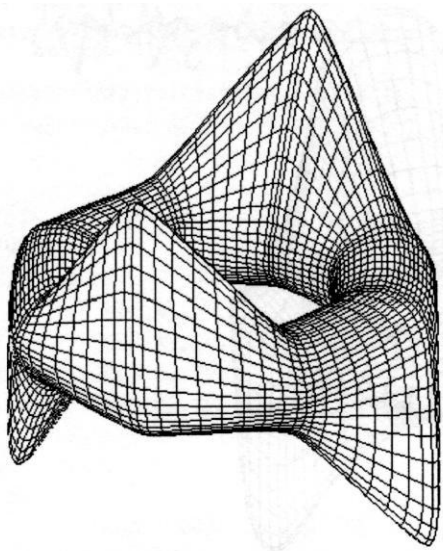
$k_{0u} = 1\ ,\ k_{0v} = 0.5$

$k_{0u} = 0.5\ ,\ k_{0v} = 1$

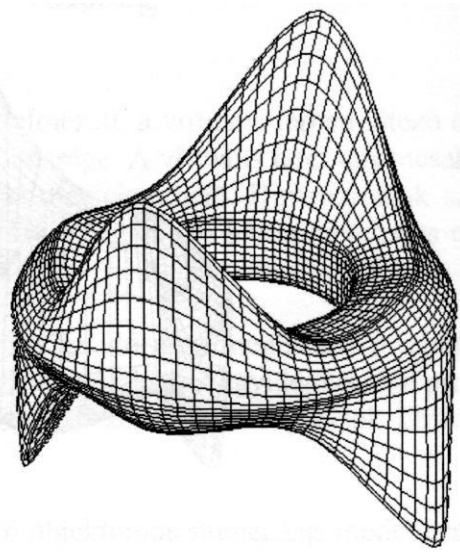
$k_{0u} = k_{0v} = 1$

3.63. ábra

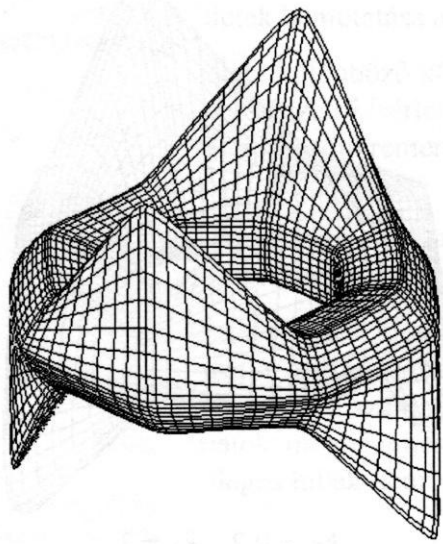
3.64. ábra



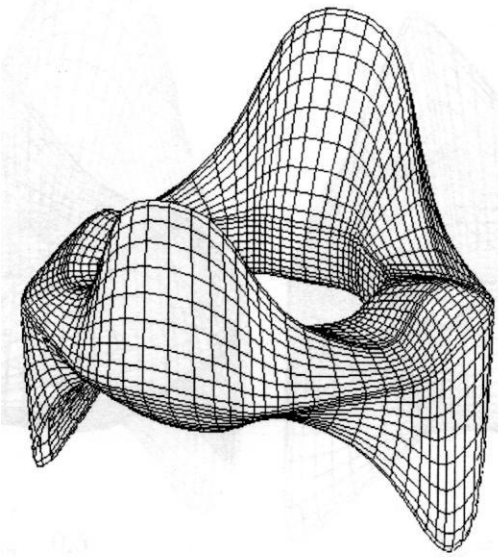
$k_{0u} = 0.2\ ,\ k_{0v} = 0.5$



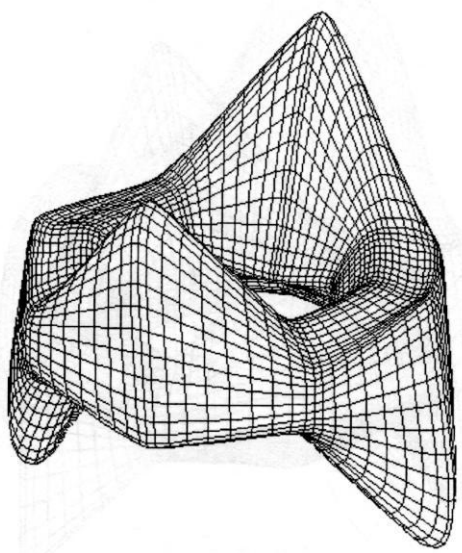
$k_{0u} = 0.5\ ,\ k_{0v} = 0.2$



$k_{0u} = k_{0v} = 0.2$



$k_{0u} = 1\ ,\ k_{0v} = 0.2$



$k_{0u} = 0.2\ ,\ k_{0v} = 1$

3.65. ábra

4. Virtuális valóság

A virtuális valóság a programozó által értelmezett, a virtuális térben létező objektumok és ott lejátszódó folyamatok összessége. A virtuális teret ugyancsak a programozó értelmezi, elvont fogalom, de megjeleníthető. Dimenzióinak száma nem korlátozott, bár szemléletesen csak a legfeljebb háromdimenziós objektumok jeleníthetők meg.

Az igazitól eltérően, a virtuális valóságban az idő nem meghatározó jellegű, nem visszafordíthatatlan. A mozgások, folyamatok kívánt egyidejűségét a programban biztosítani kell, különben ezt az egyes szimulációs eljárások gépidő-igénye határozza meg.

Számos területen alkalmazható különböző objektumok numerikus modellezése és folyamatok szimulációja:

- tervezett tárgyak, szerkezetek viselkedésének vizsgálata különböző körülmények között. Az előállítási költségek csökkentésének fontos eszköze (a prototípus előállítása előtt meg lehet határozni ennek optimális konfigurációját). Ide tartozik a tervezett épületek bemutatása a megrendelőnek.
- létező tárgyak viselkedésének vizsgálata különböző külső behatások esetén. Nem helyettesíti a valódi modellekkel végzett kísérleteket és méréseket, de kiegészíti ezeket (a költséges prototípus tönkremenetelével járó tesztek számát lehet csökkenteni)
- a valóságban megvalósíthatatlan objektumok, folyamatok megjelenítése (számítógépes játékok, videoreklámok, filmgyártás).
- költséges eszközöket kezelő személyzet (pilóták, űrrepülők) felkészítése. Itt fontos a valósághű, valós időben zajló szimuláció megvalósítása, amely komoly igényeket támaszt a számítógéppel szemben is (nagy műveleti sebesség szükséges a gyors folyamatok megjelenítéséhez). Nem elhanyagolható előny azonban, hogy az esetleges hibák nem vezetnek balesetekhez sem anyagi károkat nem okoznak.
- személyzet felkészítése elképzelhető de elkerülni óhajtott helyzetekre (háború, természeti csapások, bűncselekmények)

4.1. Virtuális tulajdonságok

A virtuális objektumokat olyan tulajdonságokkal kell felruházni, amelyek meghatározzák ezek megfelelő viselkedését az illető alkalmazásban előforduló helyzetek bármelyikében. Ezek a tulajdonságok a valódi világban egyértelműen a tárgyakhoz tartoznak (alak, halmazállapot, tömeg, elektromos töltés, szín. hőmérséklet), megváltoztatásuk nem mindig egyszerű feladat, a virtuálisban azonban tetszés szerint hozzárendelhetők az értelmezett objektumokhoz és tetszés szerint változtathatók.

Az alábbi, BILIARD.PAS program egy biliárdjáték forráskódja. A "golyók" (tulajdonképen körlapok) a következő tulajdonságokkal rendelkeznek:

szilárd halmazállapotúak, alakjuk nem változik, más objektumokkal (golyó vagy a pálya széle) való áthatás kizárt
tökéletesen rugalmasak, ütközéskor nincs energiavesztés
"guruláskor" lassulnak - mintegy a "súrlódás" hatására
nem hagyják el a pálya síkját (kétdimenziós - síkbeli - modell)

A valódi biliárdjáték itt figyelmen kívűl hagyott jelensége például a golyók perdülete, amelynek hatására pályájuk görbe is lehet.

A golyók mozgásba hozása ("megütése") úgy történik, hogy a megfelelő golyót az egérrel "megfogjuk" (lenyomva az egér bal gombját), a kívánt irányba a kívánt kezdősebességgel arányos távolságba elmozdítjuk, majd "elengedjük" (felengedve az egérgombot).

```
{$N+}
uses crt, dos, graph, grafind, eger kez, kozos;

const
  n=11; r=8; ta=0.031;
  xmin=36+r;    xmax=603-r;    ymin=36+r;    ymax=443-r;
  sg=0.998; {energiavesztés (fékeződés) guruláskor }

type

  pontr = object (valós koordinátájú síkbeli pont objektum)
    x, y: real;
    procedure uj(ux, uy: real);
  end;

  golyo = object
    sz: byte; p:
    pointer;
```

```
    hr, hk, v: pontr;
    h, hg: pont;
    procedure fal;
    procedure rajz;
    procedure megj;
   end;

var
   a: array[1..n] of golyo;
   mm: word;
   i, j, k, 1, xa, w: integer;
   fl, ct, fm, kp: byte;

procedure pontr.uj(ux, uy: real);
begin
   x:=ux; y:=uy; end;

procedure golyo.rajz; (első megjelenítés és bitmap tárolása)
begin
   sz:=i; (sorszám - szín összerendelés)
   setcolor(sz); setfillstyle(1,sz); fillellipse(h.xp,h.yp,r,r);
   getmem(p, mm);
   getimage(h.xp-r,h.yp-r,h.xp+r,h.yp+r,p^);
end;

procedure golyo.megj; (megjelenítés)
begin
   putimage(h.xp-r,h.yp-r,p^,xorput);
end;

procedure golyo.fal; (fallal való ütközések kezelése)
begin
   if hr.x<xmin then begin hr.x:=2*xmin-hr.x; v.x:=-v.x; end;
   if hr.x>xmax then begin hr.x:=2*xmax-hr.x; v.x:=-v.x; end;
   if hr.y<ymin then begin hr.y:=2*ymin-hr.y; v.y:=-v.y; end;
   if hr.y>ymax then begin hr.y:=2*ymax-hr.y; v.y:=-v.y; end;
end;

procedure utkoz; (golyók egymással való ütközésének kezelése)
var
   u, wk, w, vni, vti, vnj, vtj, xi, yi, xj, yj, t:
real; begin
   for j:=i+1 to n do with a[j] do begin
       w:=sqr(a[i].hr.x-hr.x)+sqr(a[i].hr.y-
       hr.y); if w<4*r*r then begin
        w:=sqrt(w);
        wk:=sqrt(sqr(a[i].hk.x-hk.x)+sqr(a[i].hk.y-hk.y));
        t:=to*(2*r-w)/(wk-w); xi:=((2*r-w)*a[i].hk.x+(wk-
        2*r)*a[i].hr.x)/(wk-w); yi:=((2*r-
        w)*a[i].hk.y+(wk-2*r)*a[i].hr.y)/(wk-w);
        xj:=((2*r-w)*hk.x+(wk-2*r)*hr.x)/(wk-w);
        yj:=((2*r-w)*hk.y+(wk-2*r)*hr.y)/(wk-w);
        vni:=(a[i].v.x*(xj-xi)+a[i].v.y*(yi-yj))/2/r;
```

```
          vti:=(-a[i].v.x*(yi-yj)+a[i].v.y*(xj-
          xi))/2/r; vnj:=(v.x*(xj-xi)+v.y*(yi-yj))/2/r;
          vtj:=(-v.x*(yi-yj)+v.y*(xj-xi))/2/r; u:=vnj;
          vnj:=vni; vni:=u;
            {sebességek ütközés után)
          a[i].v.uj((vni*(xj-xi)-vti*(yi-yj))/2/r,
                (vni*(yi-yj)+vti*(xj-xi))/2/r);
               v.uj((vnj*(xj-xi)-vtj*(yi-yj))/2/r,
               (vnj*(yi-yj)+vtj*(xj-xi))/2/r);
               {ütközött golyók új helyei)
          a[i].hr.uj(xi+a[i].v.x*t,yi-a[i].v.y*t);
          hr.uj(xj+v.x*t,yj-v.y*t);
          fl:=1; ct:=ct+1;
        end;
      end;
end;

procedure kezd;
begin
   for i:=1 to n do begin
     a[i].h.uj(320,240); a[i].rajz; cleardevice;
   end; xa:=round(r*sqrt(3));
       {golyók kiindulási helyzetei)
   a[1].h.uj(60,240); a[2].h.uj(400,240); a[3].h.uj(400+xa,240-
   r); a[4].h.uj(400+xa,240+r); a[5].h.uj(400+2*xa,240+2*r);
   a[6].h.uj(400+2*xa,240); a[7].h.uj(400+2*xa,240-2*r);
   a[8].h.uj(400+3*xa,240-3*r); a[9].h.uj(400+3*xa,240-r);
   a[10].h.uj(400+3*xa,240+r); a[11].h.uj(400+3*xa,240+3*r); for
   i:=1 to n do a[i].hr.uj(a[i].h.xp,a[i].h.yp);
       {pályamegjelenítés)
   setfillstyle(1,11); bar(0,0,getmaxx,getmaxy);
   setfillstyle(1,15); bar(35,35,getmaxx-35,getmaxy-
   35); bkeret(30,30,getmaxx-30,getmaxy-30,6,6);
   setcolor(3); line(38,getmaxy-36,getmaxx-38,getmaxy-
   36); line(getmaxx-36,38,getmaxx-36,getmaxy-38);
end;

procedure dako; {dákó modellezése a megütendő golyó elmozdításával)
var kz: byte;
begin
   eger_1; w:=0; kz:=1; kp:=0;
   with reg do begin
     repeat {megütendő golyó kiválasztása)
       if keypressed then kp:=1;
       ax:=3; intr($33,reg);
       if (bx=1) and (getpixel(cx,dx)<15) then w:=15-getpixel(cx,dx);
     until (w>0) or (kp=1);
         {kiválasztott golyó sebessége irányának és nagyságának megadása
         a golyó képének elmozdításával)
     if (w>0) and (kp=0) then repeat
       ax:=3; intr($33,reg);
       if (bx=1) then begin
         if abs(cx-x)+abs(dx-y)>1 then begin
```

```
            eger_n;
            if lc_Z=0 then putimage(x-r,y-r,a[w].p^,xorput); kz:=0;
            a[w].v.uj(cx-a[w].hr.x,-dx+a[w].hr.y);
            putimage(cx-r,dx-r,a[w].p^,xorput);
            x:=cx; y:=dx;
            eger_l;
          end;
        end;
      until bx=0;
       eger_n; if kp=0 then putimage(x-r,y-r,a[w].p^,xorput);
    end;
  end;

  begin
    grindhi;
    eger_k; egert(38,38,getmaxx-38,getmaxy-38); eger_n;
    mm: =imagesize (0, 0, 2*r, 2*r) ; kezd; kp: =0;
    for i:=1 to n do a[i].megj;
    repeat
       for i:=1 to n do a[i].v.uj(0,0);
      dako;
       repeat
        fm:=0;
        for i:=1 to n do with a[i] do begin
          hk:=hr;
          v.uj(sg*v.x,sg*v.y); {lassulás modellezése}
          hr.uj(hr.x+v.x*ta,hr.y-v.y*ta); (új hely kiszámítása}
          fal; (esetleges fallal való ütközés kezelése)
        end;
        ct:=0;
        repeat (esetleges egymással való ütközések kezelése)
          f1:=0;
          for i:=1 to n do utkoz;
        until (f1=0) or (ct>30);
        for i:=1 to n do with a[i] do begin
          if (abs(hr.x-hg.xp)>1) or (abs(hr.y-hg.yp)>1) then begin
             megj; (golyó eltűnik a régi helyéről)
             h.uj(round(hr.x),round(hr.y));
             megj; (golyó megjelenik az új helyén)
          end;
          hg : =h;
          if abs(v.x)+abs(v.y)>0.2 then fm:=1;
        end;
      until fm=0;
    until kp=1;
  end.
```

4.2. Virtuális szerkezetek

A számítógép virtuális világában létrehozhatunk olyan szerkezeteket is, amelyeket nehezen, vagy egyáltalán nem lehet a valóságban elkészíteni. Ez azért lehetséges, mert a virtuális valóságot a programozó határozza meg, annak törvényeivel együtt. Ezek lehetnek a valódi világ természeti törvényeinek megfelelők, de lehetnek ezektől teljesen különbözők is. Amennyiben valódi jelenségeket modellezünk, vigyázni kell, hogy ne hagyjunk ki meghatározó jellegű folyamatokat. Ha például egy, a Föld légterébe kerülő meteorit mozgását vizsgáljuk, nem elég a súrlódás mozgástani hatását (fékeződés) figyelembe venni, ugyanis a hőtani hatás a meteorit olvadását és szétesését okozhatja.

