SPECCYALISTA VILÁG

A SPECCYALISTA BARÁTI KÖR LAPJA

SINCLAIR.HU

Az ULAplus szabvány a gyakorlatban - 1. rész

Interjú Jákli Imrével, a ZX-HD atyjával

NEXT BASIC - Kérem a következő! - 1.rész

Assembly ovi 9. rész

Chroma 81 ismertető

ZX-HD ismertető

NEXTBUILD IDE

Játékújdonságok ZX81/ZX Spectrum

Így készült a ZX X-MAS'18

ULA PLUS

LOAD "QUADRON"

2018. 3. SZÁM

Tartalom

Bekösztöntő....3

Arcképcsarnok

Interjú Jákli Imrével, a ZX-HD atyjával....4

LOAD ""

Játékújdonságok - ZX Spectrum....10

Quadron....29

2018-as ZX Spectrum megjelenések....49

Programozástechnika - NEXT

NextBuild IDE....19

Programozástechnika - NEXT BASIC

Kérem a következőt! - 1. rész....43

Programozástechnika - Mélyvíz

Az UlaPlus szabvány a gyakorlatban - 1.rész....22

Így készült a ZX X-MAS '18....33

Programozástechnika - Assembly ovi

Hogyan írjunk játékot ZX Spectrumra - 9. rész....45

HARDVER SIMOGATÓ

ZX-HD....27

HARDVER SIMOGATÓ - ZX81

CHROMA....47

Beköszöntő

Kedves olvasó!

Végre itt a várva várt Speccyalista Világ 2018-as utolsó száma! Ezt a számot több szempontból is mérföldkőnek érzem, egyrészt mert egy eléggé tematikus számot sikerült összehozni, ami az eddigi számoktól emiatt jelentősen eltérés, másrészt most is jópár szerzőt tudtam összeverbúválni az ügy számára.

Alapvetően az ULA+-ra próbáltam fókuszálni, mert ezt egy igazán fontos spectrumos témának tartom manapság, talán sokkal inkább minta a várva várt Spectrum NEXT-tet, ugyanakkor ezt a témát több okból sem sem lehet megkerülni, ezért erre is érdemes oldalakat szentelni.
Mindkét téma esetében cikk sorozatokkal is szolgálunk az elkövetkező számokban.

Most még inkább úgy érzem, hogy folytatni kell ezt a kiadványt. Idéntől megpróbálunk áttérni a negyedéves megjelenésre, hogy talán "lájtosabb" 16-24 oldalas számok ellenére gyakrabban jelentkezhessünk inkább.

Sajnos a külön domain-t, melyet a magazinnak regisztráltunk, még nem sikerült életre lehelni, de előbb-utóbb ez is megtörténik. Ahol majd egyszer az online olvasás mellett mindenféle hírekkel, érdekességekkel szolgálunk a várható témákról.

Most is mint mindig, igyekeztünk az egykori Spectrum Világ szellemiségét megőrizni és minden olvasónknak adni valami számára érdekes olvasmányt.

A szerkesztőségünk címére ugyan most is kevés visszajelzés érkezett, de ezt a vonalat igyekszünk továbbra is képviselni.

A Sinclair.hu portálon már rengeteg hasznos tudást halmoztunk fel az elmúlt években, amelyekből leportalanítjuk az arra érdemes írásokat és időről-időre ezekből az anyagokból is beválogatunk cikkeket, melyek így talán még több olvasóhoz juthatnak el.

Szeretnénk továbbra is, betekintést engedni a berkeinken belül folyó projektekbe. Legyen az akár hardver- vagy szoftverfejlsztés, archiválás, régészkedés.

A Spectrumos témák mellett már ez előző számokban elkezdtünk foglalkozni a nagyöreggel, azaz a ZX81-gyel is, ezzel továbbra sem szeretnénk felhagyni.

Reméljük a "lap" még mindig elnyeri a tetszéseteket és lesznek, akik szívesen bekapcsolódnának a további munkába, akár csak egy-egy cikk erejéig. Ugyanakkor mindenképp várjuk véleményeteket, ötleteiteket, melyekkel jobbá, érdekesebbé tehetjük a MI kis kiadványunkat.

Ezúton szeretném ismét megköszönni a lapunk szerzőinek és munkatársainak azt a sok munkát, melyet befektettek ezen újabb szám elkészítésébe.

2018. december 31.

Kardos Balázs (Balee)

Impresszum

Főszerkesztő: Kardos Balázs (Balee)
Lektorálás: Sári Gábor (sAGA)
Szerkesztés, tördelés: Kardos Balázs (Balee)
Grafika: Molnár Péter (Mopi)

Rovatvezetők: Böszörményi Zoltán (Zboszor), Buzogány Csaba (Makranc), Egri Imre (Zimi), Mezei Róbert (M/ZX), Nagy Dániel, Tanács Imre (Kapitány), Kardos Balázs (Balee), Lakatos Péter (Latyi.ca), László József (FPGA Jocó), Pgyuri, Povázsay Zoltán (Povi), Taletovics Dávid (G.o.D)

Szerzők: Kardos Balázs (Balee), Egri Imre (Zimi), Mezei Róbert (M/ZX), Nagy Dániel, Taletovics Dávid (G.o.D), Tanács Imre (Kapitány), Vándorffy Tamás (Vándor)

Szerkesztőség e-mail címe: speccyalista.vilag.szerkesztoseg@sinclair.hu

Kiadó: Speccyalista Baráti Kör http://sinclair.hu

A magazin honlapja: http://spv.hu

2018. december

ARCKÉPCSARNOK

INTERJÚ JÁKLI IMRÉVEL, A ZX-HD ATYJÁVAL

A Sinclair.hu mindig is fontos feladatának érezte, hogy a felkutassa a hazai „Spectrumos éra” jeles alakjait és emléket állítson akkori tevékenységüknek. De talán napjainkban még ennél is fontosabb, hogy bemutassuk azokat az embereket, akik komoly erőfeszítést tesznek azért, hogy kedvenc gépünket a mai modern eszközökkel is használhassuk.

Jákli Imre neve egy igazán impresszív mai Spectrum illesztő, a ZX-HD boot képernyőjén tűnt fel nekem. Az illesztő Ben Versteeg és Jákli Imre közös gyermeke. Mikor nemrégiben vásároltam egyet Bentől (Bytedelight), akkor megkértem, hogy hozzon miket össze, ha már földiek vagyunk.

Így most Bennek köszönhetően Imrét kérdezhetem a ZX-HD-ről, Spectrumról és egyéb fontos dolgokról.

Sinclair.hu: Elsőként mindjárt szeretném megkérdezni, hogy honnan ismered Ben Versteeget és hogyan kezdtetek dolgozni a ZX-HD projekten?

Jákli Imre:

Egy kicsit hosszú előtörténet fog következni. A Sinclair világba való „visszatérés” 2013 nyarán kezdődött, amikor egy könyvet keresve a kezembe akadt az a ZX81-es, amit Pápán a szüleimnél őrizgettem. A gép még a doktoranduszi éveim alatt a kollégiumból való nyári kiköltözés során került hozzám húszegynéhány évvel korábban. A működőképességéről nem tudtam semmit. Hirtelen kíváncsi lettem, hogy mi van vele. Kiderült, hogy a két 2114-es statikus RAM chip közül az egyik hibás volt, ezért nem működött. A javítást rögtön egy 16K-s memóriabővítéssel oldottam meg. (Később a kiszerelt hibátlan 2114-es chip, egyik barátom által nekem ajándékozott, C64-es alaplapba került szín memóriaként… Szép példája ez annak, amikor a két vetélytárs évek múltán kisegíti egymást… ☺) A „K” kurzor és úgy általában a Sinclair gépek jellegzetes karaktereinek újbóli látványa valamit „átállított” bennem. Elkezdtem erősen vágyni arra, hogy újra Spectrumot nyomkodhassak, de a régi 48K-s Spectrum+-omat elajándékoztam unokatestvéreimnek (nem sokkal később visszaszereztem tőlük… szerencsére nem dobták ki). Fórumokon sokan írták, hogy egy szürke Spectrum +2 ideális gép az „újrakezdőknek”, mert elég jó minőségű és a kompatibilitással sincsenek komolyabb problémák. A Vaterán éppen volt egy eladó. Lecsaptam rá és nagyjából ezzel kezdődött (újra) a Spectrumok iránti nagy szerelem. 2013. augusztus végén rendeltem Bentől egy DivIDE illesztőt hozzá. Közben apósom, látván lelkesedésemet, nekem adta a családi „múzeumban” őrzött Lo-Profile billentyűzetes ISSUE 6A gépüket. Ennek működőképessé tétele volt az első komolyabb Spectrum javításom. Küzdelmes volt, de a végén sikerült a gépet talpra állítani. Jó-jó, most már van két „kemény billentyűs” Spectrumom, de azért a „klasszikus” gumi billentyűzetes gép az „igazi” ZX Spectrum! Ennek a gondolatnak a folyományaként, szintén a Vaterán, vettem egy ISSUE 2-es masinát. Ezzel viszont nem működött a DivIDE megfelelően. Bennel levelezni kezdtem a problémáról. Végül kiderült, hogy az NEC ROM-os gépeket érinti a hiba. Két dióda és egy ellenállás beforrasztása a ROM gyártót kiválasztó jumper helyére megoldotta a gondot. A WoS fórumon „Another cause for DivIDE problems on 48K’s” 2013 novemberi bejegyzésében lehet olvasni erről. A levelezési kapcsolatunk innentől kezdve nagyjából folyamatos volt.

A digitális videokimenet kérdése régóta foglalkoztatta a Spectrumos hardver fejlesztőket. Ahogy emlékszem, a DVI nem igazán tetszett az embereknek, mert úgy tűnt, hogy a helyét át fogja venni a HDMI. A gond csak az volt, hogy a HDMI Forum elég jelentős jogdíjat követelt a végfelhasználóknak szánt eszközök gyártóitól. Ez egy ilyen viszonylag szűk piacra, kis darabszámban készülő terméknél az alkalmazást szinte lehetetlenné tenné.

Ekkor jött a képbe a Raspberry Pi. Raspberry Pi-t korábban a gyerekeknek készült „busz kijelzőben” használtam. A Linux alapokon nyugvó célhardver egy

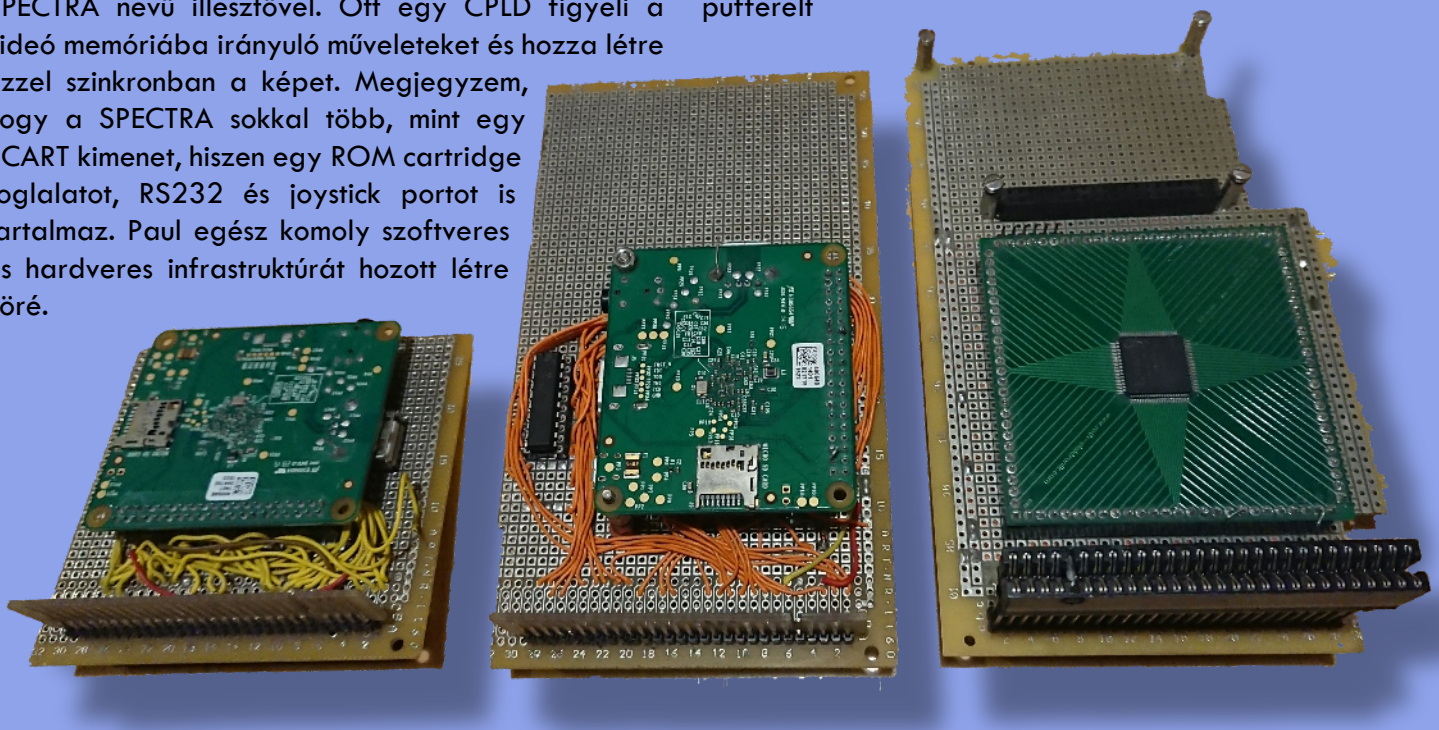
128x16-os színes RGB LED mátrixot hajt meg, a web felületén kiválasztható a BKK Futár online szolgáltatásához kapcsolódva a viszonylat és a menet. Ki tudja írni a viszonylatot és a célállomást vagy a megállókat. Lehet léptetni a megállókat, ajtózárásra figyelmeztető hangot adni (a kiválasztott menetben éppen közlekedő jármű típusa szerint), valamint egy online beszédszintetizátorhoz kapcsolódva be tudja mondani a megálló neveket is. Pár hete a web szerver fel lett vértezve egy olyan funkcióval, ami a hozzá kapcsolódó web böngészőben a Futár utastéri kijelzőnek megfelelő megjelenítést biztosít. Ezzel a szereppel megmentve a kivénhedt ASUS TF101 tabletet a selejtezéstől. A fiúk már bejelentették az igényt a járművezetőnél lévő terminál megvalósítására is… Az elképzelhető, hogy egy Windows 10 IoT Core operációs rendszert futtató 7" TFT érintőképernyős Raspberry Pi 3-assal lesz megoldva.

2015. november végén írtam Bennek, hogy láttam a Raspberry Pi Zero-t, ami igen kicsi és kiválóan alkalmazható lenne beágyazott rendszerekben. Másnapi levelében írta, hogy ő is találkozott vele és az érdekelné, hogy esetleg megoldható lenne-e a ZX Spectrum videó (és audió) vonatkozású jeleinek követésével és tárolásával, akár az ULA nélkül, a tűéles HDMI videókiment előállítása. Én rögtön rávágtam, hogy láttam már hasonló dolgot, mert a színes Primo videokimenete is nagyjából így lett megvalósítva. Ott a „videokártyán" egy második Z80-as processzor követi a videokimenetre kerülő biteket és igen szigorú időzítéssel előállítja a színes képet. A Spectrumos világban az általam igen nagyra becsült Paul Farrow hozott létre hasonlót a SPECTRA nevű illesztővel. Ott egy CPLD figyeli a videó memóriába irányuló műveleteket és hozza létre ezzel szinkronban a képet. Megjegyzem, hogy a SPECTRA sokkal több, mint egy SCART kimenet, hiszen egy ROM cartridge foglalatot, RS232 és joystick portot is tartalmaz. Paul egész komoly szoftveres és hardveres infrastruktúrát hozott létre köré.

Bizalmába fogadott, így vele is dolgoztam együtt a ZXC ROM kártyák és az RS-232 rutinok tesztelése kapcsán.

Emiatt egy kicsit most zavarban érzem magam, mert a ZX-HD működését tekintve versenytársa lett a SPECTRA-nak.

Visszatérve a ZX-HD-hez: egyáltalán nem voltunk biztosak abban, hogy a Pi elég gyors lesz a memória és I/O műveletek követésére. Akkoriban a Pi Zero szinte beszerezhetetlen volt Magyarországon (és a világon), ezért az első teszteket a Pi A+ modelljén hajtottam végre 2015 december végén és 2016 első napjaiban. (Addigra már szinte hagyománnyá vált, hogy Szilveszter éjszakáján, amikor legtöbben buliznak, valamilyen jó kis hardver/firmware fejlesztés zajlik…) Az első, legegyszerűbb hardver verzió még jelszint illesztést sem tartalmazott. Gyakran lemaradt a Pi a Spectrum memóriaműveleteiről, de arra jó volt, hogy lássuk: az ötlet működhet. A második 74HC373-asokkal pufferelt



változat január második hetére lett kész, amit egy harmadik CPLD alapú követett. Ezek a prototípusok kézi vezetékezéssel, próbapanelen készültek. A firmware „bare-metal” módban, operációs rendszer nélkül működik. Erre azért van szükség, mert a memória és I/O műveletek követése csak a hardver erőforrásainak maximális kihasználásával lehetséges. (Mintha a ’80-as években lennénk...) A kernelt C programnyelven írtam, de folyamatosan ellenőriznem kellett a GCC fordító által létrehozott assembly listafájlt, mert időnként „átértelmezte” a fordító az általam leírtakat és szétesett az időzítés. Általában hetente közöltem Bennel, hogy nincs tovább, már sikerült kihozni a hardverből a maximumot, ez nem fog menni, de a végén csak-csak mindig találtam valamilyen módot a sebesség növelésére. Mindezt anélkül, hogy a rendszert a kezdeti verziókhoz hasonlóan túl kelljen hajtani. Kb. 2016. februárra már működött az ULA+ kód, márciusra kész volt a 48K/128K modell detektálása és a „pontos”

sebesség kalibrálás, valamint a 128K-hoz tartozó dupla képernyő memória kezelése. A kalibrálásra azért volt szükség, hogy a Pi képernyő memóriájában a tartalom módosítása egy képkocka csúszással, lehetőleg az eredeti Spectrumos időzítés szerint történjen meg. Ehhez a Pi órajelét és a képernyősor megjelenítését vezérlő megszakítás frekvenciáját igazítottam hozzá a Spectruméhoz. A fenti funkciók megvalósítását követő időszakban drasztikusan csökkentenem kellett a fejlesztésre fordított időt. Annyira lelkesített a fejlesztés, hogy sajnos egy kicsit „túlpörögtem a témát”. A napi munka után szinte minden este és éjszaka a ZX-HD-n dolgoztam. A hivatalos megjelenésig terjedő időszakban apróbb hibajavításokat hajtottam végre, valamint a konfigurációs paraméterek beolvasását és kezelését oldottam meg. Sokat szenvedtem azzal, hogy miközben a Pi a képernyő memóriából előállítja a monitoron megjelenített képet, az az átskálázás során időnként kissé elmosódottá válik. Ezt végül csak részben sikerült orvosolni és egy kicsit monitorfüggő a kép minősége. Végül a „vájtszeműek” számára létrehoztam egy olyan opciót, ahol a megjelenített paletta a felhasználó ízlése szerint átalakítható. Az élénk RGB színek helyett a Spectrum eredeti színeihez jobban hasonlító árnyalatok is beállíthatóak. A firmware az alapos tesztelésnek köszönhetően elég jól sikerült, mert úgy emlékszem, hogy a kibocsátás óta nem volt újabb verzió.

Sinclair.hu: Lesz-e még folytatása ennek projektnek?

Jákli Imre:

A Spectrum 35 találkozón valaki felvetette, hogy lehetne csinálni kétmonitoros módot két ZX-HD felhasználásával (igen, lehet többet is csatlakoztatni). Ehhez 128K módban le kellene tiltani a képernyőlapozást és a firmware-t tároló SD kártyán egy konfigurációs paraméterrel „fixálni" az egyik és a másik ZX-HD-t az elsődleges és a másodlagos videómemóriára.