A GYONGY_K.PAS nevű program egy logikai játék forráskódja. Színes golyók többféle képpen elmozdíthatók, ami a konfiguráció megváltozásához vezet. Ezt átmérős és gyűrű alakú "fiókok" eltolásával illetve forgatásával lehet elérni. A mozgatási parancsokat billentyűzetről vagy a megfelelő vizuális bil- lentyű egérrel való lenyomásával lehet megadni.

```
{$N+)
program gyongy_k;
uses
  crt, dos, graph, grafind, eger kez, kerdesek, kozos, file kez;
const
  r0=14; r1=16; r2=8; xk=240; yk=260;
  r:array(2..10] of integer =
                      (2*r1,3*r1,4*r1,5*r1,6*r1,7*r1,8*r1,9*r1,10*rl);
  fx=510; fy=360; fs=1.2; re=round(rl/fs);
  v_tol: kars =
  valasz: kars = ['A'..'F','I'..'N','Q','X'];
  gb: array [ 1 .. 2 ] of byte = (0, 0) ;

type
  konf = record
    a: array[1..5,1..6] of byte;
    a0: byte;
    h: array[1..6] of shortint;
  end;

var
  ff: file of konf;
  k, 1, x1, y1, rk, rl, rm: integer;
  cs, sn: array[1..6] of real;
  kf, kz, kt: konf;      {kf: aktív; kz: kezdő; kt: tárolt)
  sz: byte;
  f: array [ 1..100 ] of char;
```

```
procedure parancs; {vizuális billentyűk értelmezése}
begin
  settextstyle(0,0,1);
  gomb(400,21,'X','-bemut.',gb[1]); gomb(400,38,'Q','-
  kilép',gb[2]);
end;

procedure cim; {a játék neve és szerzője}
const
  cx=560; cy=30;
begin
  setcolor(0);
  setfillstyle(1,10); fillellipse(cx, cy, 65, 20);
  setfillstyle(1,15); fillellipse(cx, cy, 57, 15);
  setcolor(12); outtextxy(cx-50,cy-4,'Emoke játékai');
  outtextxy(cx-34,cy-6,'"'); setcolor(5); settextstyle(1,0,4);
  outtextxy(10,10,'GYONGYKEREK');
  settextstyle(1,1,4); outtextxy(39,7,':');
  settextstyle(0,0,1); outtextxy(200,10,'/');
  setcolor(5); outtextxy(5,468,'C Füzi János'); circle(8,471,7);
end;

procedure felirat; (a forgatásokat vezérlő vizuális billentyűk
                    megjelenítése)
var w: integer;
begin
  settextstyle(1,0,1); setlinestyle(0,0,1); setcolor(0);
  circle (fx, fy, re) ;
  for k:=1 to 3 do begin
   setlinestyle(0,0,1); setcolor(0);
   circle(fx,fy,(2*k+1)*re);
   w:=round(2*k*re);
   setlinestyle(0,0,3); setcolor(12);
   outtextxy(fx+w-2,fy-10,chr(72+2*k)); outtextxy(fx-w-9,
   fy-10,chr(71+2*k));
   setcolor(7); line(fx+w-8,fy-10,fx+w-8,fy+10);
   moveto(fx+w-8,fy+10); linerel(-3,-7); moveto(fx+w-8,fy+10);
   linerel(3,-7);
   line(fx-w+8,fy-10,fx-w+8,fy+10);
   moveto(fx-w+8,fy+10); linerel(-3,-7); moveto(fx-w+8,fy+10);
   linerel(3,-7);
  end;
  setlinestyle(0,0,1); setcolor(0);
  line(fx-90,fy-re,fx+90,fy-re); line(fx-90,fy+re,fx+90,fy+re);
end;

procedure fiok(szl, sz2, sz3: byte);
     {az eltolási műveleteket végző "fiókok" és a kezelésükhöz
   használt vizuális billentyűk megjelenítése}
begin
  rl:=r[2]-5;
  for 1:=1 to 6 do with kf do begin
   setlinestyle(0,0,1); setcolor(szl);
```

```
     moveto(xk+round(rl*cs[1]+rl*sn[1]),yk+round(-rl*sn[1]+r1*cs[1]));
     lineto(xk+round(rk*cs[1]+rl*sn[1]),yk+round(-rk*sn[1]+rl*cs[1]));
     lineto(xk+round(rk*cs[1]-rl*sn[1]),yk+round(-rk*sn[1]-rl*cs[1]));
     lineto(xk+round(rl*cs[1]-rl*sn[1]),yk+round(-rl*sn[1]-rl*cs[1]));
     line(xk+round(rm*cs[1]+rl*sn[1]),yk+round(-rm*sn[1]+rl*cs[1]),
        xk+round(rm*cs[1]-rl*sn[1]),yk+round(-rm*sn[1]-
     rl*cs[1])); setlinestyle(0,0,3); setcolor(sz2);
     moveto(xk+round((rk-2)*cs[1]+r2*sn[1]),yk+round(-
        (rk-2)*sn[1]+r2*cs[1]));
     lineto(xk+round((rm+2)*cs[1]+r2*sn[1]),yk+
        round(-(rm+2)*sn[1]+r2*cs[1]));
     linerel(round(7*cs[1]+3*sn[1]),round(-7*sn[1]+3*cs[1]));
     moveto(xk+round((rm+2)*cs[1]+r2*sn[1]),yk+
        round(-(rm+2)*sn[1]+r2*cs[1]));
     linerel(round(7*cs[1]-3*sn[1]),round(-7*sn[1]-3*cs[1]));
     setcolor(sz3); settextstyle(1,0,1);
     case 1 of
     1: moverel(-15,-20);
     2: moverel(-20,-15);
     3: moverel(-15,0);
     5: moverel(10,-5);
     6: moverel(5,-20);
     end;
     outtext(chr(64+1));
  end;
end;

procedure rajz; (a játék mindenkori állásnak megfelelő megjelenítése)
begin
  with kf do begin
     setlinestyle(0,0,1); setcolor(0);
     setfillstyle(1,a0); fillellipse(xk,yk,r0,r0);
     for k:=1 to 5 do begin
      if k<4 then circle(xk,yk,r[2*k+1]);
      for 1:=1 to 6 do begin
        x1:=round(xk+r[2*k]*cs[1]); y1:=round(yk-r[2*k]*sn[1]);
        setfillstyle(1,a[k,1]); if a[k,1]=15 then setcolor(15);
        fillellipse(xl,yl,r0,r0); setcolor(0); end;
     end;
     fiok(0,7,12);
  end;
end;

procedure tol(p:byte); (eltolási művelet)
begin
  with kf do
     if h[p]>-1 then begin
      fiok(15,15,15); (az elmozdított "fiók" régi alakjának törlése
                       a háttér színével való rárajzolással)
      h[p] :=h[p]-2;
      if p<4 then begin
                    h[p+3]:=h[p+3]+2;
```

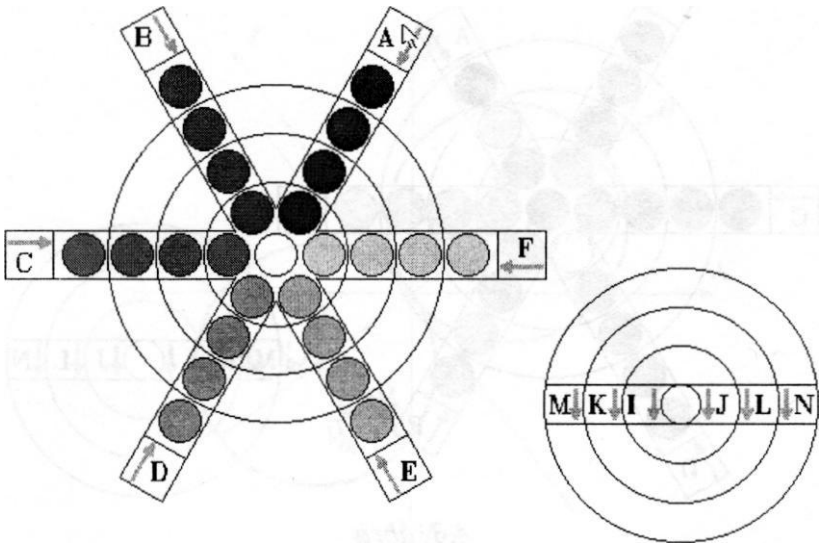
```
                    for k:=4 downto 1 do a[k+1,p+3]:=a[k,p+3];
                    a[1,p+3]:=a0; end
                else begin
                    h[p-3]:=h[p-3]+2;
                    for k:=4 downto 1 do a[k+1,p-3]:=a[k,p-3];
                    a[1,p-3]:=a0; end;
        a0:=a[1,p];
        for k:=1 to 4 do a[k,p]:=a[k+1,p];
        a[5,p]:=15;
     end;
end;

procedure forg(p,q:byte); (forgatási művelet)
var sd: byte;
begin
   with kf do begin
     case q of
     1:begin
         sd:=a [p, 6] ;
         for 1:=5 downto 1 do a[p,1+1]:=a[p,1];
         a[p,1]:=sd; end;
     2:begin
         sd:=a[p,1];
         for 1:=1 to 5 do a[p,1]:=a[p,1+1];
         a [p, 6 ] : =sd; end;
     end;
   end;
end;

procedure muvelet; {műveletek és "forró" betűk összerendelése)
begin
   case qr of
   'A':tol(1); 'B':tol(2); 'C':tol(3); 'D':tol(4); 'E':tol(5);
   'F':tol(6);
   'I':forg(1,1); 'J':forg(1,2); 'K':forg(2,1);
   'L':forg(2,2); 'M':forg(3,1); 'N':forg(3,2);
   end;
end;

procedure feladat; (feladatgenerálás)
var w: integer;
begin
   randomize; sz:=0;
   with kf do begin
     for w:=1 to 97 do begin
        qr:=chr(65+random(6)+8*random(2));
        if not (qr in v_tol) or (h[ord(qr)-64]>-1) then begin
          sz:=sz+1; f[sz]:=qr; muvelet; end;
     end;
```

```
  for w:=1 to 6 do
      if h[w]=3 then begin
         qr:=chr(64+w); sz:=sz+1; f[sz]:=qr; muvelet;
    end;
  end;
end;

procedure kezd; (kezdőállás (megoldás))
begin
  with kz do begin
    a0:=7;
    for 1:=1 to 6 do begin
      a[5,1]:=15;
      for k:=1 to 4 do a[k,1]:=8+1;
      h(1]:=1; end;
  end;
end;

procedure megold; {bemutató műveletsor - a feladatgenerálás fordított
                   sorrendben való megjelenítése}
var w: integer;
begin
  kezd; kf:=kz; feladat; rajz; delay(3000);
  for w:=sz downto 1 do begin
    case f [w] of
     'A':qr:='D';              'B':qr:='E';
     'C':qr:='E';              'D':qr:='A';
     'E':qr:='B';              'F':qr:='C';
     'I':qr:='J';              'J':qr:='I';
    end; muvelet; rajz; delay(600);
    end; delay(3000);
end;

procedure kilep; (kilépés)
begin
  qm:='
  kerdez(200,200,'    Menteni a konfigurációt ?','gen','em','I','N');
  repeat kattint(200,200,'I','N',qm);
  until (qm='I') or (qm='N');
  if qm='I' then begin
    eger_n; ment('GYK'); (konfiguráció tárolása)
    if v='I' then begin
      assign(ff,kn); rewrite(ff); write(ff,kf); close(ff); end;
    eger_1;
  end;
end;

procedure konfig; (konfigurációkiválasztás)
begin
  settextstyle(0,0,1);
  gj     ,;
  kerdez(200,200,'        Konfiguráció ?','áték','anulás','J','T');
```

```
  repeat kattint(200,200,'J','T',qj); until (qj='J') or (qj='T');
  if qj='T' then kf:=kz
  else repeat
     qf:=' ';
     kerdez(200,200,'       Konfiguráció
     repeat kattint(200,200,'U','T',qf);
     until (qf='U') or (qf='T');
     case of of
     'T': begin
       eger_n; olvas('GYK'); (tárolt konfiguráció betöltése)
       if ko=1 then begin
        assign(ff,kn); reset(ff); read(ff,kf); close(ff); end;
       eger_1;
     end;
     'U': begin kf:=kz; feladat; ko:=1;
     end; end;
  until ko=1;
end;

procedure par_katt(var vl: char); (parancs kiválasztása)
var
  fn: byte;
  x, y: real;
begin
                  {egérrel}
  gb_katt(400,21,fn); if fn=1 then vl:='X';
  gbkatt(400,38,fn); if fn=1 then vl:='Q';
  with reg do begin
     ax:=3; intr($33,reg);
     if bx=1 then begin
        for 1:=1 to 6 do with kf do begin {eltolások vezérlése egérrel)
        rk:=r[10]+h[1]*r1+4; rm:=r[8]+h[1]*r1+4;
        if sqr(cx-(xk+round((rk-r1)*cs[1])))+
           sqr(dx-(yk-round((rk-r1)*sn[1])))<400 then v1:=chr(64+1);
      end;
      for 1:=1 to 3 do begin            {forgatások vezérlése egérrel)
        y:=fy;
        x:=fx+2*1*re;
        if sqr(cx-x)+sqr(dx-y)<144 then vl:=chr(72+2*1);
        x:=fx-2*1*re;
        if sqr(cx-x)+sqr(dx-y)<144 then v1:=chr(70+2*1+1);
      end;
     end;
  end;
if keypressed then v1:=upcase(readkey); {billentyűzetről)
end;

procedure mezo;
begin
  setfillstyle(1,15); bar(14,64,625,456);
  setcolor(3); line(16,456,623,456); line(625,66,625,454);
end;
```

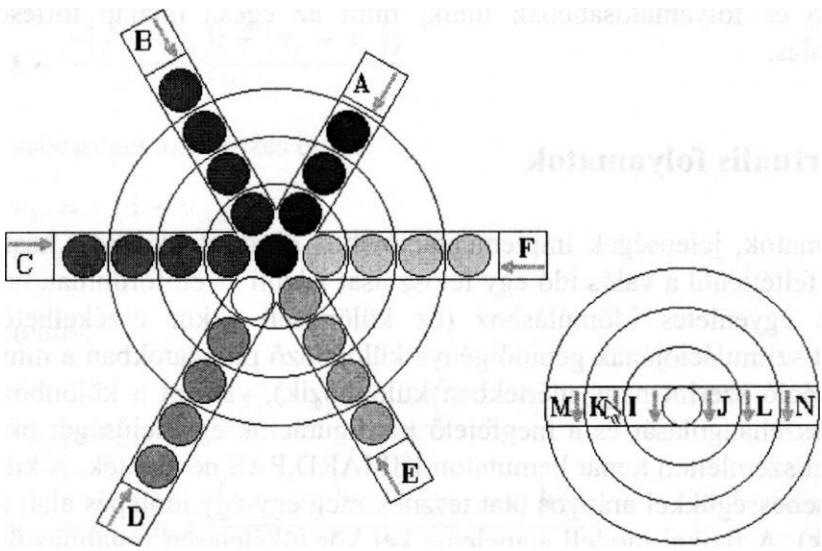
```
procedure alias;
begin
  eger_n; eger_t(0,0,getmaxx,getmaxy); mezo;
  parancs; felirat; rajz; eger_l;
end;

begin
  grindhi; kezd;
  for k:=1 to 6 do begin cs[k]:=cos(k*pi/3); sn[k]:=sin(k*pi/3); end;
  eger_k; eger_n;
      setfillstyle(1,11); bar(0,0,640,480); bkeret(10,60,629,460,4,4);
cim; eger_1;
  repeat
     eger_n; qu:='N'; qk:='N'; mezo; konfig; alias;
     repeat
       qr:=' '; repeat par_katt(qr); until qr in valasz; eger_n;
       case qr of

       'X': begin (bemutató animáció)
              gb[1]:=1; parancs;
              kt : =kf; megold; kf : =kt; rajz;
              gb(i]:=0; parancs;
            end;

       'Q': begin {kilépés}
              gb[2]:=1; parancs;
              repeat kilep until not ((v='N') and (qm='I'));
              vege; if qk='N' then ujkonf; gb(2):=0; parancs; alias;
            end;
       else begin muvelet; rajz; end;
       end;
       eger 1;

     until (qk='I') or (qu='I');
  until qu=' N' ;
  closegraph;
end.
```

A játék megoldott állapotban, fekete-fehéren, a *4.1.ábrán* látható.