A ZX-HD nagy hiányossága, hogy csak a képet viszi át, a hangot sajnos nem. Ennek elkészítése nagy áttörés lenne. A gond, hogy egyelőre fogalmam sincs, hogy hogyan lehetne ezt megoldani. Sajnos a Broadcom nem dokumentálta le a „bare-metal” szinten való megvalósításhoz szükséges mélységben a Pi hardverét. Így ha eszembe jut és van egy kis időm, akkor kb. félévente körül szoktam nézni, hogy az elmúlt időszakban esetleg tettek-e közzé újabb információt a hangkezeléssel kapcsolatban. Érdekes, hogy a Pi bootolásakor először a GPU firmware töltődik be, ami inicializálja a neki alárendelt CPU-t (itt a GPU a főnök). Ez a firmware biztosan tartalmaz rutinokat a HDMI hang kezeléséhez (a Linux ezeket használja), de ezeket a rutinokat egyelőre „bare-metal” módban nem tudtam megszólítani. A másik akadály eléggé objektív: nem nagyon maradt szabad „időrés” a firmware-ben a hang kezeléséhez. Kisebb problémák akadtak a spanyol Investronica Spectrumok kapcsán. A tesztelést csak úgy láttam megoldhatónak, hogy beszerzek egy 48K-s és egy 128K-s gépet. A 128K spanyol „Toastrack” hibátlanul működött, de a 48K-s verzió teljesen más ULA-val rendelkezik (Texas Instruments TAHC10). Az időzítése a 128K-s verzióval megegyező, ezért nem tudja helyesen érzékelni a ZX-HD a Spectrum modellt. Ilyenkor az SD kártyán tárolt konfigurációs paraméterrel lehet letiltani az automatikus detektálást és kiválasztani a helyes Spectrum modellt. Az Investronica gépek után kíváncsi lettem, hogy mi a helyzet a Timex által gyártott masinákkal. Abból is begyűjtöttem mind a három változatot: az NTSC-s

Timex Sinclair 2068-as gépet a Vaterán vettem, egy PAL rendszerű Timex Computer 2048-ast és egy ezüst színű 2068-ast az eBay-en szereztem egy portugál eladótól, akivel azóta is tartjuk a kapcsolatot. A Timex gépek megbízhatóbb billentyűzettel rendelkeznek, mint a gumibillentyűs Spectrumok. A Commodore gépekhez hasonlóan egy vezető gumi ér hozzá a gombhoz tartozó fém érintkezőhöz és az zárja az áramkört. Ezek az érintkezők az évek során szennyeződhetnek, oxidálódhatnak és a billentyű csak erős lenyomásra reagál, vagy teljesen használhatatlanná válik. Ilyenkor nincs más megoldás, mint a billentyűzet szétszedése és megtisztítása. Ehhez viszont el kell távolítani a billentyűk jelentését is tartalmazó öntapadós fóliát, ami alatt vannak a csavarok. A vékonyabb fóliával rendelkező modellek esetében nekem ezt nem sikerült sérülésmentesen megtenni. A gyártó egyes modelleket vastagabb fóliával látott el, amelyek óvatosan leszedhetők voltak. A fólia pótlására próbáltunk a portugál sráccal megoldást találni, ami többé-kevésbé sikerült is.

Amikor a fentebb említett „nyugati klónokat” beszereztem, elkezdett motoszkálni bennem a gondolat, hogy miért csak a „nyugati klónokkal” foglalkozom, hiszen a „keleti blokkban” is szép számmal születtek Spectrum másolatok, amelyek sok esetben különbek voltak, mint az eredeti. Egy novoszibirszki sráctól vásároltam pár éve egy Pentagon replika kitet. A legkisebb fiammal összeforrasztottuk és egy DkTronics dobozban helyeztük el. Nem értettem addig, hogy jónéhány demó miért fut olyan furcsán a 128K-s Spectrumon... Most már tudom, hogy azért, mert 128K-s Pentagonra készítették őket. A Pentagon esetében várakozás nélkül tud hozzáférni a CPU és a videó áramkör a memóriához, ezért a programok érezhetően gyorsabban futnak. A floppy meghajtóval felvértezett „ZX Spectrum” pedig álomszerű a magnóval, vagy akár a Microdrive-val összevetve. Apropó Microdrive: talán érdemes megemlítenem, hogy vagy fél évet töltöttem el azzal, hogy összehozzam a maximális 8 ZX Microdrive-ból álló konfigurációt (ROM cserével később 10-ig tudtam felmenni…). A szükséges meghajtók és illesztőelemek beszerzése/hackelése után látszólag egyszerűnek tűnt a feladat: összedugom, és már működik is. Sajnos 5-6 egység felett az amúgy hibátlan meghajtók is teljesen összezavarodtak. Sok-sok kísérletezés után végül két kondenzátor beépítésével, amelyeknek a helye ott van a nyáklapon, sikerült működésre bírni az íróasztalt átérő „szörnyeteget”. (https://www.youtube.com/watch?v=fJSGElZn8A8&t=66s)

Sinclair.hu: Ben más projektjeiben is részt veszel?

Jákli Imre:

A retró gépek mellett a 3D nyomtatás is érdekelni kezdett és készítettem a Ben-től 2014 elején megvásárolt DivMMC EnJoy! illesztőhöz egy 3D nyomtatott házat. A ház sorozatgyártásra alkalmassá tétele volt az első közös „projektünk”. Mindkettőnk számára izgalmas volt az 1000. példányhoz egyedileg gyártott 24 karátos aranyozású fémház elkészítése. Ben megadta a módját a jubileumi példánynak! A fém alapanyag miatt számos ponton át kellett tervezni, a műanyag por alapú, SLS technológiához készített változatot.
Ezt követte a PlusD Lite hajlékony lemezmeghajtó vezérlőjének hibakeresése, ahol sikerült megtalálnom, hogy miért akadt le időnként a gép az inicializálás közben. A PlusD Lite-ba Ben nem épített be parallel printer csatlakozást, mert feleslegesnek érezte, de a ROM kódban benne maradt az inicializálás, ahol időnként végtelen ciklusba került a program, várva a nem létező periféria válaszára.

Ben az utóbbi időben küldött tesztelésre a frissen kifejlesztett SD kártya illesztőiből. Tudja, hogy szinte az összes alaplap verzióval rendelkezem, és ki tudom próbálni sokféle készülékkel az új hardvert. Talán érdekes megjegyezni, hogy személyesen először tavaly október végén a Cambridge-ben tartott Spectrum 35 rendezvényen találkoztunk.

Sinclair.hu: Lesz még folytatása ennek a projektnek?

Jákli Imre:

Ez részben rajtam múlik. Amióta Ben a retro számítógépekkel kapcsolatos tevékenységére alapozva egy családi vállalkozást indított, azóta eléggé elfoglalt lett. Engem meg a lakásvásárlás és eladás kötött le az elmúlt fél évben. Ha ezek a hullámok kicsit elcsendesednek, valószínűleg újult erőre kaphat az együttműködésünk. Aztán lehet, hogy a másik nagy kedvencem a modellvasút fogja a következő évet meghatározni. A nagyobb hely lehetőséget adhat arra, hogy átgondoljuk a terepasztalt és egy sokkal változatosabb és életszerűbb vasútüzemet hozzunk létre. (Egy volt és egy aktív gyermekvasutas is van a családban, tehát az igény eléggé megalapozott.) Elképzelhető, hogy a retro számítógép korszak után újra a modellvasút fogja egy ideig a vezető szerepet játszani. Természetesen a ZX Spectrum is össze lett már kapcsolva a digitális vezérlésű (éppen szétszedett) terepasztallal. Készült egy kis C program, ami a zseniális Spectranet hálózati interfész segítségével UDP csomagokat képes küldeni a terepasztalt vezérlő Roco Z21 egységnek. Ezen keresztül pedig a mozdonyokat lehet irányítani ZX Spectrumról is. Persze ezt üzemszerűen nem használjuk, inkább csak a poén kedvéért csináltam meg a programot. Meg hogy kicsit megismerjem a Z88dk fejlesztő környezetet. Nagyon élveztem, hogy C-ben, többé-kevésbé szabványos socket könyvtárral dolgozhattam.

Sinclair.hu: Térjünk vissza a kezdetekhez. Mikor találkoztál a számítástechnikával, a Spectrummal először?

Jákli Imre:

A történet 1984-ben kezdődött, amikor keresztanyáméknál nyaraltunk. A nyaralás során unokatestvérem férje – aki fizikusként a KFKI-ban dolgozott - hozta magával a Casio PB-100-as kis kézi számítógépét. Ez roppant módon érdekelni kezdett és kértem, hogy mutassa meg, hogy hogyan működik. Ő vezetett be a BASIC nyelv rejtelmeibe. Amikor ősszel elkezdődött a suli, terjesztettem tovább a „ragályt” és a nyolcadikos haverokkal megalapítottuk a „CPC Software” nevű társaságot. Gépről inkább csak álmodoztunk. Autodidakta módon próbálkoztunk programírással. Egyik barátom édesanyja a pápai Petőfi Sándor gimnáziumban dolgozott, így ő bejuthatott a gimnázium számítógép termébe, ahol aztán bepötyögte a programokat, amit papírra leírtunk. Közben a pápai Jókai Művelődési Központban elindult szombatonként egy számítástechnikai szakkör. Ott találkoztam először élő közelségben a ZX Spectrummal. Mondanom sem kell, hogy megkedveltem a gumibillentyűs masinát. Közben a „bizományi” kirakatában megjelent egy ZX81-es. Ha arra vitt az utam, mindig áhítozva nézegettem. Tavaszra elkészült ötünk munkájaként (a korábban említett „távprogramozással”) a 8 modulból álló „Kémia” nevű program, ami tartalmazott pár kémiában hasznos dolgot: periódusos rendszert, jellemerősségi sort, oldhatósági táblázatot, koncentráció átszámítást, oldat elegyítési és aránypáros számításokat stb. Felmerülhet a kérdés a Tisztelt olvasóban, hogy hogyan jön ide a kémia? Úgy jön ide, hogy hetedikes koromban elkezdtem járni a TIT által szervezett Kis Kémikusok Baráti Köre (KKBK) szakkörre, ahova a „CPC-s” barátaim is jártak. Így adódott a dolog, hogy kapcsoljuk össze a kettőt. Ez a két téma aztán későbbi életemben is elkísért. Sajnos a gimnáziumi években a sors a korábbi barátaimtól elsodort, a csapat szétszéledt. Nagybátyámék révén Belgiumból, 1985 nyarán hozzájutottam a korábban említett Spectrum+ 48K-s géphez, viszont ezzel egyedül maradtam, mert az osztályban mindenkinek C64-ese volt. Ez persze nem vette el a kedvemet és továbbra is szívesen ültem le programozni és játszani a gép elé. Vicces volt, hogy a Spectrum „zabálta” a képcsöveket. Általában évente kellett cserélni az Uranus fekete-fehér TV-ben a lengyel képcsöveket. Azt hiszem velünk nem csináltak jó boltot az általánydíjas javítási szerződéssel a helyi Gelkások. Amikor a pályaválasztásra került a sor, akkor is a kémia és a programozás versengett egymással. Végül a kémia győzött, az ELTE-re jelentkeztem vegyésznek. Az ELTE Kémai Tanszékcsoportnál hamar kapcsolatba kerültem a helyi számítástechnikai csoporttal, ahol akkoriban úttörőnek számító technológiákkal találkozhattam. Tudományos diákköri dolgozatot 1993-ban a matematika-fizika-számítástechnika tanár szakra járó Balogh Laci barátommal közösen készített – NET & NET névre keresztelt – peer-to-peer hálózati operációs rendszerből készítettünk. Laci készítette a hardver közeli hálózati protokoll részt én pedig a BIOS és MS-DOS illesztő rutinokat. Persze assemblyben, hogy kevesebb legyen a memóriafoglalásunk és gyorsabbak legyünk, mint a „versenytárs” termékek. Ezzel a XXI. OTDK-n az informatika szekcióban különdíjat kaptunk. Két évvel később egy teljesen más témában kémiából készítettem TDK dolgozatot, az a kémiai és vegyipari szekcióban kiemelt I. díjat kapott.