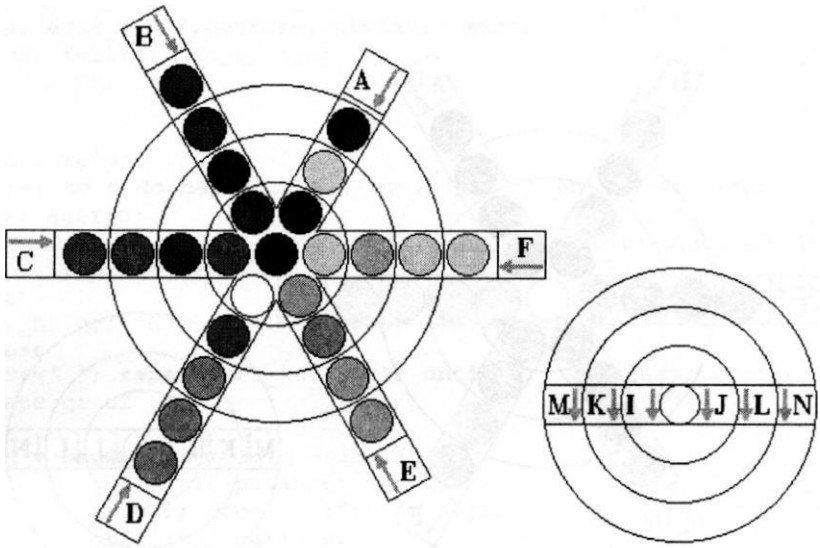


4.1. ábra

Az A parancs hatására a megfelelő átmérős fiók eltolásának eredményét mutatja a *4.2. ábra.* A *4.3. ábrán* a középső gyűrű alakú fiók órajárással ellentétes irányba való elfordításának eredménye látható (K parancs).



4.2. ábra



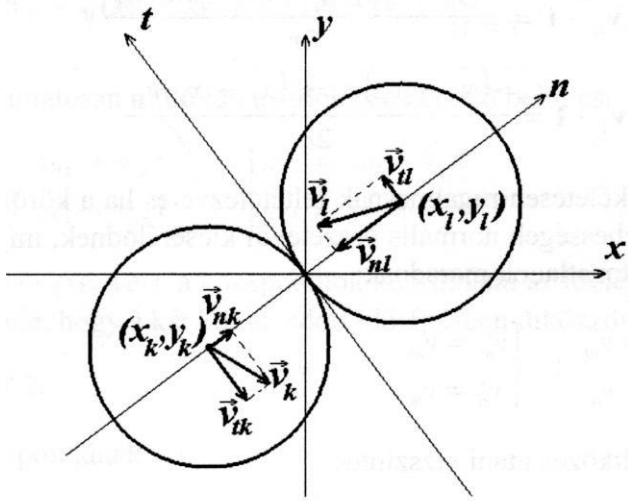
4.3. ábra

A megjelenítés úgy történik, hogy a szerkezetet a háttér színével (fehér) a régi helyzetben újrarajzoljuk (ezáltal eltűnik), majd megrajzoljuk az új helyzetben a megfelelő színekkel. A golyókat egyszerűen rárajzoljuk az új helyre - ezáltal elfedve az előzőleg ott álló golyót (megjelenítési szempontból van még egy-egy fehér színű külső golyó, az átmérős eltolásokhoz). Ezzel a megjelenítés gyorsabb és folyamatosabbnak tűnik, mint az egész rajzlap törlése és az újrarajzolás.

4.3. Virtuális folyamatok

A folyamatok, jelenségek implementációjához diszkrét időlépést használunk. Ez nem feltétlenül a valós idő egy felosztását jelenti és előfordulhat, hogy nem is vezet egyenletes időmúláshoz (ez különösen akkor érzékelhető, ha a folyamat szimulációjának gépidőigénye különböző pillanatokban a mindenkori konfiguráció szerint nagy mértékben különbözik), viszont a különböző mozgások összehangolását és a megfelelő konfigurációk egyidejűségét biztosítani lehet. Ezt szemlélteti a már bemutatott BILIARD.PAS nevű játék. A különböző golyók sebességükkel arányos utat tesznek meg egy-egy időlépés alatt (ha nem ütköznek). A fizikai modell alapeleme két kör tökéletesen rugalmas ütközése. Az ütközés pillanatában a sebességeknek az érintési pontban a közös érintőre

merőleges összetevőire érvényes az impulzusmegmaradás elve, míg az érintőmenti összetevők változatlanok maradnak *(4.4. ábra)*.



4.4. ábra

A közös érintőre merőleges (normális) tengely (átmegy mindkét kör középpontján), illetve az érintő tengely irányvektorainak képletei:

$$\mathbf{n} = \frac{(x_l - x_k)\mathbf{i} + (y_l - y_k)\mathbf{j}}{2r}$$
$$\mathbf{t} = \frac{-(y_l - y_k)\mathbf{i} + (x_l - x_k)\mathbf{j}}{2r} \qquad (4.1$$

A körök sebességei az ütközés

$$\mathbf{v}_k = v_{xk}\mathbf{i} + v_{yk}\mathbf{j}$$
$$\mathbf{v}_l = v_{xl}\mathbf{i} + v_{yl}\mathbf{j} \qquad (4.2$$

Ezek ...

$$v_{nk} = \mathbf{v}_k \cdot \mathbf{n} = \frac{(x_l - x_k)v_{xk} + (y_l - y_k)v_{yk}}{2r}$$
$$v_{nl} = \mathbf{v}_l \cdot \mathbf{n} = \frac{(x_l - x_k)v_{xl} + (y_l - y_k)v_{yl}}{2r}, \qquad (4.3$$

illetve érintőmenti összetevői:

$$v_{tk} = \mathbf{v}_k \cdot \mathbf{t} = \frac{-(y_l - y_k)v_{xk} + (x_l - x_k)v_{yk}}{2r}$$
$$v_{tl} = \mathbf{v}_l \cdot \mathbf{t} = \frac{-(y_l - y_k)v_{xl} + (x_l - x_k)v_{yl}}{2r} \qquad (4.4)$$

Az ütközést tökéletesen rugalmasnak feltételezve és ha a körök "tömege" azonos, akkor a sebességek normális összetevői kicserélődnek, míg az érintőmenti összetevők változatlanok maradnak:

$$\begin{cases} v'_{nk} = v_{nl} \\ v'_{tk} = v_{tk} \end{cases} ; \quad \begin{cases} v'_{nl} = v_{nk} \\ v'_{tl} = v_{tl} \end{cases} . \qquad (4.5$$

A sebességek ütközés utáni vízszintes

$$v'_{xk} = v'_{nk}\mathbf{n} \cdot \mathbf{i} + v'_{tk}\mathbf{t} \cdot \mathbf{j} = \frac{(x_l - x_k)v'_{nk} - (y_l - y_k)v'_{tk}}{2r}$$
$$v'_{xl} = v'_{nl}\mathbf{n} \cdot \mathbf{i} + v'_{tl}\mathbf{t} \cdot \mathbf{j} = \frac{(x_l - x_k)v'_{nl} - (y_l - y_k)v'_{tl}}{2r} \qquad (4.6$$

illetve függőleges

$$v'_{yk} = v'_{nk}\mathbf{n} \cdot \mathbf{j} + v'_{tk}\mathbf{t} \cdot \mathbf{i} = \frac{(y_l - y_k)v'_{nk} + (x_l - x_k)v'_{tk}}{2r}$$
$$v'_{yl} = v'_{nl}\mathbf{n} \cdot \mathbf{j} + v'_{tl}\mathbf{t} \cdot \mathbf{i} = \frac{(y_l - y_k)v'_{nl} + (x_l - x_k)v'_{tl}}{2r} \qquad (4.7$$

A rajzolóprogram a köröket bizonyos időlépéssel (i) jeleníti meg. A körök helyzetének kiszámítása az ütközések pillanatában a következő módon történik:

a két kör középpontja közötti távolság az időlépés elején:

$$de_{kl} = \sqrt{(xe_l - xe_k)^2 + (ye_l - ye_k)^2}\Big|_{t=0} , \qquad (4.8)$$

illetve végén:

$$dv_{kl} = \sqrt{\left(xv_l - xv_k\right)^2 + \left(yv_l - yv_k\right)^2}\Bigg|_{t=\tau}, \qquad (4.9$$

ahol *t* a folyamatosan múló idő az időlépés keretén belül és:

$$\begin{cases} xv_k = xe_k + v_{xk}\tau \\ yv_k = ye_k + v_{yk}\tau \end{cases}; \quad \begin{cases} xv_l = xe_l + v_{xl}\tau \\ yv_l = ye_l + v_{yl}\tau \end{cases}, \qquad (4.10$$

ahol (x_{ek},y_{ek}) és (x_{el},y_{el}) a középpontok koordinátái az időlépés elején. annak a feltétele, hogy a két kör az adott időlépésben ütközzön:

$$dv_{kl} \le 2r \qquad (4.11$$

- az ütközés időpontj

$$t_0 = \frac{de_{kl} - 2r}{de_{kl} - dv_{kl}}, \qquad (4.12$$

illetve a középpontok ütközés pillanatában való helyzetének kiszámítása:

$$\begin{cases} x_k = \dfrac{\left(2r - dv_{kl}\right)xe_k + \left(de_{kl} - 2r\right)xv_k}{de_{kl} - dv_{kl}} \\ y_k = \dfrac{\left(2r - dv_{kl}\right)ye_k + \left(de_{kl} - 2r\right)yv_k}{de_{kl} - dv_{kl}} \end{cases} \quad \begin{cases} x_l = \dfrac{\left(2r - dv_{kl}\right)xe_l + \left(de_{kl} - 2r\right)xv_l}{de_{kl} - dv_{kl}} \\ y_l = \dfrac{\left(2r - dv_{kl}\right)ye_l + \left(de_{kl} - 2r\right)yv_l}{de_{kl} - dv_{kl}} \end{cases}. \qquad (4.13)$$

A körök középpontjainak koordinátái az időlépés végén tekintetbe véve az időlépésben lezajlott ütközést:

$$\begin{cases} xv_k = x_k + v'_{xk}\left(\tau - t_0\right) \\ yv_k = y_k + v'_{yk}\left(\tau - t_0\right) \end{cases}; \quad \begin{cases} xv_l = xe_l + v'_{xl}\left(\tau - t_0\right) \\ yv_l = ye_l + v'_{yl}\left(\tau - t_0\right) \end{cases}. \qquad (4.14$$

A pálya falaival való ütközések számítása egyszerűbb, ugyanis a fal merev, így a golyó sebességének a falra merőleges összetevője előjelt vált, a golyó helyzete az időlépés végén pedig a fal hiányában számított helyzetnek a fal felületé-

hez viszonyított tükörképe lesz. Így nem kell kiszámítani a fallal való ütközés pillanatát az időlépésen belül.

Többszörös ütközések hasonló módon kezelhetők, csupán a számítási eljárást kell ugyanabban az időlépésben megismételni, míg több ütközés nem történik.

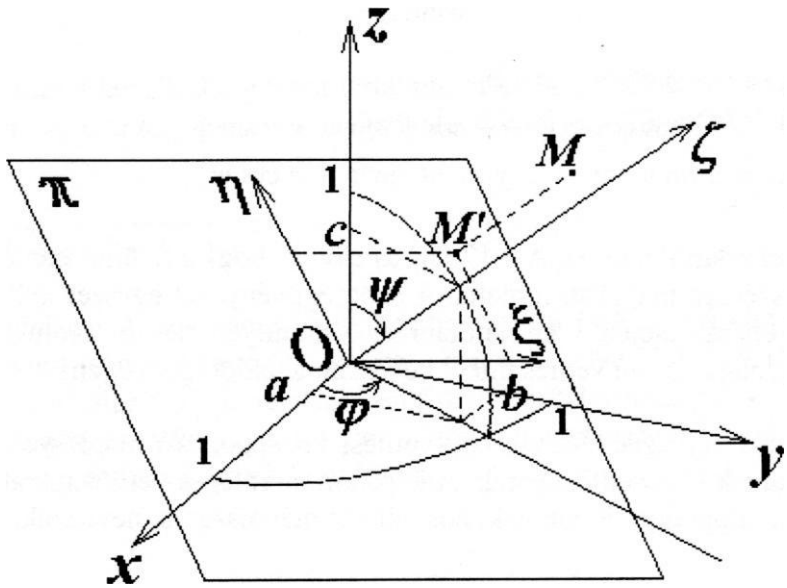
A valóságban egyenletesen lassuló mozgást itt exponenciális lassulással közelítettük (minden golyó sebessége egy l -nél kevéssel kisebb állandóval szorzódik minden időlépés végén), kiegészítve egy, ebben az esetben elengedhetetlen, megállási feltétellel.

4.4. Poliéderek megjelenítése

Bonyolult térbeli alakzatok megjelenítésének hatékony eszköze a poliéderes közelítés. A poliédereket sokszögek határolják, ezek vetületei a képsíkon ugyancsak sokszögek lesznek, és a *Turbo Pascal* igen előnyös sokszögmegjelenítési utasításokat bocsát a programozó rendelkezésére *(drawpoly* - sokszög kerületének meghúzása, *fillpoly* - adott színű sokszöglap rajzolása, *floodfill* - adott színű kontúr megtöltése adott színnel).

4.4.1. Merőleges és középpontos vetítés

A merőleges vetítés során az adott ponton átmenő, a térbeli *Oxyz* koordinátarendszer kezdőpontján átmenő képsíkra merőleges egyenest (vetítősugár) metsszük a képsíkkal. A képsíkon egy derékszögű Oξη koordinátarendszert értelmezünk, amelynek függőleges tengelye (Oη) a térbeli *Oz* tengely vetülete *(4.5. ábra).* Ezt a rendszert kiegészítjük egy, a képsíkra merőleges (a vetítősugarakkal párhuzamos) *O*ξ tengellyel, amelynek segítségével megadhatjuk a vetített pont képsíktól való távolságát (magasságát). Erre az egymást legalábbis részben takaró poliéderek megjelenítési sorrendjének meghatározásához lesz szükség.



4.5. ábra

A vetítési irányt megadhatjuk ennek irányítényezőivel *(a, b, c),* amelyeket, ha négyzetösszegük egységnyi, iránykoszinuszoknak is nevezünk, vagy pedig a gömbi koordinátarendszer szögeivel (φ és ψ):

$$\begin{cases} a = \sin\psi \cos\varphi \\ b = \sin\psi \sin\varphi \\ c = \cos\psi \end{cases} ; \quad a^2 + b^2 + c^2 = 1 ; \quad \begin{cases} \varphi \in [0,2\pi] \\ \psi \in [0,\pi] \end{cases}. \quad (4.1$$

Az *M (x,y,z)* pont M'(ξ,η) vetületének képsíkbeli koordinátái és az *M* pont képsíkhoz viszonyított magassága a

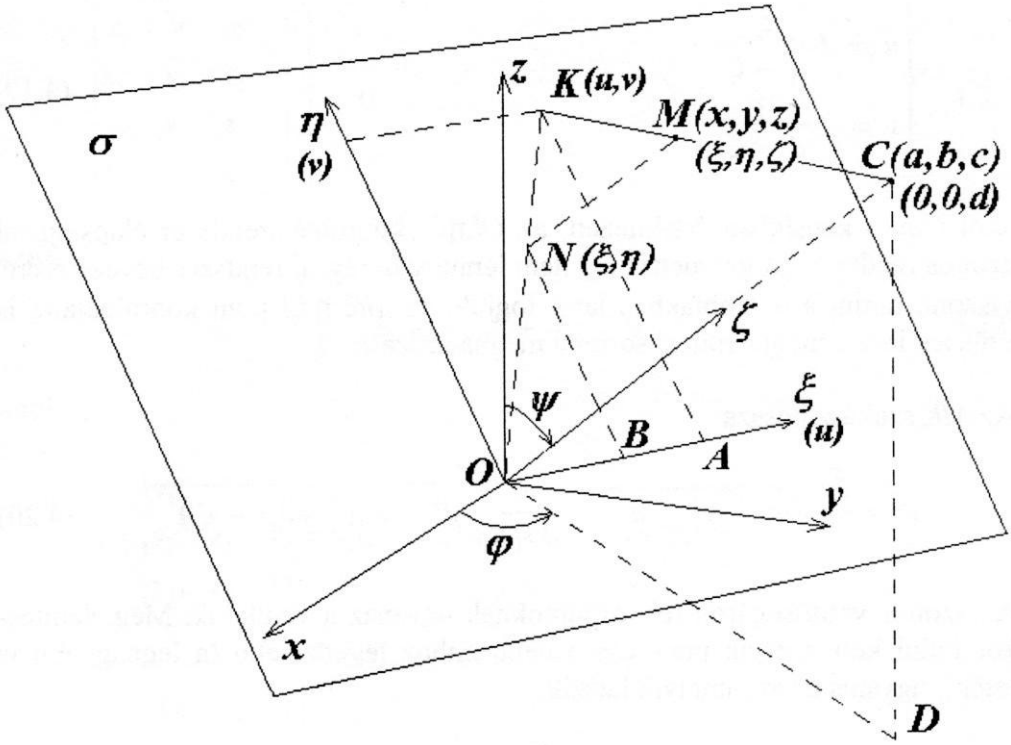
$$\begin{cases} \xi = \dfrac{-bx + ay}{\sqrt{a^2 + b^2}} \\ \eta = \dfrac{-c(ax + by) + (a^2 + b^2)z}{\sqrt{a^2 + b^2}} \\ \zeta = ax + by + cz \end{cases} \qquad (4.16)$$

vagy a

$$\begin{cases} \xi = -x \sin\varphi + y \cos\varphi \\ \eta = -x \cos\psi \cos\varphi - y \cos\psi \sin\varphi + z \sin\psi \\ \zeta = x \sin\psi \cos\varphi + y \sin\psi \sin\varphi + z \cos\psi \end{cases} \qquad (4.17)$$

képletekkel számíthatók ki. **A 4.17 felírás előnye, hogy a felülnézetet** *(a = b =* 0) természetszerűen magában foglalja. A szögfüggvényeket egyszer kell - a megjelenítési eljárás elején - kiszámítani és valamilyen néven tárolni, hogy az esetenként nagy számú vetítés minél kevesebb gépidőt igényeljen.