A diplomamunkámat és később a doktori értekezésemet is olyan témákból írtam, ahol a kémia

és az informatikai szorosan összefonódtak. Talán az informatika súlya nagyobb manapság az életemben. Inkább érzem magam informatikusnak, mint vegyésznek (Közben úgy érzem, hogy egyikhez sem értek igazán.) Sokan meglepődnek informatikus környezetben, amikor kiderül, hogy vegyész vagyok. Talán itt érdemes megemlíteni édesapámat, aki műszerészként is dolgozott. Sokszor javított otthon mindenféle elektronikai eszközöket. Mindig nagy érdeklődéssel figyeltem, ahogy dolgozik. Ő tanított meg forrasztani és a mérőműszerek használatára. A Pápai Vasas Tekeszakosztályának a pályáját munkatársaival együtt éveken keresztül felügyelte. Ha valami gond volt vele, akkor ők hárították el a hibákat. A pálya korszerűsítése után a régi pálya „felét" megvette egy vállalkozó és Szanyban egy kocsmában működött tovább. Oda rendszeresen elkísértem és együtt csináltuk a karbantartási feladatokat. Hálás vagyok neki, hogy megszerettette velem az elektronikát. A tőle tanultaknak nagy hasznát vettem mostanában a régi gépek javítása során.

Sinclair.hu: Mivel foglalkozol jelenleg?

Jákli Imre:
Csomagolással... ☺. Hamarosan költözünk. Már nyáron bedobozoltam a gyűjteményem jelentős részét, hogy az eladni kívánt lakásunk ne egy raktárra emlékeztessen. Csak az éppen szerelés alatt álló dolgokat és a számomra nagyon kedves és értékes gépeket nem tettem be a bérelt raktárba. A Spectrumok nagy része azóta is ott pihen. Remélem, hamarosan ismét viszontláthatom őket. Nemrég „betáraztam" javításra három szovjet ZX Spectrum klónt (Parus VI-201, Ikar-64, Olimpic-C) és egy Videoton TV Computert, amiknek a feltámasztása lesz a következő „retro projekt", ha az új helyünkön rendeződnek a viszonyok.

Sinclair.hu: Említetted a gyerekeket. A család hogyan viszonyul a régi számítógépekhez?

Jákli Imre: Van három fiam és egy lányom. A lányomat nem igazán érdekli a dolog. Ő inkább humán érdeklődésű, szereti a zenét, a hegedű és a zongora után most basszusgitározni tanul. A fiúk szívesen leülnek játszani a régi gépekkel. Az egyik kedvenc Spectrumos játékuk a Bruce Lee nevű program, amivel ketten is tudnak játszani. Általában össze szoktak dolgozni és nem egymás kiütésére törekszenek. C64-esen a Frogger, vagy a szintén ketten játszható Wiz of War a favorit. A legkisebb fiam Balázs 6-7 évesen (még) szívesen jött forrasztani. Sokat segített a Pentagon replika összerakásában. A Bentől kapott 48K-s Harlequin kitet teljesen ő forrasztotta össze, de a Spectrum alaplap kondenzátor cserékben is szokott segédkezni.

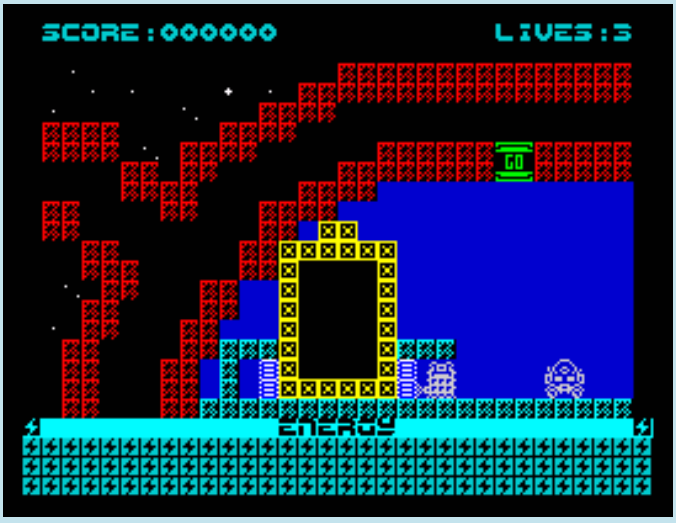
Az utóbbi időben lelkesedése mintha alábbhagyott volna. Nagy szeretettel említeném meg a páromat, aki végtelen türelemmel viseltetett a helyet, időt és pénzt nem kímélő hobbijaim iránt. Párom régóta győzködött, hogy jó lenne váltani, nagyobb élettér kell a családnak. (Amíg árultuk a lakást, sokan megjegyezték, hogy a helykihasználást sikerült igen magas szintre fejleszteni.) Sokáig ellenáltam a költözés gondolatának, de aztán a felhalmozódó géppark „meggyőzött" arról, hogy tarthatatlan a helyzet, lépni kell! Ha nem lett volna ez a hobbim, akkor páromnak valószínűleg nehezebb lett volna rávennie a váltásra. A költözés jó alapot teremt az élet dolgainak átgondolására, a hangsúlyok átértékelésére. A következő időszakban a legfontosabb a családi otthon újbóli megteremtése lesz, és aztán jöhet minden más. Sokszor megfogadtam, hogy ez lesz az utolsó régi gép, amit begyűjtök és megpróbálok felélesztени, aztán mindig jött egy újabb… Az utóbbi időben egyre bizonytalanabb állapotú gépeket szereztem be, hogy a „műszaki kihívás" a lehető legtovább kitartson (ld. betárazott gépek). A nyáron hónapokig „restauráltam" egy épeszű gyűjtőnél biztosan alkatrész bányaként szolgáló C128D-t. Ezzel párhuzamosan az Orel BK-08-as javítása zajlott. A „fél gépet" ki kellett cserélni, mire hibátlanul megmozdult a szerkezet. Végtelenül érdekes volt a cirill betűs módban is működő „ZX Spectrumot" látni! Azt hiszem csak ezért megérte ezzel foglalkozni. Nyilván a gyűjtemény bővítése – aminek célja részben a hardverfejlesztésekhez megfelelő teszt gépek beszerzése volt – nem folytatható minden határon túl. A mennyiségi fejlődés után újra olyan fejlesztések felé kell fordulnom, amelyek ezeknek a régi hardvereknek a használhatóságát javítanák. Vannak a tarsolyomban ötletek, amiket meg szeretnék valósítani…

Az interjút készítette:
Kardos Balázs (Balee)

JÁTÉKÚJDONSÁGOK - ZX SPECTRUM

ASTRONAUT LABYRINTH

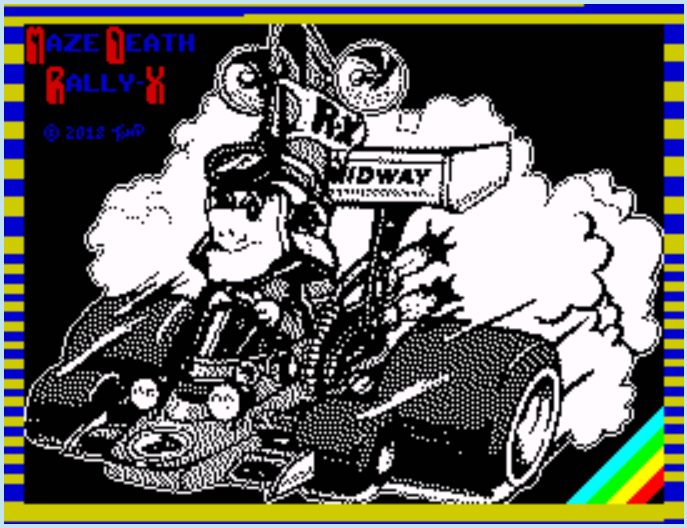
Szerzők: **Jaime Grilo (programozás, grafika), Andy Green (betöltőkép), Visual (zene), Damien Guard**
HW igény: **48K/128K**
Stílus: **arcade/akció**
Vezérlés: **Billentyűzet (definiálható, alapértelmezésben Q, A, O, P, M, Space, H - fel, le, balra, jobbra, tűz, teleport, pillanat állj) és Kempston, Sinclair Joystick**
Megjelenés: **2018. november, ingyenes**



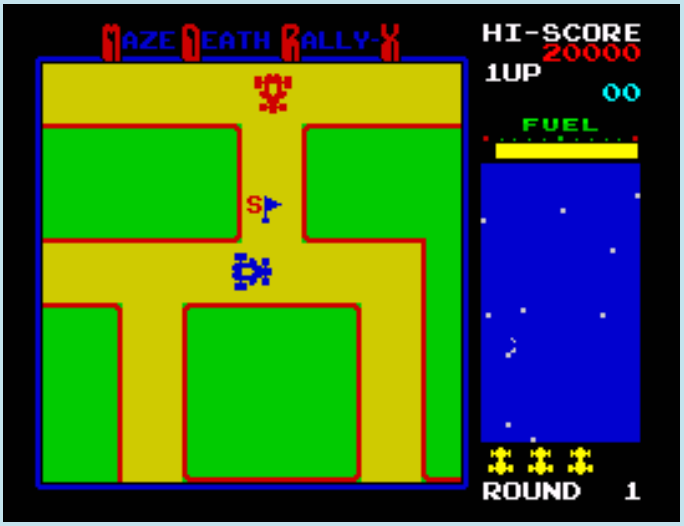
Ha jól számolom, akkor Jaime Grilo kilencedik AGD-vel készült játéka az Astronaut Labyrinth, egész pontosan ez a játék az AGDX rendszer segítségével született meg. Az AGDX az Arcade Game Designer eXpert verziója, amit Jonathan Cauldwell AGD-jének módosításával Allan Turvey készített el.