Középpontos vetítésnél létezik egy vetítési középpont (a megfigyelő szeme), ebből indulnak ki a vetítősugarak. A képsíkra merőleges vetítősugarat fősugárnak, ennek talppontját főpontnak, hosszát szemtávolságnak nevezzük.



4.6. ábra

A *4.6. ábrán* a képsík főpontként tartalmazza az *Oxyz* koordinátarendszer kezdőpontját. A *C(a, b, c)* centrum megadható gömbi koordinátákkal is (dm ,w), yi), ahol

$$d = \sqrt{a^2 + b^2 + c^2} \qquad (4.18$$

a szemtávolság. A merőleges vetítésnél bemutatott *(4.5. ábra)* ξηζ koordinátarendszert itt is alkalmazzuk. Ebben a centrum helye *C (0,0,d).* Az *M(x,y,z)* pont koordinátáit az Oξηζ rendszerben a 4.17 transzformáció adja. Az *MNK* és *COK* valamint az *NBO* és *KAO* háromszögek hasonlóságából adódnak az *M* pont *K* vetületének képsíkbeli koordinátái:

$$\begin{cases} u = d\,\dfrac{\xi}{d-\zeta} \\ v = d\,\dfrac{\eta}{d-\zeta} \end{cases}, \qquad (4.19)$$

ahol *Ouv* a képsíkban értelmezett, az Oξηζ koordinátarendszer alapsíkjával azonos rendszer. Itt geometriailag nem lenne szükség új rendszer bevezetésére, viszont, amint a továbbiakban látni fogjuk, az M(ξηζ) pont koordinátáira is szükség lesz a megjelenítési sorrend meghatározásánál.

Az *MK* szakasz hossza:

$$w = \frac{\zeta}{d}\sqrt{u^2 + v^2 + d^2} = \frac{\zeta}{d-\zeta}\sqrt{\xi^2 + \eta^2 + (d-\zeta)^2} \quad . \qquad (4.20$$

Az azonos vetítősugáron fekvő pontoknak ugyanaz a vetületük. Megjelenítéskor tudni kell, melyik pont van a centrumhoz legközelebb (a legnagyobb w érték), ugyanis ez az, amelyik látszik.

4.4.2. Irányított sokszögek megjelenítése

A sokszögek irányítását éleik körbejárási irányával (csúcsaik sorrendjével) adhatjuk meg. Például egy sokszögnek az a pozitív oldala, amelyik a körbejárási irányban forgatott jobbsodrású fúró haladási irányába mutat. Csúcsainak sorrendjét megszabva egyértelműen meghatározzuk a sokszög irányítását. Ha zárt poliéder egyik oldallapjáról van szó, akkor a pozitív oldal mindig a külső, ennek megfelelően kell megadni a csúcsok sorrendjét. Vagyis ha a poliéder lapját élei mentén körülsétáljuk úgy, hogy a poliéderen mint egy "bolygón" állunk, akkor a szóban forgó lap mindig bal kezünk felé esik.

Irányított sokszögekhez úgynevezett normálisvektort (a sokszögre merőleges, ennek pozitív irányába mutató egységvektort) rendelhetünk. Ha egy sokszög három nem kollineáris csúcsa pozitív sorrendben *A1(x1* y1 *z1),* A2(x2, y2, z*2),* A3(x3, y3, z*3),* akkor a sokszög síkjának egyenlete:

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0 \quad , \tag{4.21}$$

azaz

$$a_h x + b_h y + c_h z + d_h = 0 \ , \tag{4.22}$$

ahol:

$$a_h = \begin{vmatrix} y_1 & z_1 & 1 \\ y_2 & z_2 & 1 \\ y_3 & z_3 & 1 \end{vmatrix} = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$

$$b_h = -\begin{vmatrix} x_1 & z_1 & 1 \\ x_2 & z_2 & 1 \\ x_3 & z_3 & 1 \end{vmatrix} = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$

, (4.23)

$$c_h = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$

$$d_h = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} = x_1(y_3 z_2 - y_2 z_3) + y_1(z_3 x_2 - z_2 x_3) + z_1(x_3 y_2 - x_2 y_3)$$

a normálisvektor

$$\mathbf{n} = a_h \mathbf{i} + b_h \mathbf{j} + c_h \mathbf{k} \ . \tag{4.24}$$

Egy sokszögnek akkor látjuk a pozitív oldalát, ha a normálisvektor a vetítősu gár képsík felé mutató vektorával tompaszöget zár be. Zárt poliéderek lapjainak csak a pozitív oldaluk látható. Konvex poliéderek megjelenítéséhez a lapok irányítását tekintetbe vevő láthatósági teszt elegendő, ugyanis takarások nem fordulhatnak elő.

Az irányítás előjele azonban a vetítés során megmarad, ezért egy gyorsabb (ke vesebb műveletet igénylő) láthatósági tesztet alkalmazhatunk. Legyenek a fenti sokszög adott csúcsainak képsíkra eső (merőleges vagy középpontos) vetületei $A'_1(u_1, v_1)$, $A'_2(u_2, v_2)$, $A'_3(u_3, v_3)$. Akkor a vetületsokszög pozitív irányítású (normálisvektora a megfigyelő felé mutat), ha:

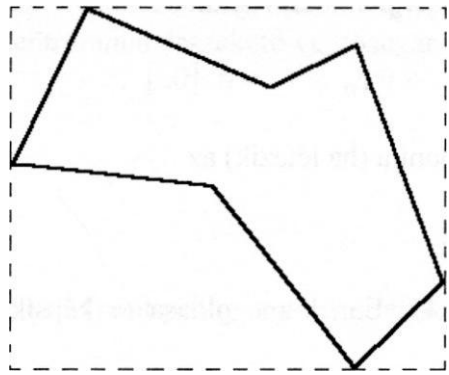
$$u_1(v_2 - v_3) + u_2(v_3 - v_1) + u_3(v_1 - v_2) > 0 \qquad (4.2$$

4.4.3. Takarásnak megfelelő megjelenítési sorrend

Egymást részben takaró sokszögekből álló ábrát úgy készíthetünk, hogy a látható sokszögeket a megfelelő sorrendben jelenítjük meg (az alul levő sokszöget előbb, a felül levőt később). A sorrendmegállapítást a következő módon végezhetjük:

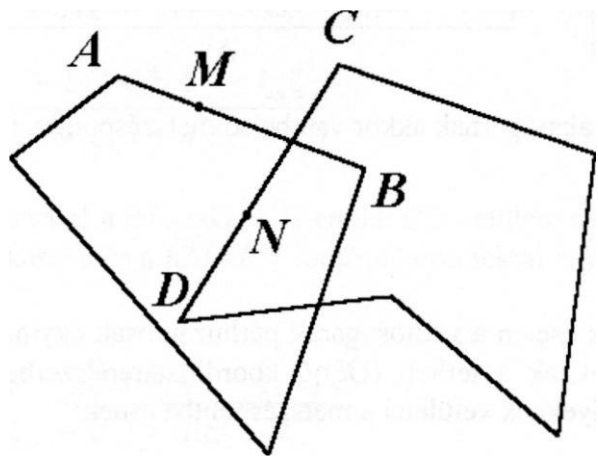
- először a sokszögeket a súlypontjuk képsíkhoz viszonyított magassága szerinti fordított sorrenbe helyezzük. Csillagszerű poliéderek (van olyan belső pontjuk, amelyből a poliéder minden lapjának a belső oldala látszik) megjelenítéséhez ez elegendő is. A számítások gyorsítása céljából itt a súlypont magasságát a csúcsok magasságának középarányosával helyettesíthetjük.
- nem csillagszerű poliéderek esetén meg kell vizsgálni a takarási viszonyokat. Olyan pontot keresünk, amely mindkét összehasonlítandó sokszög vetületének belsejében van, majd kiszámítjuk a sokszögbeli megfelelőinek magasságát (metsszük a sokszögek síkjait a vetítősugárral).

Bonyolult alakzatok esetén a takarási viszonyok megállapítása időigényes művelet, ezért érdemes az összehasonlító eljárást elkerülni azokra a sokszögpárokra, amelyeknek vetületei diszjunktak. Ennek egyik leggyorsabb módja, hogy a vetületeket téglalap alakú "dobozokba" zárjuk *(4.7. ábra)* és csak azokat az eseteket vizsgáljuk, amelyekben a dobozok legalábbis részben egymásra tevődnek.



4.7. ábra

Közös vetületpont keresésének egyik módja, hogy kiszámítjuk a vetületek éleinek metszéspontját *(4.8. ábra)*.



4.8. ábra

Két (kép)síkbeli szakasz közötti metszéspont meghatározása a következő módon történik:

Legyen *M* az *AB* szakaszon mozgó pont. Helyvektora (tulajdonképen az *AB* szakasz paraméteres előállítása):

$$\mathbf{r}_M = (1-u)\,\mathbf{r}_A + u\,\mathbf{r}_B \;; \qquad u \in [0,1] \;. \tag{4.26}$$

A *CD* szakaszon mozgó *N* pont

$$\mathbf{r}_N = (1 - v)\,\mathbf{r}_C + v\,\mathbf{r}_D \;; \qquad v \in [0,1] \;. \tag{4.2}$$

A két szakasz metszéspontja (ha létezik)

$$\mathbf{r}_M = \mathbf{r}_N$$

egyenlettel számítható ki. Ennek megoldása (a képsík *Oxy* koordinátarendszerében):

$$\Delta = (x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)$$

$$\Delta \neq 0: \quad \begin{cases} u = \dfrac{(x_C - x_A)(y_C - y_D) - (x_C - x_D)(y_C - y_A)}{\Delta} \\ v = \dfrac{(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)}{\Delta} \end{cases} \tag{4.29}$$

A szóbanforgó szakaszpárnak akkor van belső metszéspontja, ha:

$$\begin{cases} u \in (0,1) \\ v \in (0,1) \end{cases} . \tag{4.30}$$

Merőleges vetítés esetén a vetítősugarak párhuzamosak egymással, ezért meghatározhatók azoknak a térbeli (Oξηζ koordinátarendszerben) pontoknak a magasságai, amelyeknek vetületei a metszéspontba esnek:

ahol $\begin{cases} \zeta_m = (1 - u)\,\zeta_a + u\,\zeta_b \\ \zeta_n = (1 - v)\,\zeta_c + v\,\zeta_d \end{cases}$, kisbetűvel jelöltük azokat a térbeli pontokat, amelyeknek vetületeit a megfelelő nagybetűk jelölik. (4.31

Középpontos vetítésnél az itt kihasznált hasonlóság nem áll fenn. Ebben az esetben a vetítősugarat kell metszeni a két térbeli szakasszal. A *4.9. ábra* síkja a *K(0, 0, d)* centrumon és a képsíkban fekvő *AB* szakaszon (illetve az *ab*

szakaszon, amelynek ez a vetülete) megy át. Az OT r koordinátarendszerben az M(ξ,η,0*)* pontot a centrummal összekötő vetítősugár paraméteres előállítása:

$$\begin{cases} \xi = u\xi_M \\ \eta = u\eta_M \\ \zeta = (1-u)d \end{cases}, \qquad (4.32$$

ahol a *K* pont az $u = 0$, az M pont pedig az $u = 1$ értékeknek felel meg. Ezt a vetítősugarat metszve az *ab* szakasszal:

$$\mathbf{r} = (1-t)\,\mathbf{r}_a + t\,\mathbf{r}_b \;; \qquad t \in [0,1] \;, \qquad (4.33$$

az *m* pontnak a vetítősugáron megfelelő paraméter értékére a következő képletet kapjuk (az *ab* és *KM* szakaszok egysíkúságát a vetítési eljárás biztosítja):

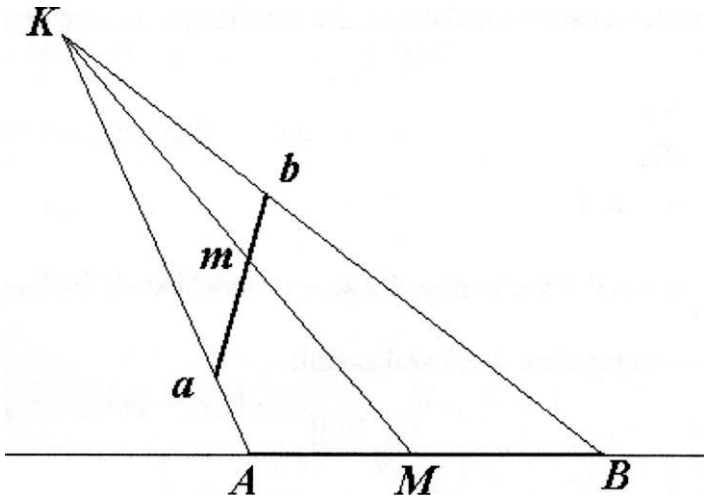
$$u = \frac{d(\xi_b - \xi_a) + \xi_a\zeta_b - \xi_b\zeta_a}{\xi_M(\zeta_b - \zeta_a) + d(\xi_b - \xi_a)} \;. \qquad (4.34$$

Hasonlóképen járva el a *cd* szakasz és ennek *CD* vetülete esetében, meghatározható a *cd* szakaszon és a *KM =KN* vetítősugáron fekvő *n* pontnak megfelelő v paraméterérték:

$$v = \frac{d(\xi_d - \xi_c) + \xi_c\zeta_d - \xi_d\zeta_c}{\xi_N(\zeta_d - \zeta_c) + d(\xi_d - \xi_c)} \;. \qquad (4.35$$

Az *m* és *n* pontok közül tehát az van a vetítősugáron a centrumhoz közelebb, amelyikre kisebb paraméterérték adódik. Következésképpen azt a sokszöget kell utóbb megjeleníteni, amelynek az élén a centrumhoz közelebbi pont fekszik.

Az egyik sokszög vetületének minden élét a másik sokszög vetületének minden élével össze kell hasonlítani addig, amíg egy metszéspontot találunk, amelyre a fent leírt módon a sorrendet meg tudjuk állapítani. A valódi metszéspontokra (ahol a vetületek metszéspontja pontosan a térbeli *ab* és *cd* szakaszok metszéspontjának vetülete) ez nem megtehető, ilyenkor tovább kell keresni.