A játék főszereplője Brian Bentley, egy űrkutató, akit beszívott egy intergalaktikus portál, és egy űrhajós alakú labirintus belsejében landolt űrhajójával. Briannek össze kell gyűjtenie egy űrhajós rajzának 6 részét, hogy megtalálja a teleport kulcsát és kijusson a labirintusból. Sajnos a részek szétszóródtak a labirintusban. Ezen felül még meg kell találnia egy rejtett helyen lévő tápegységet is a kijutáshoz. Brian egyszerre egy tárgyat szállíthat. Ha már van nála egy tárgy, azt a képernyő tetején villogó [AL] felirat jelzi. Az űrhajós darabjait, vagyis a képének hat részét a kezdőképernyőre kell visszavinni, ahol egy monumentális képpé állnak majd össze a darabok. Vigyázz a 'STOP/GO' rendszerajtókkal, a 'STOP' állapot elpusztít mindent, ami a nyílásban van, a 'GO' átereszti az űrhajódat. Gyűjtögetni lehet még betűket is (az ASTRONAUT LABYRINTH 19 betűjét), melyek helyét az AL feliratú tárgy jelzi, ezekrét pontokat lehet kapni.

MAZE DEATH RALLY-X



Szerző: **Tom Dalby**
HW igény: **48K+AY**
Stílus: **arcade/autóverseny**
Vezérlés: **Billentyûzet (Q, A, O, P, B..Sp - fel, le, balra, jobbra, tűz) és Kempston, Sinclair, Cursor Joystick**
Megjelenés: **2018. november, ingyenes**



Tom Dalby harmadik Spectrum játékával, a Maze Death Rally-X-szel a ZX-DEV M.I.A. - Remakes-re nevezett be. A játék a Namco Rally-X arcade klasszikusának portja, de a szerző szándéka szerint a PSS 1983-ban megjelentetett 16K-s Maze Death Race című játékának a remékje is egyben.

Ta vagy a kék versenyautó pilótája egy többirányú, görgethető labirintusban, ahol piros versenyautók vadásznak rád. Veszélyesnek tűnhet a szűk folyosókon száguldozni, pláne üldözőkkel a hátsódban, de ne aggódj, az autód félig-meddig önvezető, ha beleszaladnál egy falba, akkor autód elkanyarodik balra vagy jobbra, és huss, megy tovább (persze nem biztos, hogy a neked megfelelő irányba). Ez az automatizmus sajnos még csak a 90 fokos fordulókra van megoldva, ha egy kupac törmeléknek száguldasz neki, azon nem tud segíteni. Szóval legyél éber, figyelj minden pillanatban. Ellenségeid folyamatosan követnek, ha többen vannak, még össze is dolgoznak, sőt lesznek olyan labirintusok, ahol nálad gyorsabbak is lesznek. Ellenük jól megválasztott útvonallal, és ha túl közel járnak hozzád, akkor füst eregetésével védekezhetsz (tűz gombbal). No, és nem is céltalanul száguldozol, minden labirintusban tíz zászló van elszórva, ezeket kell összeszedned, hogy átjuss a következő útvesztőbe.

A zászlók értéke fokozatosan nő, az első 100 pont, a második a 200, a harmadik a 300, és így tovább. Vannak speciális zászlók is ("S" betűvel jelezve), ami a további felvett zászlók értékét megduplázza. Minden futam után kapsz egy üzemanyag-bónuszt is, attól függően, hogy mennyi üzemanyag marad a tankodban. Az üzemanyag fogyása két dologtól függ, egyrészt nyilvánvalóan a megtett út hosszától, másrészt a kieregetett füst mennyiségétől. A pályák egyre nehezednek, ami vagy több, vagy gyorsabb ellenségben, vagy több útakadályban, vagy ezek ötvözésében nyilvánul meg.

THE EGGSTERMINATOR

Kiadó: **The Death Squad**
Szerzők: **Davey Sludge (programozás), Yerzmyey (zene)**
HW igény: **48K+AY**
Stílus: **arcade/akció**
Vezérlés: **Billentyűzet (Q, A, O, P, Space - fel, le, balra, jobbra, tűz) és Sinclair Joystick**
Megjelenés: **2018. november, ingyenes**

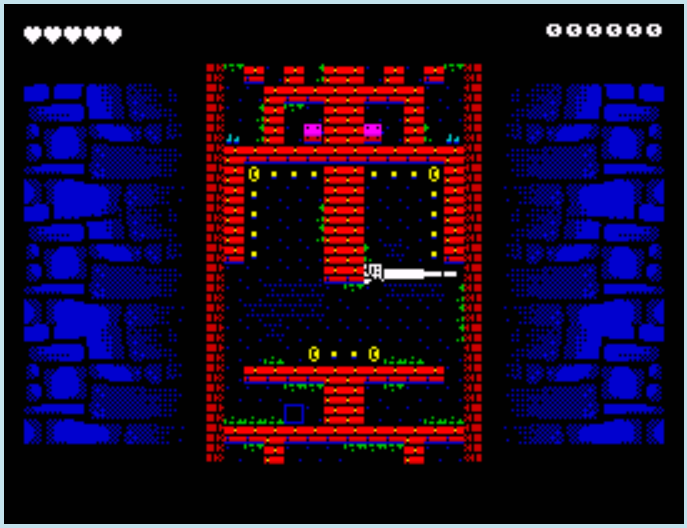
Davey új játéka a mai világból merített ihletet, egy olyan problémára hívja fel a figyelmet, amivel még talán nem foglalkozott senki, de mindenki csak suttogva mer róla beszélni, mindenki retteg tőle... Mert valójában mi is történne, ha a terroristák mérgeskígyókat rejtenének el húsvéti csokitojásokban? Az egyetlen logikus válasz a kérdésre: hívnánk az Eggsterminatort!

A játékban te vagy az Eggsterminator, aki ismerős lehet más játékokból, ugyanis pont úgy néz ki, mint Scott, a bunyós srác, de ezen lépj is túl gyorsan, koncentrálj a feladatra, mert időben meg kell semmisítened az összes tojást, mielőtt még egy csokival kikent szájú kisgyerek beléjük harap! De nem minden húsvéti tojás látszik ám, nem bizony, van amelyik csak akkor tűnik elő, ha a piros dobozokat egymás mellé lökdösöd, hogy egy nagyobb dobozt alkossanak. Közben persze nem vagy egyedül, zaklatnak a kisebb-nagyobb-hosszabb terroristák, de nem hiába vagy te a híres Eggsterminator, gyorsabb is vagy náluk, meg van, amelyiknél erősebb is, intézd el őket!

OLD TOWER

Kiadó: **RetroSouls Team**
Szerzők: **Denis Grachev (programozás, grafika), Oleg Nikitin (zene), Ivan Seleznev (betöltőkép)**
HW igény: **48K, 128K**
Stílus: **arcade/akció**
Vezérlés: **Billentyűzet (Q, A, O, P, M vagy Sp - fel, le, balra, jobbra, karakterek váltása) és Kempston, Sinclair Joystick + Kiegészítő billentyű: R - pálya újrakezdése**
Megjelenés: **2018. november, ingyenes**

Denis Grachev a ZX-DEV M.I.A. - Remakes játékgyártó versenyre nevezett be az Old Towerrel, amire két oka is megvolt, ugyanis vagy MIA (Missing in Action) státuszú játék megírásával, vagy egy más platformon már megjelent játék átiratával lehetett beszállni. A két okot a szerző így foglalta össze: "Találtam egy hardcore MIA, soha ki nem adott játékot, a Jumpery-t, és úgy döntöttem, hogy



elindulok a versenyen (meglepetés!) Valójában ez egy remake, amelyet egy másik platformból származó játék inspirál és még nincs Speccyn." Tehát inkább a remake-oldalt erősíti az Old Tower, a játék eredetije az Androidra és iOS-re kiadott Tomb of the Mask. A megvalósítás érdekessége, hogy Denis használta először a Nirvana motort képernyőt scrollozó játékhoz, így az Old Tower a világ első multicolor scrollerje ZX Spectrumon.

Felfedező vagy egy toronyban, amely tele van halálos csapdákkal és denevérekkel. Gyűjtsd össze a kincseket, és megnyílik az út a következő szintre.
Kis felfedeződ egy villámléptű emberke, gyakorlatilag faltól falig, vagy valamilyen ütközőig rohan, ráadásul fel-le, jobbra, balra, mint a villám. Közben felkapkodja a trehány módon elszórt kincseket. Játék közben folyamatosan új trükköket fogsz megtanulni, az egy átsuhanással aktiválható falat, a kevésbé strapabíró, törhető falat, vagy hogy nem is vagy egyedül…

MANIC MIXUP

Szerzők: **Andy Ford, Ian Rushforth**
HW igény: **48K**
Stílus: **Manic Miner MOD**
Vezérlés: **Billentyűzet (Q, E, T, U, O - balra, W, R, Y, I, P - jobbra, CS..Sp - ugrás, CS+Sp - kilépés)**
Megjelenés: **2018. október, ingyenes**

Andy Ford és Ian Rushforth a Jet Set Wiily & Manic Miner Community oszlopos tagjai 2018-ban már sokadik MOD-jukat készítették el Miner Willy valamelyik játékának. Ezúttal kettőt is, ugyanis a Manic Mixup csomagja tartalmazza a 2017. júliusban megjelent Jet Set Mini című MOD-ot is.

Szegény öreg Willyt megviselték az évtizedek, folyamatos tivornyázasa pénzügyileg megrendítette a helyzetét. Annak ellenére, hogy kastélya minimalizálva lett (Jet Set Mini), egyre nehezebben tudja otthonát fenntartani.