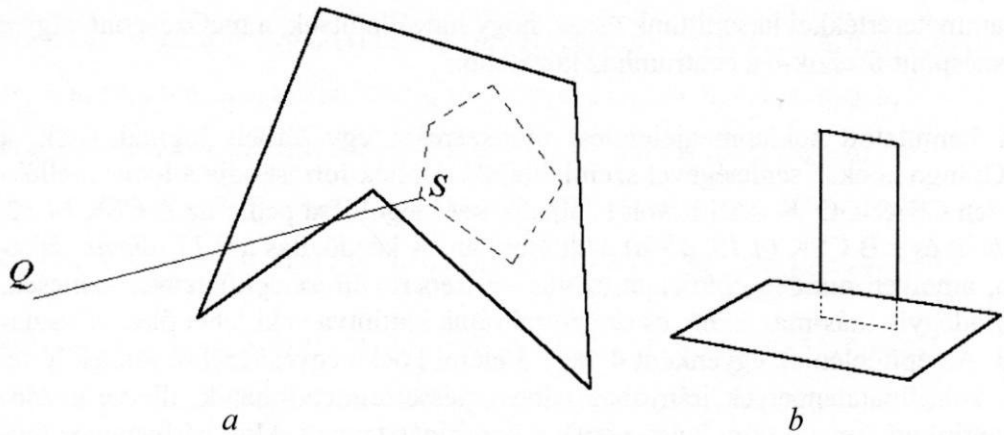


4.9. ábra

Előfordulhat, hogy ez a módszer nem alkalmazható, éspedig akkor, ha az egyik sokszög vetülete teljesen a másikéba esik *(4.10.a. ábra),* vagy ha a vetületek élei közötti metszéspontok mind valódi metszéspontok *(4.10. b. ábra).*

Ezekben az esetekben olyan csúcspontot keresünk, amelynek vetülete a másik sokszög vetületének belsejébe esik. Tekintsünk egy képsíkbeli *Q* pontot, amelyik biztosan a vizsgált sokszögön kívül fekszik *(4.10.a. ábra).* A másik sokszög egy csúcsának vetülete legyen *S.* Akkor megnézzük, hogy a *QS* szakasz hány élét metszi az első sokszög vetületének. Ha a metszéspontok száma páratlan, akkor *S* a vetületsokszögön belül, ha páros, akkor ezen kívül fekszik. Ezt a műveletet a második sokszög minden csúcsára elvégezzük, majd a sokszögeket felcserélve megismételjük addig míg egy belső pontot találunk.

Ezután a csúcshoz tartozó vetítősugarat metszük a másik sokszög síkjával és összehasonlítjuk a metszéspont és a csúcs magasságát (merőleges vetítés esetén) illetve középponttól való távolságát (centrális vetítésnél) a megjelenítési sorrend megállapítása érdekében.



4.10. ábra

Legyen az első sokszög három nem kollineáris csúcsa pozitív sorrendben $a_1(\xi_1, \eta_1, \zeta_1)$, $a_2(\xi_2, \eta_2, \zeta_2)$ és $a_3(\xi_3, \eta_3, \zeta_3)$. Akkor a sokszög síkjának egyenlete:

$$\begin{vmatrix} \xi & \eta & \zeta & 1 \\ \xi_1 & \eta_1 & \zeta_1 & 1 \\ \xi_2 & \eta_2 & \zeta_2 & 1 \\ \xi_3 & \eta_3 & \zeta_3 & 1 \end{vmatrix} = 0 \quad , \tag{4.36}$$

azaz

$$a_h\xi + b_h\eta + c_h\zeta + d_h = 0 \ , \tag{4.37}$$

ahol az együtthatókat a 4.23 -hoz hasonló képletek adják. Ezt a síkot metszve a *KS* vetítősugárral (4.32 előállítás) kapjuk a metszéspontnak megfelelő

$$u = \frac{c_h d + d_h}{c_h d - a_h \xi_S - b_h \eta_S} \tag{4.38}$$

paraméterértéket, amelyet az *S* vetületű *s* csúcsot jellemző

$$v = 1 - \frac{\zeta_s}{d} \tag{4.39}$$

paraméterértékkel hasonlítunk össze, hogy megállapítsuk, a metszéspont vagy a csúcspont fekszik-e a centrumhoz közelebb.

A bemutatott soklapmegjelenítési módszereket egy térbeli logikai játék, a "Csángó kocka" segítségével szemléltetjük. A játék forráskódja a lemezmellékleten CSANGO_K.PAS néven található, két megoldása pedig az A.CSK *(4.12. ábra)* és a B.CSK *(4.13. ábra)* adattárokban. A kezdőállás a *4.11. ábrán* látható, amelyen minden építőelem szürke. A képernyőn az építőelemek színesek. mindegyik más-más színű, és az egérrel rájuk kattintva is ki lehet őket választani. Az építőelemek egyenként 4 vagy 5 elemi kocka egyesítéséből jönnek létre. A koordinátatengelyek irányában adott lépéssel elmozdíthatók, illetve kezdőpontjukon átmenő (egy belső pont), a koordinátatengelyekkel párhuzamos tengelyek körül 90°-kal elforgathatók. A merev építőelemek közötti áthatás kizárását ebben az esetben úgy is meg lehet oldani, hogy a teret az építőelemek elemi kockáinak megfelelő kockákra bontjuk és egy mátrixban tároljuk a kockák szabad (0) vagy foglalt (1) állapotát jelző számot. Csak azok a mozgatások megengedettek, amelyek során az építőelem új helye szabad kockákból áll.

```
($N+)
program csango_k;

uses crt, dos, graph, grafind, eger kez, file kez, kerdesek, kozos;

type
  ml = array[1..12,0..8] of byte; mp =
  array[1..81 of pont;

const
  a=13; kx=260; ky: array[0..2] of integer =
  (380,300,275); kz=1500; {a szemtávolság pixelben)
  x1=20; y1=50; x2=510; y2=460; y3=70;
  fel=#72; le=#80; bal=#75; jobb=#77;
  pr: array[1..5,1..4] of char =
        (('X','U','R','E'),('Y','V','S','F'),
        ('Z','W','T','G'),(fel,le,bal,jobb),('A','B','Q',' '));
        valasz• kars =
        ['X','U','R','E','Y','V','S','F','Z','W','T','G',
                                    fel,le,bal,jobb,'A','B','Q');
  fi0=pi/18-pi/6; dfi=pi/6; si0=7*pi/18; dsi=pi/6;
  x0=x2-40; y0=y1+40;
    {építőelemek csúcsainak száma)
  nc: array [ 1.. 6 ] of byte = (18,15,15,15,15,16);
    {építőelemek lapjainak oldalszáma és a megfelelő csúcsok sorszáma}
  ns: array[1..6] of ml =
                  (((4,1,4,8,5,0,0,0,0), (4,2,6,7,3,0,0,0,0),
```

```
(4,9,10,14,13,0,0,0,0), (4,11,12,16,15,0,0,0,0), (4,8,9,17,5,0,0,0,0),

(4,7,6,18,12,0,0,0,0), (4,13,16,18,17,0,0,0,0), (4,5,6,2,1,0,0,0,0)),

((4,1,4,8,5,0,0,0,0), (6,9,10,3,2,6,13,0,0), (4,11,10,9,15,0,0,0,0),

(4,1,2,3,4,0,0,0,0), (4,9,13,14,15,0,0,0,0), (6,4,3,10,11,12,8,0,0),

(4,15,14,12,11,0,0,0,0), (4,5,8,12,7,0,0,0,0), (4,6,7,14,13,0,0,0,0),
     (4,5,6,2,1,0,0,0,0), (0,0,0,0,0,0,0,0,0), (0,0,0,0,0,0,0,0,0)),

((6,1,4,15,11,12,5,0,0), (4,2,6,7,3,0,0,0,0), (4,11,15,10,14,0,0,0,0),

(4,1,2,3,4,0,0,0,0), (4,14,13,12,11,0,0,0,0), (6,4,3,7,9,10,15,0,0),

(4,13,14,10,9,0,0,0,0), (4,8,9,7,6,0,0,0,0), (4,12,13,8,5,0,0,0,0),
     (4,5,6,2,1,0,0,0,0), (0,0,0,0,0,0,0,0,0), (0,0,0,0,0,0,0,0,0)),

((4,1,4,8,5,0,0,0,0), (6,3,2,6,7,15,11,0,0), (4,10,11,15,14,0,0,0,0),

(4,1,2,3,4,0,0,0,0), (4,7,13,14,15,0,0,0,0), (6,4,3,11,10,9,8,0,0),

(4,9,10,14,13,0,0,0,0), (4,8,9,12,5,0,0,0,0), (4,12,13,7,6,0,0,0,0),
     (4,5,6,2,1,0,0,0,0), (0,0,0,0,0,0,0,0,0), (0,0,0,0,0,0,0,0,0)),

((6,1,4,10,14,13,5,0,0), (4,2,6,7,3,0,0,0,0), (4,10,11,15,14,0,0,0,0),

(4,1,2,3,4,0,0,0,0), (4,8,13,14,15,0,0,0,0), (6,4,3,7,12,11,10,0,0),

(4,11,12,8,15,0,0,0,0), (4,9,12,7,6,0,0,0,0), (4,8,9,5,13,0,0,0,0),
     (4,5,6,2,1,0,0,0,0), (0,0,0,0,0,0,0,0,0), (0,0,0,0,0,0,0,0,0)),

((4,1,4,8,5,0,0,0,0), (4,6,15,11,7,0,0,0,0), (4,10,11,15,14,0,0,0,0),

(4,1,2,3,4,0,0,0,0), (8,13,14,15,6,16,2,1,5), (8,4,3,12,7,11,10,9,8),

(4,9,10,14,13,0,0,0,0), (4,16,12,3,2,0,0,0,0), (4,8,9,13,5,0,0,0,0),
      (4,6,7,12,16,0,0,0,0), (0,0,0,0,0,0,0,0,0), (0,0,0,0,0,0,0,0,0)));

type
 pont3 = object
   x, y, z: real;
   procedure uj(a,b,c: real);
 end;

 hely3 = object
   u, v, w: shortint;
   procedure uj(a,b,c: shortint);
 end;

 mcs = array[1..18] of pont3;
```

```
  ms = array[1..18] of pont;
  mbp = array[1..5] of hely3;

  elem = object(pont3) {építőelem objektum értelmezése)
    ncs, nk, nl, nlh: byte; {csúcsok, elemi kockák, lapok illetve
                             látható lapok száma)
    cs: mcs; {csúcsok koordinátái az 1-es számú elemi kocka
              középpontjához viszonyítva)
    s, p: mcs; {csúcsok vetületeinek koordinátái középpontos (s) és
                párhuzamos (p) vetítésnél)
    bp: mbp; {elemi kockák középpontjainak az I-es számúéhoz
              viszonyított helyei (áthatáskizáráshoz)}
    lap: ml; {lapok csúcsainak száma, illetve a laphoz tartozó
              csúcsok sorszámai}
    lth: array[1..12] of byte; {lapok láthatósága}
    mg: array[0..12] of real; {elem középpontjának magassága és a
                               lapok középmagassága}
    1_sorr: array[0..8] of byte; {látható lapok megjelenítési
                                  sorrendje}
    procedure vet;
    procedure tol(n: byte);
    procedure forg(n: byte);
    procedure athatas;
    procedure sorrend;
  end;

  mk = array[1..14] of elem;

  bill = object {vizuális billentyű értelmezése)
    xl,yl,x2,y2: integer;
    nev: char;
    ny: byte;
    procedure mj;
    procedure uj (x, y, a, b, n: integer; nv: char) ;
    procedure nyom;
  end;

var
  kc: mk; ff:  file of mk;
  lsz, i, j: integer;
  vb: array[1..5,1..4] of bill;
  fi, si, sf, cf, sp, cp: real;
  nsi: byte;
  xt, yt, zt, ut, vt, wt: real;
  ks: byte; {a kiválasztott építőelem sorszáma)
  mf: array[0..9,0..9,0..6] of byte; (tértartományok elfoglalási
                                      mátrixa
                                      - áthatások kiküszöböléséhez}
  rgh, ujh: mbp; {az elfoglalt tartományok régi és új címei}
  at: byte; {áthatásteszt eredménye)
  e_sorr: array[0..14] of byte; {elemek megjelenítési sorrendje}
  l_sorr: array[0..100] of pont; {lapok megjelenítési sorrendje)
```

```
  nlh: byte; {látható lapok száma}
  dbx, dbn: array[0..100] of pont3; {lapok vetületeit bezáró "dobozok"
                                     sarkai}

procedure pont3.uj(a,b,c: real);
begin
  x:=a; y:=b; z:=c;
end;

procedure hely3.uj(a,b,c: shortint);
begin
  u:=a; v:=b; w:=c;
end;

procedure cim; (a játék neve és szerzője)
const
  cx=330; cy=25;
begin
  setcolor(0); setlinestyle(0,0,1);
  setfillstyle(1,10); fillellipse(cx, cy, 65, 20);
  setfillstyle(1,15); fillellipse(cx, cy, 57, 15);
  settextstyle(0,0,1); setcolor(12); outtextxy(cx-50,cy-4,
                                      'Emoke játékai');
  outtextxy (cx-34, cy-6, "") ; setcolor(5); sett'xtstyle (1,
  0, 4) ; outtextxy(10,10,'CSANGO KOCKA');
  settextstyle(0,0,1); outtextxy(58,10,'/       /');
  setcolor(5); outtextxy(5,468,'C Füzi János'); circle(8,471,7);
end;

procedure nyil(x0,y0: integer; alf: real);
const sh=7; sr=3; h=12;
var a,b,c,d,e,f: integer;
begin
  a:=round(h*cos(alf)); b:=round(h*sin(alf));
  c:=round(-sh*cos(alf)+sr*sin(alf));
  d:=round(sh*sin(alf)+sr*cos(alf)); e:=round(-
  sh*cos(alf)-sr*sin(alf)); f:=round(sh*sin(alf)-
  sr*cos(alf)); moveto(x0-a,y0+b); lineto(x0+a, y0-
  b); linerel(c,d); moveto(x0+a,y0-b); linerel(e,f);
end;

procedure iv (x0, y0, n: integer) ; {forgatási irányt jelző ív)
var a,b: integer;
begin
  setcolor(12); .
  ellipse(x0,y0,30,330,8,16);
  case n of
  0:begin a:=x0+7; b:=y0+8;
      moveto(a,b); linerel(0,5); moveto(a,b); linerel(-4,3); end;
  1:begin a:=x0+7; b:=y0-8;
      moveto(a,b); linerel(0,-5); moveto(a,b); linerel(-4,-3); end;
  end;
```

```
  setcolor(0);
end;

procedure mezo; (képsíktartomány törlése}
begin
  setfillstyle(1,15); bar(x1+4,y1+4,x2-4,y2-4);
  setcolor(3); line(xl+6,y2-4,x2-6,y2-4); line(x2-4,y1+6,x2-4,y2-6);
end;

procedure bill.mj; {vizuális billentyű térhatású keretének
                    megjelenítése}
begin
  case ny of
  0 : kkeret (xl, yl, x2, y2, 1, 1) ; (felengedve}
  1:bkeret(xl-1,yl-1,x2+1,y2+1,1,1); (lenyomva}
  end;
end;

procedure bill.uj(x,y,a,b,n: integer; nv: char); (vizuális billentyű
                                                  felirata}
begin
  xl : =x; yl : =y; x2 : =x+a; y2 : =y+b; nev:=nv; mj;
  setcolor(3); settextstyle(1,0,2);
  if n>0 then outtextxy(x+(a div 3)+1,y+(b div 4),nev);
  setcolor(0);
  case n of
  2:nyil(x1+10,y+20,pí/2);
  3:nyil(x1+10,y+20,-pi/2);
  4:begin nyil(x1+22,y+20,0); iv(x1+22,y+20,0); end;
  5:begin nyil(x1+22,y+20,0); iv(x1+22,y+20,1); end;
  end;
  ny:=0;
end;

procedure bill.nyom; (billentyűlenyomás eredménye)
begin
  katt(xl,yl,x2,y2,ny); if ny=1 then qr:=nev;
end;

procedure nlap(x,y: integer; nv: char);
begin
  kkeret (x, y, x+110, y+120, 2, 2) ;
  setcolor(0); settextstyle(0,0,1); outtextxy(x+50,y+10,nv);
  vb[lsz,l].uj(x+5,y+25,45,40,2,pr[lsz,1J);
  vb[lsz,2].uj(x+60,y+25,45,40,3,pr[lsz,2]);
  vb[tsz,3J.uj(x+5,y+75,45,40,4,pr[lsz,3J);
  vb[lsz,4].uj(x+60,y+75,45,40,5,pr[lsz,4));
end;

procedure szinlap; (kiválasztott építőelem színének kijelzése}
begin
  setfillstyle(l,ks); bar(x2+45,y3-28,x2+87,y3-2);
end;
```

```
procedure keret;
begin
  setfillstyle(1,11); bar(0,0,getmaxx,getmaxy);
  bkeret (xl, yl, x2, y2, 4, 4); mezo;
  for lsz:=1 to 3 do nlap(x2+10,y3+lsz*130-110,chr(119+1sz));
  kkeret(x2+10,10,x2+120,80,2,2); vb[5,3].uj(450,10,60,35,1,'Q');
  setcolor(0); settextstyle(0,0,1);
  outtextxy(x2+25,20,'Kiválasztás'); outtextxy(462,14,'kilép');
  ablak(x2+42,y3-30,x2+90,y3); vb[5,1].uj(x2+16,y3-30,20,30,1,'A');
  vb[5,2].uj(x2+95,y3-30,20,30,1,'B');
  setcolor(0);
  nyil(x2+20,y3-15,-pi/2); nyil(x2+99,y3-15,pi/2);
  vb[4,4].uj(270,464,40,14,0,jobb); vb[4,3].uj(220,464,40,14,0,ba1);
  vb[4,1].uj(2,200,14,40,0,fel); vb[4,2].uj(2,250,14,40,0,1e);
  setcolor(12); setlinestyle(0,0,3);
  nyil(290,471,0); nyil(240,471,pi); nyil(9,270,-pi/2);
  nyil(9,220,pi/2);
  színlap;
  vb[5,4] .uj (0,0,0,0,0,' `);
end;

procedure gb ny; (vizuális billentyű lenyomása)
var kn: byte;
begin
     {egérrel}
  with reg do begin
     ax:=3; intr($33,reg);
     if bx=1 then begin
      for k:=1 to 5 do
        for 1:=1 to 4 do vb[k,l].nyom;
         {építőelem kiválasztása színe szerint, rákattintással)
       if (cx>xl) and (cx<x2) and (dx>yl) and (dx<y2) then begin
        kn:=getpixel(cx,dx);
        if (kn>0) and (kn<15) then begin ks:=kn; szinlap; end;
       end;
     end;
  end;
     (billentyűzetről}
  if keypressed then qr:=upcase (readkey);
end;

procedure lenyom(k,1: byte); {billentyűlenyomás megjelenítése}
begin
  vb[k,l).ny:=1; vb[k,l].mj;
end;

procedure feleng(k,l: byte);
begin
  vb[k,1].ny:=0; vb[k,1].mj; (billentyűfelengedés megjelenítése)
end;
```

```
procedure szogf; {vetítési szögek szögfüggvényei}
begin
  sf:=sin(fi); cf:=cos(fi); sp:=sin(si); cp:=cos(si);
end;

procedure vetit(x,y,z: real);
var t: real;
begin
  ut:=-x*sf+y*cf;              {párhuzamos vetítés}
  vt:=-x*cp*cf-y*cp*sf+z*sp;
  wt:=x*sp*cf+y*sp*sf+z*cp; {képsíkhoz viszonyított magasság)
  xt:=kz*ut/(kz-wt);           (perspektíva)
  yt:=kz*vt/ (kz-wt) ;
  t:=sqrt(xt*xt+yt*yt+kz*kz); {pont távolsága a képsíkbeli
                               vetületétől}
  zt:=t*wt/kz;
end;

procedure kdr; (koordinátarendszer megjelenítése)
begin
  setcolor(0);
  vetit(30,0,0);moveto(x0,y0); lineto(x0+round(xt),y0-round(yt));
  outtext('x');
  vetit(0,30,0);moveto(x0,y0); lineto(x0+round(xt),y0-round(yt));
  outtext('y');
  vetit (0, 0, 30) ;moveto (x0, y0) ; lineto (x0+round (xt) , y0-
  round (yt)) ;
  outtext('z');
end;          .

procedure lap megj(k,l: byte); (k.-ik elem 1.-ik lapjának
                                megjelenítése)
var 1p: mp; q: byte;
begin
  setcolor(16); setfillstyle(l,k);
  with kc[k] do begin
    for q:=1 to lap[1,0] do
      1p[q].uj(kx+round(s[lap[1,q]].x),ky[nsi]-round(s[lap[l,q)].y));
    fillpoly(lap[1,0],lp);
  end;
end;

procedure osszeh(kl,ll,k2,12:byte; var sk: byte);
  {k1.-ik kocka 11.-ik lapja és k2.-ik kocka 12.-ik lapja közötti
  átfedés vizsgálata a megjelenítési sorrend megállapításához} const
  bz=le-3; (biztonsági távolság, a valódi metszéspontokban
                    előforduló összehasonlítási hiba elkerülésére)
        bt=le-2;
  var
    i,j,mz,ct: byte;
    a, b, c, d, aa, bb, cc, dd, ee, m: pont3;
```

```
  procedure metsz;
    (két szakasz képsíkra eső vetületének metszéspontja)
  begin
    del:=(bb.x-aa.x)*(cc.y-dd.y)-(bb.y-aa.y)*(cc.x-dd.x);
    if del<>0 then begin
      u:=((cc.x-aa.x)*(cc.y-dd.y)-(cc.y-aa.y)*(cc.x-dd.x))/del;
      v:=((bb.x-aa.x)*(cc.y-aa.y)-(bb.y-aa.y)*(cc.x-aa.x))/del;
         if (u>bz) and (u<1-bz) and (v>bz) and (v<1-bz) then mz:=1;
    end;
  end;

  procedure egyutth; {lap síkja egyenletének együtthatói}
  begin
    ah:=a.y*(b.z-c.z)+b.y*(c.z-a.z)+c.y*(a.z-b.z);
    bh:=a.z*(b.x-c.x)+b.z*(c.x-a.x)+c.z*(a.x-b.x);
    ch:=a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y);
    dh:=a.x*(c.y*b.z-b.y*c.z)+a.y*(c.z*b.x-b.z*c.x)+a.z*
        (c.x*b.y-b.x*c.y);
  end;
begin
  sk:=0;
  for i:=1 to kc[kl].lap[11,0] do
    for j:=1 to kc[k2].lap[12,0] do
      if sk<>1 then begin
        with kc[k1] do begin
          aa:=s[lap[ll,i]]; a:=p[lap[11,i]];
          if i<lap[11,0] then begin bb:=s[lap[11,i+1]];
                                    b:=p[lap[ll,i+1]];
          end else begin bb:=s[lap[11,1]]; b:=p[lap[11,1]]; end;
        end;
        with kc[k2] do begin
          cc:=s[lap[12,j]]; c:=p[lap[12,j]];
          if j<lap[12,0] then begin dd:=s[lap[12,j+1]];
                                    d:=p[lap[12,j+1]];
          end else begin dd:=s[lap[12,1]]; d:=p[lap[12,1]]; end;
        end;
        mz:=0; u:=0; v:=1; metsz;
        if mz=1 then begin
          ah:=(1-u)*aa.x+u*bb.x;
          bh:=(1-v)*cc.x+v*dd.x;
          n1:=ah*(b.z-a.z)+kz*(b.x-a.x);
          n2:=bh*(d.z-c.z)+kz*(d.x-c.x);
          if (abs(nl)<bz) or (abs(n2)<bz) then sk:=2 else begin
            u:=(kz*(b.x-a.x)-a.z*b.x+a.x*b.z)/ni;
            v:=(kz*(d.x-c.x)-c.z*d.x+c.x*d.z)/n2;
                  if abs(u-v)<bt then sk:=2 else if v<u then sk:=1;
          end;
        end;
      end;
  cc . uj (500, 500, 0) ;
  if sk=2 then begin
    for i:=1 to kc[ki].lap[11,0] do
      if sk=2 then begin
```

```
        ct:=0;
        dd:=kc[kl].s[kc[ki].lap[11,i]];
        ee:=kc[kl].p[kc[ki].lap[11,i]];
        for j:=1 to kc[k2].lap[12,0] do begin
         with kc[k2] do begin
           aa:=s[lap(12,j]];
           if j<lap[12,0] then bb:=s[lap[12,j+1]]
                          else bb:=s[lap[12,1]];
         end;
         mz:=0; metsz; if mz=1 then ct:=ct+1;
        end;
        if odd(ct) then begin {átfedésteszt}
         with kc[k2] do begin
           a:=p[lap[12,1]]; b:=p[lap[12,2]]; c:=p[lap[12,3]]; end;
         egyutth;
         n1:=ah*dd.x+bh*dd.y-ch*kz;
         if abs(ni)>bz then begin
           u:=-(ch*kz+dh)/nl;
           v:=1-ee.z/kz;
           if u<v-bz then sk:=1;
         end;
        end;
      end;
  end;
  if sk=2 then begin
     for i:=1 to kc[k2].lap[12,0] do
      if sk=2 then begin
        ct:=0;
        dd:=kc[k2].s[kc[k2].lap[12,i]];
        ee:=kc[k2].p[kc[k2].lap[12,i)];
        for j:=1 to kc[kl].lap[11,0] do begin
         with kc[kl] do begin
           aa:=s[lap[ll,j]];
           if j<lap[11,0] then bb:=s[lap[11,j+1]]
                          else bb:=s[lap[11,1]];
         end;
         mz:=0; metsz; if mz=1 then ct:=ct+1;
        end;
        if odd(ct) then begin {átfedésteszt}
         with kc[kl] do begin
           a:=p[lap[11,1]]; b:=p[lap[11,2]]; c:=p[lap[11,3]]; end;
         egyutth;
         nl:=ah*dd.x+bh*dd.y-ch*kz;
         if abs(ni)>bz then begin
           u:=-(ch*kz+dh)/n1;
           v:=1-ee.z/kz;
           if v<u-bz then sk:=1;
         end;
        end;
      end;
  end;
end;
```

```
procedure elem.sorrend; {lapok kezdeti sorrendje egy elemen belül}
begin
  nlh:=0;
  for j:=1 to nl do if lth[j]=1 then begin nlh:=nlh+1;
                                           l sorr[nlh]:=j end;
end;

procedure doboz(k,l: byte); (k.-ik elem 1.-ík lapját bezáró "doboz"}
var
  ux, un, vx, vn, u, v: real;
  q: byte;
begin
  with kc[k] do begin
     ux:=s[lap[l,l]].x; un:=ux;
     vx:=s[lap[1,1]].y; vn:=vx;
     for q:=2 to lap[1,0] do begin
      u:=s[lap[l,q] ] .x;
      if u<un then un:=u else if u>ux then ux:=u;
      v:=s[lap[l,q]].y;
      if v<vn then vn:=v else if v>vx then vx:=v;
     end;
  end;
  dbn[i] .uj (un,vn, 0) ; dbx[i] .uj (ux,vx, 0) ;
end;

procedure 1_sorrend; {lapok megjelenítési sorrendjének megállapítása}
var sik: byte;
  procedure beszur(i,j,p: byte);
  var q: integer;
  begin
     l sorr[0]:=l sorr[i];
     for q:=i-1 downto j do 1_sorr[q+l]:=1_sorr[q];
     1_sorr[j]:=1_sorr[0];
     if p=1 then begin
      dbx[0]:=dbx[i];
      for q:=i-1 downto j do dbx[q+l]:=dbx[q];
      dbx[j]:=dbx[0];
      dbn[0]:=dbn[i];
      for q:=i-1 downto j do dbn[q+l]:=dbn[q];
      dbn[j]:=dbn[0];
     end;
  end;
begin
  nlh:=0;
     {kezdősorrend}
  for i:=1 to 14 do begin
     for j:=1 to kc[i].nlh do
      1_sorr[nlh+j].uj(i,kc[i].l_sorr[j]);
     nlh:=nlh+kc[i].nlh;
  end;
     (sorrendjavítás a lapok középmagassága szerint)
  for 1:=2 to nlh do begin
     j:=0;
```

```
      repeat
        j:=j+1; sik:=0;
        if kc[1_sorr[1].xp].mg[1_sorr[1].yp]<
          kc[1_sorr[j].xp].mg[1_sorr[j].yp] then sik:=1;
      until (sik=1) or (j=1-1);
      if sik=1 then beszur(1,j,0);
    end;
      {"dobozok" sarkainak kiszámítása}
    for i:=1 to nlh do doboz(1_sorr[i].xp,l_sorr[i].yp);
      {sorrendjavítás az átfedések figyelembe vételével)
    for 1:=2 to nlh do begin
      j:=0;
      repeat
        j:=j+1; sik:=0;
          ("dobozok" átfedésének vizsgálata}
        if (dbx[j].x>dbn[1].x) and (dbn[j].x<dbx[1].x) and
            (dbx[j].y>dbn[1].y) and (dbn[j].y<dbx[1].y) then
          {átfedésvizsgálat}
          osszeh(1_sorr[1].xp,1 sorr[1].yp,l_sorr[j].xp,
                 1_sorr[j].yp,sik);
      until (sik=1) or (j=1-1);
      if sik=1 then beszur(l,j,l);
    end;
end;

procedure rajz; {megjelenítés}
begin
    for i:=1 to 14 do kc[i].sorrend; 1_sorrend; mezo;
    setcolor(0); ("asztal" megjelenítése)
    for k:=-5 to 5 do begin
      vetit(2*k*a,-10*a,0); moveto(kx+round(xt),ky[nsi]-round(yt));
      vetit(2*k*a,10*a,0); lineto(kx+round(xt),ky[nsi]-round(yt));
      vetit(-10*a,2*k*a,0); moveto(kx+round(xt),ky[nsi]-round(yt));
      vetit(10*a,2*k*a,0); lineto(kx+round(xt),ky[nsi]-round(yt));
    end;
    for i:=1 to nlh do lap_megj(1_sorr[i].xp,l_sorr[i].yp);
    kdr;
end;

procedure forgat; {képsík (vetítési irány) forgatása)
begin
    case qr of
    fel :if nsi>0 then nsi:=nsi-1;
    le :if nsi<2 then nsi:=nsi+1;
    bal :fi:=fi+dfi;
    jobb:fi:=fi-dfi;
    end;
    si:=si0-nsi*dsi; szogf;
    for i:=1 to 14 do kc[i].vet;
    rajz;
    for 1:=1 to 4 do if qr=pr[4,1] then feleng(4,1);
end;
```

```
procedure valaszt; {építőelem kiválasztása görgetéssel)
begin
   case qr of
   'A':begin
         if ks<14 then ks:=ks+1 else ks:=1;
         delay(500); feleng(5,1);
       end;
   'B':begin
         if ks>1 then ks:=ks-1 else ks:=14;
         delay(500); feleng(5,2);
       end;
   end;
   szinlap;
end;

procedure kilep; {kilépés)
begin

   kerdez(200,200,'  Menteni a konfigurációt ?','gen','em','I','N');
   repeat kattint (200, 200, ' I' , 'N' , qm) ;
   until (qm='I') or (qm='N');
   if qm='I' then begin
     eger_n; ment('CSK'); {konfiguráció tárolása)
     if v='I' then begin
      assign(ff,kn); rewrite(ff); write(ff,kc); close(ff); end;
     eger_1;
   end;
end;

procedure elem.vet; {építőelem csúcsainak képsíkra vetítése}
var
   i, j: byte;
   xv, yv, zv, v: real;
begin
   for j:=1 to ncs do begin
     xv:=x+cs[j] .x; yv:=y+cs[j] .y; zv:=z+cs[j] .z;
     vetit(xv,yv,zv); p[j].uj(ut,vt,wt); s[j].uj(xt,yt,zt);
   end;
   vetit (x, y, z) ; mg [0] :=zt;
     (lapok láthatóságának vizsgálata)
   for j:=1 to nl do
     if s[lap[j,l}].x*(s[lap[j,2]].y-s[lap[j,3]].y)+
        s[lap[j,2]).x*(s[lap[j,3]].y-s[lap[j,l]l.y)+
        s[lap[j,3]].x*(s[lap[j,l]].y-s[lap[j,2]].y)>0
     then lth[j]:=1 else lth[j]:=0;
     {lapok középmagassága}
   for j:=1 to nl do begin
     mg[j]:=0;
     for i:=1 to lap[j,0] do
      mg[j] :=mg[j]+s[lap[j,i] ] .z;
     mg[j] :=mg[j]/lap[j,0];
   end;
end;
```