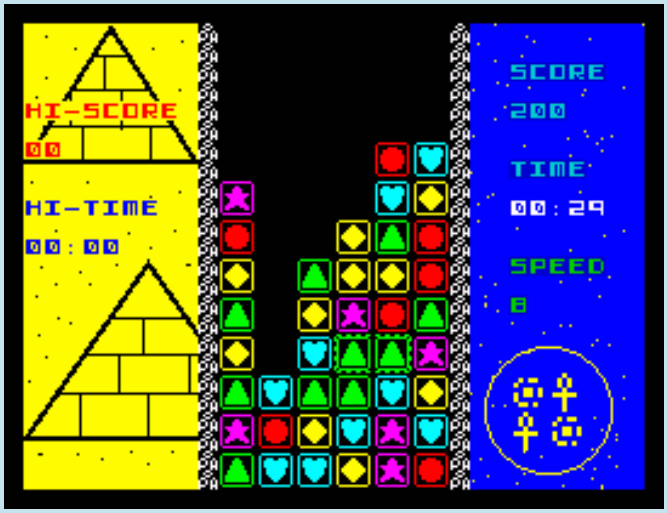
Miner Willy nem tehet mást, felveszi viseltes bányászsisakját, és visszatér a föld alá, ahol 35 évvel korábban már szerencséje volt, de mintha a 35 év alatt valami megváltozott volna...

O-PUZZ ATTACK!!

Szerző: **oblo**
HW igény: **48K**
Stílus: **kirakós**
Vezérlés: **Billentyűzet (újradefiniálható, alapértelmezésben Q, A, O, P - fel, le, balra, jobbra, Sp - csere, B - emelés) és Sinclair Joystick**
Megjelenés: **2018. december, ingyenes**

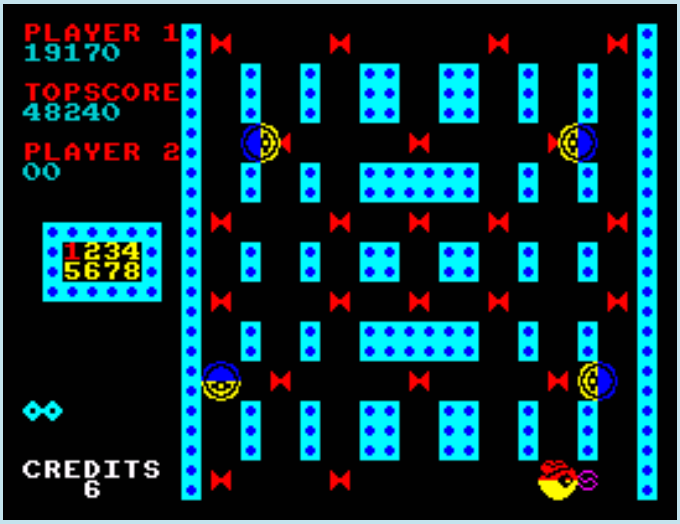
A Puzzle League sorozat első tagja, a Tetris Attack 1995-ben jelent meg SNES-re, ezt a játékot írta át a Boride 0.5 és ZX Basic Compiler 1.8.9 segítségével Spectrumra oblo, akinek könnyű felismerni a játékait a címükről: O-Cman, O-Trix és most O-Puzz Attack!!.

A játéktér hasonló a Tetrishez, a különbség annyi, hogy nem felülről potyognak tárgyak, hanem alulról folyamatosan emelkedik a színes négyzetek szintje. Az emelkedés sebessége a játék előtti nehézségi szint választásodtól függ, valamint a szerzett pontjaidtól, minden 1000 pont után kicsit gyorsabb lesz. A célod, hogy négyzetek cserélgetésével három vagy több egyező színűt juttass egymás mellé vízszintesen vagy függőlegesen, ilyenkor ezek eltűnnek, a helyükre pedig a felettük lévők potyognak, és természetesen pontokat kapsz. A 'csere' gomb megnyomásával a kurzorod által kijelölt két elem helyet cserél egymással. Ezt lehet két blokk cseréje, vagy lehet egy blokk és egy üres terület közötti helycsere is. A játéknak vége van, amikor a blokkok elérik a játéktér tetejét.



O-EYES

Szerző: **oblo**
HW igény: **48K**
Stílus: **arcade/labirintus/lövöldözős**
Vezérlés: **Billentyűzet (K, L, S, X, M - balra, jobbra, del, le, tűz) és Kempston, Sinclair Joystick**
Megjelenés: **2018. december, ingyenes**



Oblo hihetetlen sebességbe kapcsolt, második decemberi játékát jelentette meg! Az utolsó játékánál is használt Boride 0.5 és ZX Basic Compiler 1.8.9 segítségével portolta Spectrumra az Eyes című 1982-es pénzbedobóst, konkrétan még a pénz bedobását is...

Egy kalapot viselő szemgolyó vagy egy rád lövöldöző, számítógép által vezérelt szemeknek is otthont adó labirintusban, erre utal a cím is. Nem, nem a labirintusra, a szemekre. A célod az, hogy megsemmisítsd az összes pontot, vagy eldőlt homokórát, vagy ki tudja mit. Ha ez sikerül, akkor új szintre lépsz. A lényeg: kilőni kell, nem megenni, az egy másik játék. Szerencsés bolygóállások esetén, ha kilősz egy komputerizált szemet, majd nagyon gyorsan bekapod, azzal egy időre a labirintust szemtelenebbé teszed.

Játék előtt az '5' és '6' billentyűkkel tudsz pénzt bedobni a gépbe! Nem vicc, hidd csak el, dobáld tele, de csak óvatosan, mert kivenni nem lehet!

NOHZDYVE

Kiadó: **Tuckersoft**
Szerzők: **Matt Westcott, Clayton McDermott, Brian Reitzell**
HW igény: **48K**
Stílus: **arcade/akció**
Vezérlés: **Billentyűzet (O, P - balra, jobbra)**
Megjelenés: **2018. december, ingyenes**

A Netflix már-már kultikus sorozata a Black Mirror (az igazságosság kedvéért nem ők, hanem a Channel 4 indította a sorozatot), ami a Wikipediaról idézve: "egy brit antológia-sorozat, melynek alkotója Charlie Brooker. Különálló epizódjai az információs társadalom és a technológiai fejlődés veszélyeit mutatják be egy-egy disztópikus történetben".

Néhány napja még azt sem lehetett tudni, hogy a december 28-án megjelenő új rész, a Bandersnatch (ismerős a cím?) a sorozat ötödik évadának nyitó epizódja lesz, vagy esetleg valami más. Utóbbi lett, vagyis egy interaktív film a negyedik és ötödik évad közé bedugva. Az interaktivitás annyit jelent ezesetben, hogy a film időnként arra kéri nézőjét, hogy válasszon egyet a képernyőn megjelenő két lehetőség közül, amely befolyásolja a történetet, elvileg nagyon sok potenciális út van az öt különböző befejezésig. A filmben szerepel a Tuckersoft nevű cég, aminek a Netflix készített egy honlapot, ahol

szerepelnek a játékaik is, köztük a címadó Bandersnatch és egy letölthető játék, a Nohzdyne is (a filmben a cég sztárprogramozója írja BASIC-ben ezt a játékot). A film 1984-ben játszódik, szóval ez a játék a film világában 1983-1984-es lehet, ilyen szemszögből is lehet nézegetni.

A játék baromi egyszerű: kidobnak egy sikátor felett két elképesztően, végtelenül magas épület között, a célod elsősorban a túlélés, vagyis, hogy ne kenődj fel egyik ház falára sem. Találkozol klímákkal, kifeszített szárítókötelekkel, de ezeket ignorálhatod nyugodtan, két tárgycsoportra koncentrálj: az egyik a gigantikus, csattogó műfogsorok, ők harapnak, tehát nem szereted őket, a másik a kitépett szemgolyók, őket kedveled valamilyen perverz okból, így pontokat kapsz, ha felveszed őket. Lényegében ennyi, előbb-utóbb meghalsz, de a remény addig is éltessen!

WOOT! TAPE MAGAZINE ISSUE #2 - ZXMAS 2018 EDITION

Kiadó: **Stonechat Productions**
Szerzők: **Dave Hughes, djnzx**
HW igény: **48K/128K**
Stílus: **elektronikus magazin játékgyűjteménnyel**
Vezérlés: **Billentyűzet (O, P - balra, jobbra) + a játékoknál egyéni vezérlés**
Megjelenés: **2018. december, ingyenes**

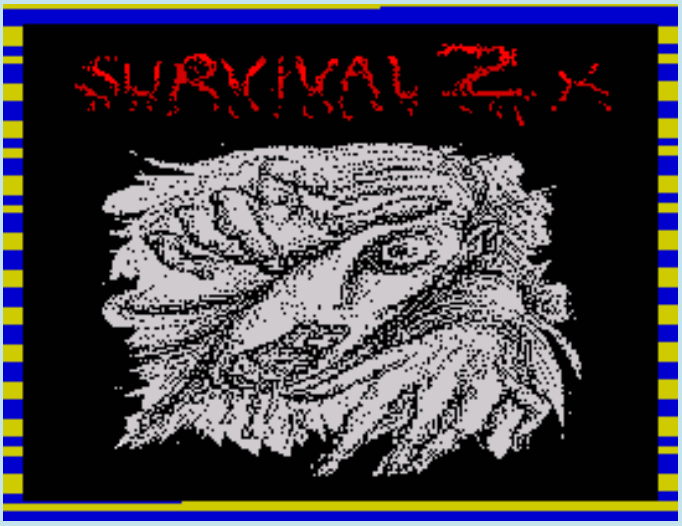
A WOOT! Tape Magazine már a harmadik számánál tart (az első a #0 volt), 2016 óta jelenik meg mindig Karácsonykor. A tartalom általában nagyon vicces hangvételű, de informatív is egyben. A mostani számban a játékokon kívül van köszöntő, játékajánlók, interjú a Gargoyle Games sztárjával, Cuchulainn-nal, ZX-Art művészetti tárlat, hamis hírek, biztonsági kérdés, screen-poénok, AGD történeti összefoglaló, AY zenei demo, no meg a játékmelléklet, néhány demóval és hat teljes játékkal.