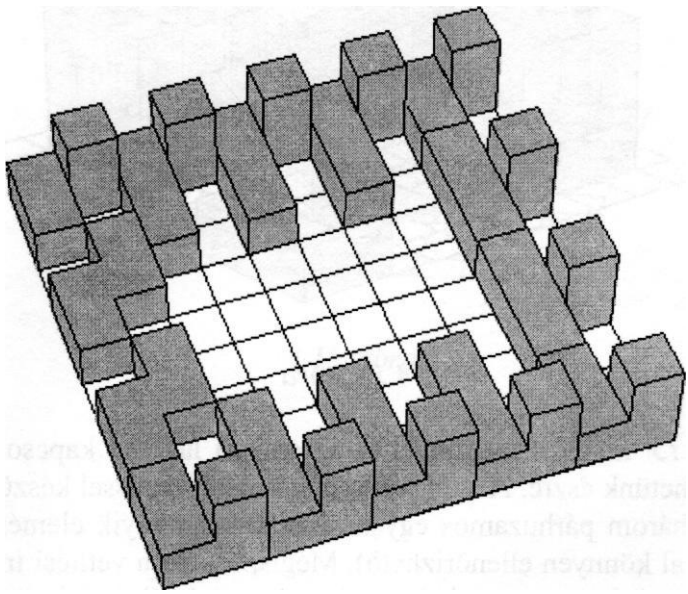
```
procedure elem.athatas; {elemek közötti áthatás vizsgálata}
begin
   j:=0;
   repeat
     j :=j+1;
     with ujh[j] do
      if (u<0) or (u>9) or (v<0) or (v>9) or (w<0) or (w>6) then
         at:=1 else if mf[u,v,w]=1 then at:=1;
   until (j=nk) or (at=1) ;
end;

procedure elem.tol(n: byte);
begin
   rgh[1]:=bp[1]; {régi hely kiszámítása}
   for i:=2 to nk do
     rgh[i].uj(bp[i].u+bp[1].u,bp[i].v+bp[1].v,bp[i].w+bp[1].w);
   with bp[1] do {új hely kiszámítása}
     case n of
     1:ujh[1] .uj (u+1,v,w) ;
     2:ujh[1].uj(u-1,v,w);
     3:ujh[1].uj(u,v+1,w);
     4:ujh[1].uj(u,v-1,w);
     5:ujh[1] .uj (u,v,w+1);
     6:ujh[1].uj(u,v,w-1);
     end;
   with ujh[1] do
   for i:=2 to nk do
     ujh[i].uj(bp[i].0+u,bp[i].v+v,bp[i].w+w);
   at : =0;
   for i:=1 to nk do
     mf[rgh[i].u,rgh[i].v,rgh[i].w]:=0; (régi hely felszabadítása}
   athatas;
   if at=0 then begin
     for i:=1 to nk do
      mf[ujh[i].u,ujh[i].v,ujh[i].w]:=1; (új hely elfoglalása}
     bp[1] :=ujh[1];
     case n of (kezdőpont eltolása)
     1:x:=x+2*a;
     2:x:=x-2*a;
     3:y:=y+2*a;
     4:y:=y-2*a;
     5:z:=z+2*a;
     6:z:=z-2*a;
     end;
   end else for i:=1 to nk do
     mf[rgh[i].u,rgh[i].v,rgh[i].w]:=1; {régi hely visszafoglalása}
end;

procedure elem.forg(n: byte);
begin
   rgh[1]:=bp[1]; {régi hely kiszámítása}
   for i:=2 to nk do
     rgh[i].uj(bp[i].u+bp[1].u,bp[i].v+bp[1].v,bp[i].w+bp[1].w);
```

```
  ujh[1]:=rgh[1]; {új hely kiszámítása}
  for i:=2 to nk do with bp[i] do
   case n of
   1:ujh[i] .uj (u,-w,v);
   2:ujh[i] .uj (u,w,-v);
   3:ujh[i] .uj (w,v,-u);
   4:ujh[i] .uj (-w,v,u);
   5:ujh[i] .uj (-v,u,w);
   6:ujh[i] .uj (v,-u,w);
   end;
  with ujh[1] do
  for i:=2 to nk do
   ujh[i] .uj (ujh[i] .0+u,ujh[i] .v+v,ujh[i] .w+w);
  at:=0;
  for i:=1 to nk do
   mf[rgh[i].u,rgh[i].v,rgh[i].w]:=0; {régi hely felszabadítása}
  athatas;
  if at=0 then begin
   for i:=1 to nk do
     mf [ujh[i] .u,ujh[i] .v,ujh[i] .w] :=1; {új hely elfoglalása}
   with ujh[1] do {elemi kockák új viszonylagos helyeinek
                   kiszámítása}
         for i:=2 to nk do bp[i].uj(ujh[i].u-u,ujh[i].v-v,ujh[i].w-w);
   for j:=1 to ncs do with cs[j] do (elem forgatása)
     case n of
     1 : uj (x, -z, y) ;
     2 : uj (x, z, -y) ;
     3:uj (z,y,-x);
     4:uj (-z,Y,x);
     5:uj (-y,x,z);
     6 : uj (y, -x, z) ;
     end;
  end else for i:=1 to nk do
   mf[rgh[i].u,rgh[i].v,rgh[i].w]:=1; (régi hely visszafoglalása)
end;

procedure mozgat;
begin
  case qr of
  'X':begin kc[ks].tol(1);  kc[ks].vet;   rajz;   feleng(1,1);   end;
  'U':begin  kc[ks].tol(2);  kc[ks].vet;  rajz;  feleng(1,2);  end;
  'R':begin  kc[ks].forg(1);  kc[ks].vet;  rajz;  feleng(1,3);  end;
  'E':begin kc[ks].forg(2); kc[ks].vet; rajz; feleng(1,4); end;
  'Y':begin kc[ks].tol(3);  kc[ks].vet; rajz; feleng(2,1); end;
  'V':begin kc[ks].tol(4);  kc[ks].vet; rajz; feleng(2,2); end;
  'S':begin  kc[ks].forg(3);  kc[ks].vet;  rajz;  feleng(2,3);  end;
  'F':begin kc[ks].forg(4); kc[ks].vet; rajz; feleng(2,4); end;
  'Z':begin kc[ks].tol(5);  kc[ks].vet; rajz; feleng(3,1); end;
  'W':begin kc[ks].tol(6);  kc[ks].vet; rajz; feleng(3,2); end;
  'T':begin  kc[ks].forg(5);  kc[ks].vet;  rajz;  feleng(3,3);  end;
  'G':begin kc[ks].forg(6); kc[ks].vet; rajz; feleng(3,4); end; end;
end;
```

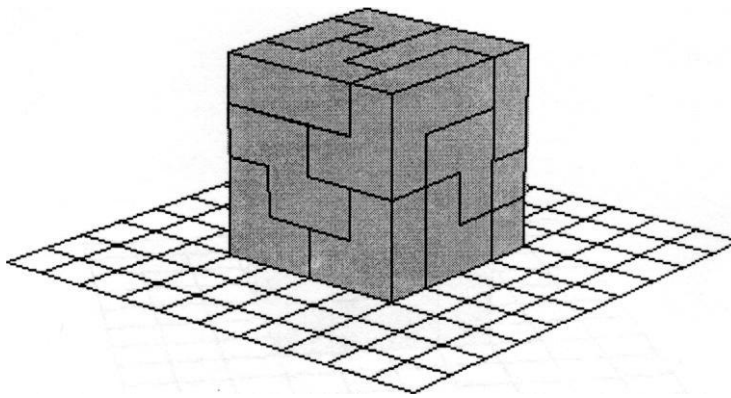
```
procedure alias;
begin
  eger_n; eger_t(0,0,getmaxx,getmaxy); rajz; eger_1;
end;

procedure kocka; {építőelemek létrehozása}
begin
  with kc[1] do begin
    ncs:=18; nk:=5; n1:=12; lap:=ns[1];
    cs[1].uj(3*a,-a,a); cs[2].uj(-3*a,-a,a); cs[3].uj(-3*a,-a,-a);
    cs[4] .uj (3*a,-a,-a); cs[5] .uj (3*a,a,a); cs[6] .uj (-
    3*a,a,a); cs[7].uj(-3*a,a,-a); cs[8].uj(3*a,a,-a);
    cs[9].uj(a,a,-a); cs[10].uj(a,3*a,-a); cs[11].uj(-a,3*a,-a);
    cs[12].uj(-a,a,-a); cs[13].uj(a,a,3*a); cs[14].uj(a,3*a,3*a);
    cs[15].uj(-a,3*a,3*a); cs[16] .uj (-a,a,3*a); cs[17] .uj
    (a,a,a); cs[18] .uj (-a,a,a); bp[1] .uj (0,0,0); bp[2] .uj
    (1,0,0); bp[3] .uj (-1,0,0);
    bp[4].uj (0, 1, 0) ;
    bp[5].uj (0, 1, 1) ;
  end;
  with kc[2] do begin
    ncs:=15; nk:=5; n1:=10; lap:=ns[2];
    cs[1].uj(3*a,-a,a); cs[2].uj(-3*a,-a,a); cs[3].uj(-3*a,-a,-a);
    cs[4] .uj (3*a,-a,-a); cs[5] .uj (3*a,a,a); cs[6] .uj (-3*a,a,a);
    cs[7].uj(-a,a,a); cs[8].uj(3*a,a,-a); cs[9].uj(-3*a,3*a,3*a);
    cs[10].uj(-3*a,3*a,-a); cs[11].uj(-a,3*a,-a); cs[12].uj(-a,a,-a);
    cs[13].uj(-3*a,a,3*a); cs[14].uj(-a,a,3*a); cs[15].uj(-
    a,3*a,3*a); bp[1].uj(0,0,0); bp[2].uj(1,0,0); bp[3].uj(-1,0,0);
    bp[4].uj (-
  1,1,0); bp[5].uj(-
  1,1,1); end;
  with kc[3] do begin
    ncs:=15; nk:=5; n1:=10; lap:=ns[3];
    cs[1].uj(3*a,-a,a); cs[2].uj(-3*a,-a,a); cs[3].uj(-3*a,-a,-a);
    cs[4] .uj (3*a,-a,-a); cs[5] .uj (3*a,a,a); cs[6] .uj (-
    3*a,a,a);
    cs[7] .uj (-3*a,a,-a); cs[8] .uj (a,a,a); cs[9] .uj (a,a,-a);
    cs[10].uj(a,3*a,-a); cs[11].uj(3*a,3*a,3*a);
    cs[12].uj(3*a,a,3*a); cs[13].uj(a,a,3*a); cs[14].uj(a,3*a,3*a);
    cs[15].uj(3*a,3*a,-a); bp[1] .uj (0,0,0); bp[2] .uj (1,0,0);
    bp[3] .uj (-1,0,0); bp[4].uj(1,1,0);
    bp[5].uj (1, 1, 1) ;
  end;
  with kc [ 4 ] do begin
    ncs:=15; nk:=4; n1:=10; lap:=ns[4];
    cs[1] .uj (3*a,-a,a); cs[2] .uj (-a,-a,a); cs[3] .uj (-a,-a,-a);
    cs[4] .uj (3*a,-a,-a); cs[5] .uj (3*a,a,a); cs[6] .uj (-a,a,a);
    cs[7].uj(-a,a,3*a); cs[8].uj(3*a,a,-a); cs[9].uj(a,a,-a);
    cs[10].uj(a,3*a,-a); cs[11].uj(-a,3*a,-a); cs[12].uj(a,a,a);
    cs[13].uj(a,a,3*a); cs[14].uj(a,3*a,3*a); cs[15].uj(-a,3*a,3*a);
    bp[1] .uj (0,0,0); bp[2] .uj (1,0,0); bp[3] .uj (0,1,0);
    bp[4] .uj (0, 1, 1) ;
  end;
```

```
    cs[1] .uj (a,-a,a); cs[2] .uj (-3*a,-a,a); cs[3] .uj (-3*a,-a,-
    a); cs[4] .uj (a,-a,-a); cs[5] .uj (a,a,a); cs[6] .uj (-
    3*a,a,a); cs[7].uj(-3*a,a,-a); cs[8].uj(-a,a,3*a); cs[9].uj(-
    a,a,a); cs[10] .uj (a,3*a,-a); cs[11] .uj (-a,3*a,-a); cs[12]
    .uj (-a,a,-a); cs[13].uj(a,a,3*a); cs[14].uj(a,3*a,3*a);
    cs[15].uj(-a,3*a,3*a); bp[1] .uj (0,0,0); bp[2] .uj (-1,0,0);
    bp[3] .uj (0,1,0); bp[4].uj(0,1,1);
  end;
  with kc[6] do begin
    ncs:=16; nk:=4; n1:=10; lap:=ns[6];
    cs[1].uj(3*a,-a,a); cs[2].uj(-a,-a,a); cs[3].uj(-a,-a,-a); cs[4]
    .uj (3*a,-a,-a); cs[5] .uj (3*a,a,a); cs[6] .uj (-3*a,a,a);
    cs[7] .uj (-3*a,a,-a); cs[8] .uj (3*a,a,-a); cs[9] .uj (a,a,-a);
    cs[10].uj(a,3*a,-a); cs[11].uj(-3*a,3*a,-a); cs[12].uj(-a,a,-a);
    cs[13] .uj (a,a,a); cs[14] .uj (a,3*a,a); cs[15] .uj (-
    3*a,3*a,a); cs[16] .uj (-a,a,a);
    bp[1].uj(0,0,0); bp[2].uj(1,0,0); bp[3].uj(0,1,0);
    bp[4].uj(-1,1,0);
  end;
  kc[9]:=kc[1]; kc[14]:=kc[6]; kc[12]:=kc[4]; kc[13]:=kc[5];
  kc[10]:=kc[2]; kc[7]:=kc[2]; kc[11]:=kc[3]; kc[8]:=kc[3];
  end;

procedure uj; {új konfiguráció}
begin
  with kc[1J do begin uj(3*a,7*a,a);   bp[1].uj(6,8,0); end;
  with kc[2] do begin uj(-7*a,-a,a);   bp[1].uj(1,4,0); end;
  with kc[3] do begin uj(7*a,-a,a);    bp[1].uj(8,4,0); end;
  with kc[4]  do begin uj(-9*a,7*a,a); bp[1].uj(0,8,0); end;
  with kc[5]  do begin uj(9*a,-9*a,a); bp[1].uj(9,0,0); end;
  with kc[6] do begin uj (3*a,-9*a,a); bp[1] .uj (6,0,0) ; end;
  with kc[7] do begin uj(-7*a,-5*a,a); bp[1].uj(1,2,0); end;
  with kc[8] do begin uj(7*a,3*a,a);   bp[1].uj(8,6,0); end;
  with kc[9] do begin uj (-3*a,7*a,a); bp[1] .uj (3,8,0);
  end; with kc[10] do begin uj (-7*a,3*a,a); bp[1] .uj
  (1,6,0) ; end; with kc[11] do begin uj(7*a,-5*a,a);
  bp[1].uj(8,2,0); end; with kc[12] do begin uj(-9*a,-
  9*a,a); bp[1].uj(0,0,0); end;
  with kc[13] do begin uj(9*a,7*a,a);  bp[1].uj(9,8,0); end;
  with kc[14] do begin uj(-3*a,-9*a,a); bp[1].uj(3,0,0);
  end; end;

procedure kezd; {kezdőkonfiguráció}
begin
  for j:=1 to 14 do
    with kc[j] do begin
      rgh[1]:=bp[1]; (elfoglalt helyek kiszámítása}
      for i:=2 to nk do
          rgh[i].uj(bp[i].0+bp[1].u,by[i].v+bp[1].v,bp[i].w+bp[1].w);
      for i:=1 to nk do
      mf[rgh[i].u,rgh[i].v,rgh[i].w]:=1; {helyfoglalás}
    end;
  for i:=1 to 14 do kc[i].vet;
```