MANIC SCROLLER

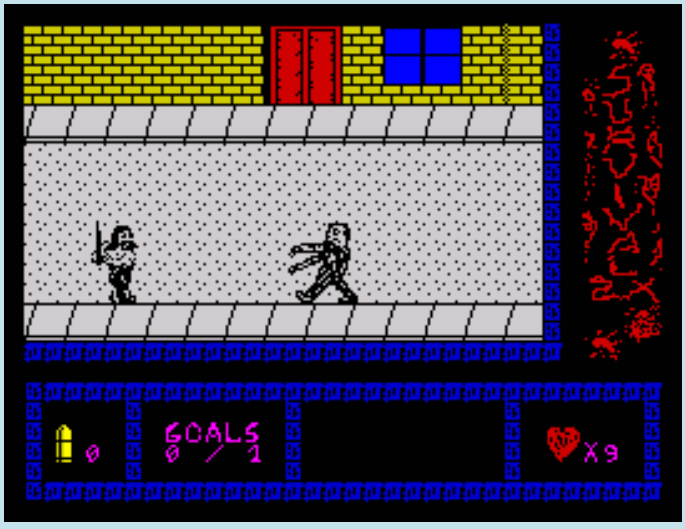
Szerzők: **Mark Woodmass, Daniel Gromann**
HW igény: **48K**
Stílus: **Manic Miner MOD**
Vezérlés: **Billentyűzet (Q, E, T, U, O - balra, W, R, Y, I, P - jobbra, CS..Sp - ugrás, CS+Sp - kilépés)**
Megjelenés: **2018. december, ingyenes**

Mark Woodmass 2008-ban készítette el a Manic Miner dupla szélességű, vízszintesen scrollozó változatát. Sajnos az a verzió hibás volt több helyen, amit Daniel Gromann (jetsetdanny) javított, majd adott ki tíz évvel az eredeti verziót követően.

SUVIVAL ZX



Szerző: **Mr Rancio**
HW igény: **128K**
Stílus: **akció/kaland**
Vezérlés: **Billentyűzet (O, P – balra, jobbra, Q, A – fel, le, M – tűz, Sp – cselekvés: keresés, ajtó nyitása, belépés, tárgy felvétel / használat, beszélgetés más túlélőkkel) és Sinclair Joystick**
Megjelenés: **2018. december, ingyenes**



Mr Rancio (Software) bemutatkozása a Spectrum világban egy tárgykeresős, felhasználós, zombihentelős akciójáték.

„Ismerem a háború borzalmait, de ez már túl sok számomra. Azok a regények, filmek, sorozatok, amelyek az úgynevezett "zombi apokalipszis"-nek nevezett dologról szólnak, meg sem közelítik azt, ami valójában történik. Elvesztettem a családomat és a barátaimat. Semmi sem maradt, amiért éljek. Az öngyilkosságra gondolok. Az ötlete gyakran átsuhan az agyamon. Ha találnék egy fegyvert… Egy lövés a fejembe megoldaná életem problémáit. Kis szerencsével nem lenne belőlem egy ilyen átkozott szörny. Az olyan napokon, mint ez is, élek, élni akarok. El kell hagynom ezt az épületet. Semmi sem biztonságos."
(Joe naplójából: 2025. május)

Joenak, a szinte az egész lakosságot kiirtó járvány egyik túlélőjének öt küldetést kell teljesítenie, hogy elérje a menedéket. Ehhez meg kell találnia és használnia a tárgyakat, valamint kapcsolatba lépnie néhány karakterrel, no meg elérnie az egyes fázisok végét. Néhány objektum el van rejtve (kukában,...), ezért keresni kell őket a cselekvés gombbal (space). A Z-vírussal fertőzött holtak emberi húsra éhesek... Döbbenet...

MISTER KUNG-FU

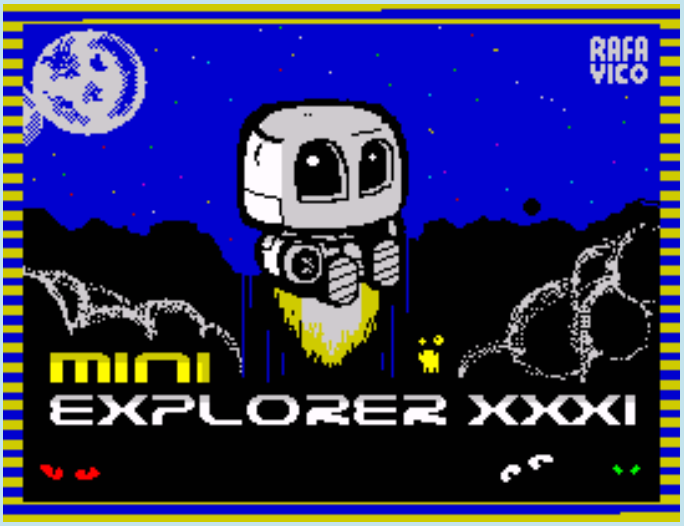
Kiadó: **Uprising games**
Szerzők: ***Elton Bird (kód, grafika, zene), Andrew McDonell (grafika), Tony Pastor (kiegészítő grafika), Stuart Campbell (borító)***
HW igény: **48K+AY**
Stílus: arcade/verekedős
Vezérlés: Billentyűzet: újradefiniálható (alapértelmezés: O, P – balra, jobbra, Q, A – ugrás, guggolás, Sp – ütés) és Kempston, Sinclair Joystick
Megjelenés: 2018. december, ingyenesen letölthető

Az Uprising Games bemutatkozása, első játéka ez a remake, amit a ZX-DEV M.I.A. - Remakes versenyre nevezett be Elton Bird. A játék eredetije az Irem 1984-ben kiadott árkádja, aminek született Spectrum átirata is, de az US Gold 1986-os verziója csalódást keltő, ezért próbált az Uprising csapata egy jobbat összedobni.

A sztori főszereplője Mister Kung-Fu (Master), alias Thomas, aki bepöccen, miután megtudja, hogy

barátnőjét, Silviát fogva tartják az Ördög Templomában tanyázó rosszak. Tomi boy úgy látja, hogy jó eséllyel felveheti a harcot egy egész sátánista karategalerival szemben, így nekilódul, hogy megmentse szerelmét, mit neki öt emeletnyi ördögtemplom, meg gazfickók...

MINI EXPLORER XXXI



Szerzők: **RafaVico (program, grafika), Sergio thEp0pE (zene), Igor Errazking (betöltőkép)**
HW igény: **48K+AY**
Stílus: **arcade/lövöldözős**
Vezérlés: **Billentyűzet (O, P – balra, jobbra, Q – fel, Sp – tűz) és Kempston, Sinclair Joystick**
Megjelenés: **2018. december, ingyenes**

RafaVico két AGD-s játékkal is nevezett a ZX-DEV M.I.A. - Remakes-re, elsőként a Mini Explorer XXXI-vel, ami a Dro Soft nevű spanyol kiadó 1988-as Explorer XXXI-jének az újratervezése.
A sztori a XXXI. századba vezet, mikor a Föld bolygó egy téridő-viharon megy keresztül, melynek következményei katasztrofálisak: az RZ-800, a Föld pályáját stabilizáló intertemporális részecske a múltba került. Az intertemporális részecske nélkül a Föld tehetetlenül rohan a Nap felé.
A Tanács téged jelölt ki, hogy hozd helyre a dolgokat.

ELON M. WITH JETPACK

Szerzők: **RafaVico (program, grafika), Sergio thEp0pE (zene), Igor Errazking (betöltőkép)**
HW igény: **48K+AY**
Stílus: **arcade/lövöldözős**
Vezérlés: **Billentyűzet (O, P – balra, jobbra, Q – fel, Sp – tűz) és Kempston, Sinclair Joystick**
Megjelenés: **2018. december, ingyenes**

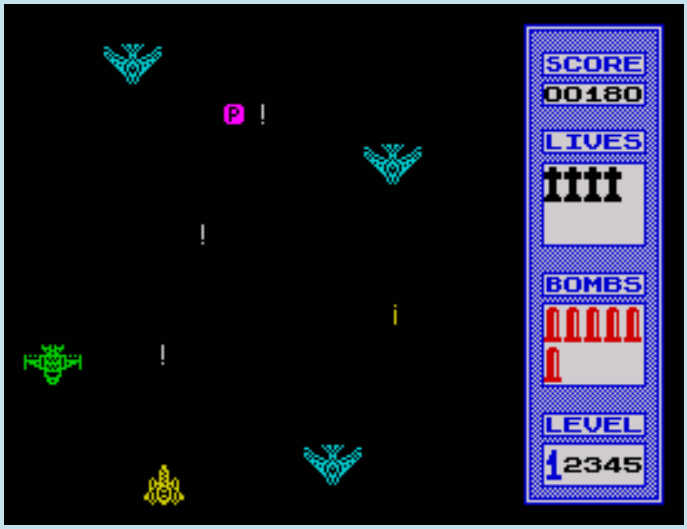
RafaVico a ZX-DEV M.I.A. - Remakes-re nevezett be egy remake-kel, melynek alanya az Ultimate Play the Game 1983-as klasszikusa, a Jetpac. Ez a szerző második játéka az AGD felhasználásával, amihez próbálta felhasználni másik játékának, a Mini Explorernek a mozgási rutinját. A Jetpac-hez hasonló a játék, de néhány szerinte javítható dologban RafaVico változtatott (pl. a lézerfegyver túlmelegedhet).

A játék hőse, Elon M(usk), aki a rakétaflottájának segítségével járja végig az összes terreformált, vagyis földszerűvé tett bolygóját, tehát a bolygók között utazgat, és ellenőrzi, hogy megfelelnek-e az élethez. A bolygókon össze kell gyűjtenie a megfelelő mennyiségű KLAX-ot, ami a rakéták üzemanyaga, és természetesen óvakodni kell a helyi létformáktól.

SARDONIC

Kiadó: **Penisoft**
Szerzők: **MadAxe (program, grafika), Fred (zene)**
HW igény: **48K+AY**

Stílus: **arcade/lövöldözős**
Vezérlés: **Billentyűzet (O, P – balra, jobbra, Q, A – fel, le, X – bomba, Sp – tűz)**
Megjelenés: **2019. január, ingyenesen letölthető**



MadAxe egy portugál programozó, akinek a Sardonic a bemutatkozása a Spectrum színtéren.

Lőjj le mindenkit, akit csak tudsz. A hírszerzés szerint az ellenség az utolsó, véges számú csapataival, öt hullámban támad. Négy különböző típusú repülőgépük van, plusz minden hullám végén egy nagyobb szerkezet.

A képernyőn rendszeres időközönként megjelennek extra erőforrások: L - élet, P - pont, B - bomba, F - tűzerő, ezek hasznosak lehetnek a célod elérése érdekében.