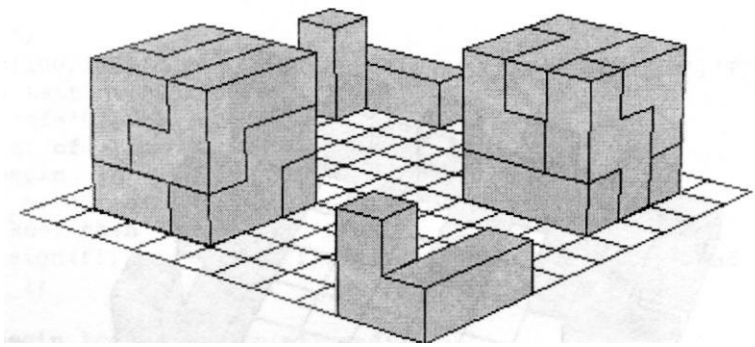
```
procedure konfig; (konfigurációkiválasztás)
begin
   for k:=0 to 9 do for 1:=0 to 9 do for i:=0 to 6 do mf[k,l,i):=0;
   settextstyle(0,0,1);
   repeat
     q f._'
     k e r d e z ( 2 0 0 , 2 0 0 , '      K o n f i g u r á c i ó
     repeat
     kattint(200,200,'U','T',qf); until
     (qf='U') or (qf='T'); case of of
     'T': begin
       eger_n; olvas('CSK'); (tárolt konfiguráció betöltése}
       if ko=1 then begin
          assign(ff,kn); reset(ff); read(ff,kc); close(ff); end;
       eger_l;
     end;
     'U': begin kocka; uj; ko:=1; end;
     end;
   until ko=1;
   kezd;
end;

begin
   fi:=fi0; si:=si0; nsi:=0; ks:=1;
   for k:=1 to 14 do e_sorr[k]:=k;
   grindhi; eger_k; eger_n; keret; cim; szogf; eger_l;
   repeat
     eger_n; qu:='N'; qk:='N'; konfig;
     for i:=1 to 14 do kc[i].vet; alias;
     repeat
       qr:=' ';
       repeat gb_ny until qr in valasz;
       eger_n;
       for k:=1 to 5 do
          for 1:=1 to 4 do
           if qr=vb[k,1).nev then lenyom(k,1);
       case qr of
       fel,le,bal,jobb: forgat;
       'A','B':valaszt;
       'Q':begin (kilépés}
            repeat kilep until not ((v='N') and (qm='I'));
            vege; if qk='N' then begin ujkonf; if qu='N' then
                       alias; end;
            feleng(5,3);
          end;
       else mozgat;
       end;
       eger1;
     until (qk='I') or (qu='I');
   until qu=' N' ;
   closegraph;
end.
```



-1.11. cíbra

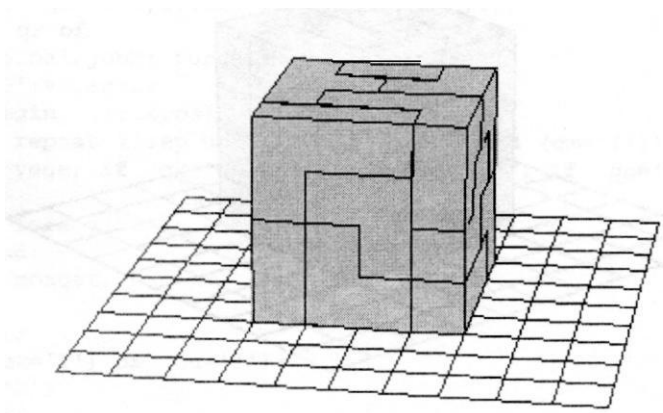


4.12. ábra

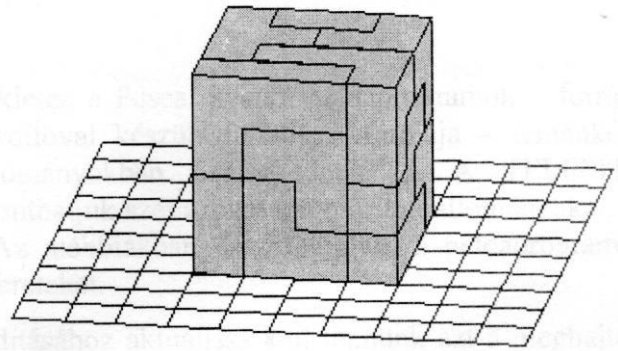


4.13. ábra

A *4.14.* és *4.15. ábrákat* megfigyelve, az emberi látással kapcsolatos érdekes jelenséget vehetünk észre. A *4.15. ábra* merőleges vetítéssel készült, rajta minden szakasz három párhuzamos egyenescsalád valamelyik elemén fekszik (ez egy vonalzóval könnyen ellenőrizhető). Mégis, főként a vetítési iránnyal legkisebb szöget bezáró egyenescsalád tagjait széttartóaknak látjuk. A középpontos vetítéssel készült *4.14. ábrán* a valóságban összefutó egyenesek párhuzamosnak tűnnek (a szerkesztés szemtávolságának megfelelő távolságból szemlélve az ábrát).



4.14. ábra



4.15. ábra

5. A lemezmelléklet használata

A lemezmellékleten a Pascal nyelvű példaprogramok — forráskódja és Turbo Pascal 7.0 fordítóval készült futtatható formája — témánként csoportosítva tömörített állományokban helyezkednek el. A TELEPIT.EXE program segítségével bonthatjuk szét azokat a tömörített állományokat, amelyekre szükségünk van. Az alábbiakban összefoglaljuk a példaprogramok telepítéséhez szükséges ismereteket.

A telepítés indításához aktuálissá kell tennünk azt a meghajtót, amelyik a lemezmellékletet tartalmazza, például az *A:*

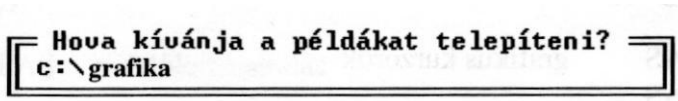
```
A:
```

Ezután következhet a telepítő program indítása:

```
A:\>telepit
```

A telepítő program megjeleníti a lemezen található tömörített állományok listáját. A kifejteni kívánt állomány(ok) nevére rá kell állni a nyílbillentyűk (↑ és ↓) segítségével. A kijelölést, illetve a kijelölés megszüntetését a szóköz billentyű lenyomásával végezhetjük el (a kiválasztott állományok neve mellett a √ karakter látható). A válogatás során a jobb oldali ablakban rövid összefoglaló jelenik meg a kurzor melletti állomány tartalmáról.

Ha az állományok kijelölése után megnyomjuk az <*Enter*> billentyűt, akkor a az alábbi inputsor jelenik meg:



Az inputsorban meg kell adnunk annak a könyvtárnak a nevét, ahova a kijelölt fájlokat másolni szeretnénk.

Az alkönyvtár kiválasztása után *(<Enter>)* elkezdődik a példaprogramok kifejtése és másolása. A példaprogramok a tömörített fájl nevével azonos nevű alkönyvtárakba kerülnek.

Megjegyezzük, ha a telepítés előtt a lemezmelléklet tartalmát a merevlemez valamely alkönyvtárába másoljuk, majd onnan indítjuk a TELEPIT.EXE programot, akkor a telepítés sokkal gyorsabban végbemegy.

A lemezmellékleten a programok a következő alkönyvtárakban találhatók:

1. KOZOS_U

Ebben az alkönyvtárban olyan programegységek (unit) találhatók, amelyek több program futtatásához szükségesek. Célszerű az alkönyvtárat elérő útvonalat *(path)* beírni a fejlesztési környezet (IDE) *Options / Directories* menü *Object directories* és *Unit directories* ablakaiba, hogy a fordító bárhonnan megtalálja.

EGER KEZ.PAS	egér programozása grafikus módban
EGER_KP.PAS	kontrollpontok mozgatása egér segítségével
FILE KEZ.PAS	kontrollpont-konfigurációt tartalmazó adattárak kezelése
GOMB_KEZ.PAS	vizuális billentyűk értelmezése
GRAFIND.PAS	grafikus üzemmód elindítása
KERDESEK.PAS	kommunikációs keret
KOMM_FEL.PAS	konfigurációkiválasztás-tárolás felületszerkesztéshez
KONF2D.PAS	kontrollpont-konfiguráció meghatározása síkgörbeszerkesztéshez
KONF3D.PAS	kontrollpont-konfiguráció meghatározása térgörbe- és felületszerkesztéshez
KOZOS.PAS	általában használt eljárások
KOZOS_3D.PAS	térgörbe- és felületszerkesztésnél használt eljárások
KOZOSGB.PAS	görbeszerkesztésnél használt eljárások
RAJZ F3D.PAS	felületek drótvázas vagy képies megjelenítése

2. ANIMACIO

GR_KZ_O.PAS GR_KZ_X. PAS EGKZ_K.PAS	grafikus kurzorok
BILIARD.PAS	többszörös ütközés szimuláció
CSANGO_K.PAS	térbeli összerakó játék
A.CSK, B.CSK	megoldásokat tartalmazó adattárok

GYONGY_K.PAS logikai játék

3. GB_2D

GB_2D_CH.PAS	síkgörbeszerkesztés általánosított (helyi ellenőrzésű) Coons-Hermite interpolációval
A.EC2, A.IC2	érintőket illetve karakterisztikus keretet tartalmazó adattár
GB_2D_BT.PAS	síkgörbeszerkesztés $_0$-spline közelítéssel
GB_2DIP.PAS	síkgörbeszerkesztés különböző interpolációs (Coons-Hermite, spline) és approximációs (B-spline, Bézier) eljárásokkal
A.IP2, B.IP2, C.IP2, D.IP2, X.IP2, Y.IP2, Z.IP2	karakterisztikus kereteket tartalmazó adattárak
GB_P_2D.PAS	síkgörbeszerkesztés Fourier sorbafejtéssel
A.GBP	együttható-adattár

4. GB_3D

GB_3D_BT.PAS	térgörbeszerkesztés $_\beta$-spline közelítéssel
GB_3D_IP.PAS	térgörbeszerkesztés különböző interpolációs (Coons-Hermite, spline) és approximációs (B-spline, Bézier) eljárásokkal
A.IP3, B.IP3	karakterisztikus kereteket tartalmazó adattárak

S. FT NYIL T

CSAV_GY.PAS	felületgenerálás
F_EM_SPL.PAS	Euler-Monge előállítású spline felület szerkesztése
A.FEM	karakterisztikus keret tár
FP_HERM.PAS	Gauss előállítású nyílt felület szerkesztése Hermite interpolációval
A.FHC	karakterisztikus keret tár
FP_BEZR.PAS	Gauss előállítású nyílt felület szerkesztése Bézier közelítéssel
A.FPB	karakterisztikus keret tár

F_P_BETS.PAS	Gauss előállítású nyílt felületek szerkesztése β-spline közelítéssel
F_P_BSPL.PAS	Gauss előállítású nyílt felületek szerkesztése B-spline közelítéssel
A.FBS, Z.FBS	karakterisztikus keret tár
F_P_HSPL.PAS	Gauss előállítású nyílt felületek szerkesztése helyi ellenőrzésű spline interpolációval
F_PSPLN.PAS	Gauss előállítású nyílt felületek szerkesztése spline interpolációval
A.FPS	karakterisztikus keret tár

6. FT ZART

ZM_BETSP.PAS	gömb topológiájú zárt felületek szerkesztése β-spline közelítéssel
ZM_BSPL.PAS	gömb topológiájú zárt felületek szerkesztése B-spline közelítéssel
A.ZMB, B.ZMB, K1	.ZMB, K2.ZMB, K3.ZMB karakterisztikus keret tárak
ZM_HSPL.PAS	gömb topológiájú zárt felületek szerkesztése helyi ellenőrzésű spline interpolációval
ZM_SPLN.PAS	gömb topológiájú zárt felületek szerkesztése spline interpolációval
A.ZMS, B.ZMS, K1.	ZMS, K2.ZMS, K3.ZMS karakterisztikus keret tárak
ZY_BETSP.PAS	gyűrű topológiájú zárt felületek szerkesztése 13-spline közelítéssel
ZY_BSPL.PAS	gyűrű topológiájú zárt felületek szerkesztése B-spline közelítéssel
A.ZYB	karakterisztikus keret tár
ZY_HSPL.PAS	gyűrű topológiájú zárt felületek szerkesztése helyi ellenőrzésű spline interpolációval
ZY_SPLN.PAS	gyűrű topológiájú zárt felületek szerkesztése spline interpolációval
A.ZYS	karakterisztikus keret tár

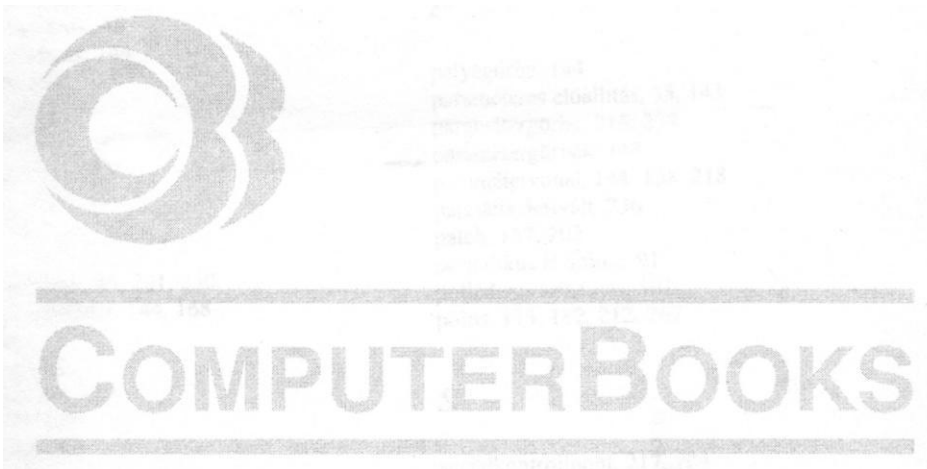
Irodalomjegyzék

1. J.D. Foley, A. Van Dam: *Fundamentals of Interactive Computer Graphics,* Addison-Wesley, 1983.
2. W.M. Newman, R.F. Sproull: *Interaktív Számítógépes Grafika,* Műszaki Könyvkiadó Budapest, 1985.
3. P. Lancaster, K. Salkauskas: *Curve and Surface Fitting,* Academic Press London, 1986.
4. G.E. Farín (ed.): *Geometric Modelling. Algorithms and New Trends,* Society for Industrial and Applied Mathematics, 1987.
5. G.E. Farín: *Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide,* Academic Press London, 1988.
6. Hajós György: *Bevezetés a geometriába* (nyolcadik kiadás), Tankönyvkiadó, Budapest, 1987.
7. B.A. Barsky: *Computer Graphics and Geometric Modelling Using Beta-splines,* Springer-Verlag Berlin Heidelberg, 1988.
8. Strommer Gyula: *Geometria,* Tankönyvkiadó, Budapest, 1988.
9. P. Burger, D. Gillies: *Interactive Computer Graphics. Functional, Procedural and Device-Level Methods,* Addison-Wesley, 1989.
10. Newman, R.F. Sproull: *Principles of Interactive Computer Graphics,* McGraw-Hill, 1989.
11. Lőrincz Pál, Petrich Géza: *Ábrázoló geometria* (negyedik kiadás), Tankönyvkiadó, Budapest, 1989.
12. J.D. Foley, A. van Dam, S.K. Feiner, J.F.Hughes: *Computer Graphics. Principles and practice,* Addison-Wesley, 1990.
13. Pirkó József: *Turbo-Pascal 5.5,* LSI Oktatóközpont Budapest, 1990.
14. Vermes Imre: *Geometria útmutató és példatár,* Tankönyvkiadó, Budapest, 1991.
15. Benkő Tiborné, Benkő László, Kiss Zoltán Tóth Bertalan: *Objektum-orientált programozás Turbo Pascal 6.0-ban. Turbo Vision,* ComputerBooks, Budapest, 1991.
16. S.G. Hoggar: *Mathematics for Computer Graphics,* Cambridge University Press, 1992.
17. M. Bret: *Image Synthesis,* Kluwert Academic Publishers, Dodrecht, 1992.
18. P. Dierckx: *Curve and Surface Fitting with Splines,* Clarendon Press Oxford, 1993.

19. A. Gray: *Modern Differential Geometry of Curves and Surfaces,* CRC Press, Boca Raton, 1993.
20. C. de Boor, K.Höllig, S.Riemenschneider: *Box Splines,* Springer-Verlag Berlin Heidelberg, 1993.
21. Benkő Tiborné, Benkő László, Tamás Péter, Tóth Bertalan: *Programozás Borland Pascal 7.0 rendszerben,* ComputerBooks, Budapest, 1994.
22. Füzi János: *3D grafika és animáció IBM PC-n,* ComputerBooks, Budapest, 1995.
23. Székely Vladimir, Poppe András: *A számítógépes grafika alapjai IBM PC-n,* ComputerBooks, Budapest, 1992.

Tárgymutató

V



Nyomta és kötötte: Séd Nyomda, Szekszárd
Felelős vezető: Dránovits István

CD PANORÁMA

MULTIMÉDIA MAGAZIN MINDENKINEK

- Körkép az új multimédiás CD-kről
- Fülbemászó audio-CD-k
- Az interaktív média hardver- és szoftvereszközei

A CD-mellékleten megelevenedik a magazin

Megrendelhető:

Computer Panoráma Kiadói Kft.

1091 Budapest,
Üllői út 25.

Telefon:
218-3011 /302, 369

Fax: 217-2646

ISBN 963-618-149-7

Ára: 1.993,- Ft (Áfával)