Mezei Róbert (m/zx)

NEXTBUILD IDE

Az előző cikkemben úgy zártam a soraimat, hogy az új Spectrum személyi számítógép, a ZX Spectrum NEXT már szinte itt van a sarkon. Bő félév múlva azt kell belátnom, hogy nagyot tévedtem. A NEXT project, bár hatalmas lépéseket tett meg a végkifejlet felé, a mai napig sem fejeződött be.
Érdekes módon a problémát nem maga a hardware, hanem a gép „kasztnija" okozza (amibe beleérthetjük magát a dobozt, a billentyűzetmembránt és a billentyűzet gombjait is). Ráadásul megint eltűnt jelzés nélkül egy beszállító cég, ami ezúttal is nagyot tudott lassítani a folyamatokon. A legfontosabb cél most a teljes, dobozos NEXT prototípusának legyártása és tesztelése. Ha ez megvan és sikeres, csak akkor indulhat a sorozatgyártás és az elkészült termék kiküldése a Kickstarter tagoknak. Ennek az időpontját találgatni teljesen értelmetlen, csak remélni lehet, hogy legalább karácsonyra minden lezárul. Ami biztos, hogy az alkotók nagyon keményen dolgoznak, és szívüket-lelküket beleteszik a projektbe. Aggódni ezért szerintem felesleges, a gép meglesz, csak ki kell várni valahogy.

A hardware mellett a gép firmware-je, illetve operációs rendszere és felhasználói kézikönyve intenzív fejlesztés alatt van. Több kisebb-nagyobb játék és egyéb szoftver is be lett jelentve, sőt néhány már el is készült és megvásárolható az interneten keresztül letöltés formájában vagy korlátozott példányszámban akár dobos termékként is.

NEXTBUILD, egy eszköz mind felett?

A számos, fejlesztés alatt álló projekt közül szeretném kiemelni a NEXTBUILD integrált fejlesztőrendszert, ami nem más, mint több fejlesztőeszköz, két emulátor és egy nagyszerű IDE (integrált fejlesztői környezet) együttese.

Miért izgalmas ez? Egy számítógépes platform szíve-lelke a hatékony és lehetőség szerint kényelmes fejlesztői eszközök megléte. A fejlesztők nem tudnak új programokat készíteni, ha ezek nem állnak rendelkezésre és programok nélkül a hardware maga halálra van ítélve. Minél jobbak ezek az eszközök, annál többen mernek belevágni új projektekbe (akár hobbistaként vagy amatőrként is), annál jobban pezseg a platform körül az élet. Az eszközök mellett persze fontos a jól dokumentált és elérhető fejlesztői könyvtárak/API-k megléte is (amin keresztül a programozó el tudja érni és vezérelni tudja a gép egyes részeit: a grafikát, hangot, háttértárat, stb.) Ezek még azért erősen képlékeny állapotban vannak jelenleg, hiszen maga a gép is fejlesztés alatt áll, de egyre több információ áll rendelkezésre a NEXT weboldalán és a különböző fórumokon, Illetve szerencsére maga a NEXT közösség is nagyon segítőkész.

A NEXTBUILD lényegében egy keretrendszer, ami lehetővé teszi azt, hogy magát a programozást, a kód lefordítását, tesztelését illetve kipróbálását egy programon belül el lehessen végezni. Integrált eszköz nélkül ezekre a fejlesztés során külön fordítót, szövegszerkesztőt és emulátort kell használni, de ezeknek a megtanulása időigényes lehet és lassíthatja is a fejlesztést. A rendszert kezdő és profi fejlesztőknek egyaránt ajánlják, és akár normál ZX Spectrum programokat is lehet majd benne fejleszteni, bár az ajánlott célplatform a NEXT.
Lássuk akkor, milyen komponensekből is áll az új rendszer:

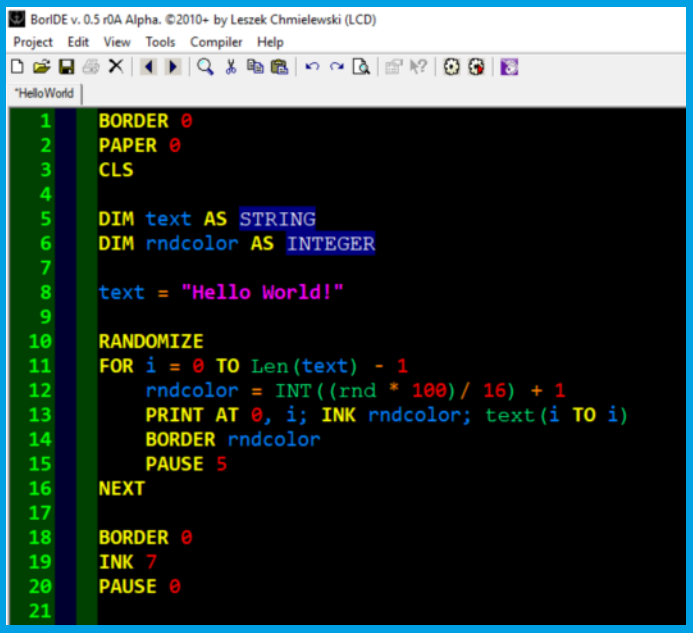
- BorIDE: Integrált fejlesztői környezet. A programkód kényelmes szerkesztését teszi lehetővé, „Syntax highlighting" támogatással.
- ZXB: Basic fordító (a rendszer „lelke") amivel assembly sorokat is tartalmazó programokat is fordíthatunk. A fordító a felhasználó által definiált könyvtárak használatát is megengedi.
- CSpect és FUSE: két nagyszerű emulátor. A CSpect ZX Spectrum Next-et, a FUSE normál Spectrumokat képes emulálni.
- NextLibs – ZX Spectum NEXT könyvtárak, amelyek segítségével a ZXB fordító el tudja érni a Next hardware-t és így látványos (a rendszer képességeit teljesen kihasználó) programokat tudunk készíteni. A könyvtár persze még szintén fejlesztés alatt áll, eddig az SD kártya filekezelő és a NEXT továbbfejlesztett grafikai képességeit kihasználó rutinokból készült el néhány (Layer2, sprite-ok).
- Maga a ZXB fordító is tartalmaz saját könyvtárakat (matematikai, grafikus, szövegkezelő rutinokat.)

Hello World!

Hogy lássuk, milyen egyszerű folyamat a programkészítés a NEXTBUILD rendszerben, készítsük el a már-már kötelező „Hello World" programot egy kicsit felturbósítva:

A Nextbuild rendszert elindítva nyomjuk meg a „New” gombot vagy válasszuk a Project->New menüpontot.

A megjelenő ablakon gépeljük be a következő rövid programot:



Az F11 gomb megnyomásával a Nextbuild lefordítja programunkat, elindítja a beállított emulátort (az alapbeállítás szerint a CSpect), majd betölti az emulátorba a programunkat, ami el is indul automatikusan.

A sima Spectrum programok esetében - amik nem használják ki a Next különleges képességeit - beállíthatjuk, hogy a CSPect helyett a FUSE emulátor fusson. Ehhez csak a „Launch Fuse” checkboxot kell kiválasztani a jobb oldalon megjelenő opciók közül.

Fontos megemlíteni, hogy jelenleg a CSpect emulátor csak .sna image fájlokat képes fogadni, tehát ilyen esetben a fordító mindig sna image-t fog készíteni. A többi választható output-típus (pl. tap, tzx, asm, vagy bin) beállítása ilyenkor nincs hatással a működésre.

A program futtatásának eredményeként, a „Hello World” feliratnak karakterenként és minden karakter esetében különböző színnel kell megjelennie, valahogy így:

Rövid magyarázat a programhoz:

Az első három soron beállítjuk a képernyő és a keret színét feketére, majd töröljük a képernyőt.

Az 5-6. soron két szám típusú változót deklarálunk. Az első magát a szöveget, a másik az aktuális betűszín számkódját tartalmazza majd a futás során.

A 8. soron beállítjuk a megjelenő szöveget a deklarált, „text” nevű és string típusú változónkba.

Utána a RANDOMIZE utasítással a Spectrum Basic véletlenszám-generátorát inicializáljuk, hogy minden egyes futtatásnál más és más színben jelenjenek meg a szöveg egyes betűi.

A 11. és 16. soron egy ciklust definiálunk. A ciklus magját (12-15. sorok) annyiszor fogja a program végrehajtani, ahány betű van a szövegünkben.

A ciklusmagon belül a véletlenszám-generátor segítségével meghatározzuk az aktuális betűszínt, majd a 11. soron ki is íratjuk azt. Az AT és az INK parancs segít, hogy az „i” ciklusváltozó által meghatározott sorba és oszlopba kerüljön a betű a megfelelő színnel.

A 14. soron beállítjuk a BORDER (keret) színét az aktuális színre (hogy legyen egy kis „effekt” a program futása során), a betű kiírtatása után várunk egy picit, majd jöhet a következő betű.

Ha teljes szöveg megvan, akkor a keretet feketére, a betű színét fehérre állítjuk, majd a PAUSE 0 paranccsal megállítjuk a programot egy gombnyomásra várva. Erre a CSpect emulátor miatt van szükség, mivel a PAUSE parancs nélkül a program futása után az emulátor azonnal reseteli a gépet.

A programablak alatt egy üzenetablak kapott helyett, amiben a fordító üzenetei (például az esetleges fordítási hibák szövege) jelennek meg. A hibák szövegét a program egy jegyzettömb ablakban is megmutatja nekünk. Ezt az ablakot zárjuk le még a következő indítás előtt, különben a fordítás hibára fog futni!

Amint látható, a programozás és a tesztelés nagyon egyszerű, a program gombnyomással indítható és az eredmény azonnal látható.

Fontos megjegyezni, hogy a NextBuild program még nem készült el teljesen, pár funkció még nincs elkészítve vagy éppen befejezve, és hibás működésre is számítanunk kell a használat során. Mindezek ellenére a Nextbuild már most is egy nagyon kényelmes és hasznos fejlesztőeszköz, amit nyugodt szívvel lehet ajánlani mindenkinek, aki a Spectrum Next programozására kedvet érez.

Több igen fejlett példaprogram is található a csomagban, érdemes közülük szemezgetni. Ezekből kettőt ajánlanék: az egyik a BigScroller, ami a demó programokban néha látható hatalmas szöveget „tekeri végig" a képernyőnkön nagyon simán és nagy sebességgel. A másik a Chicken, ami akár egy rendes mászkálós/platformer játék egyik helyszíne is lehetne.

Magához a ZXB fordítóhoz is sok példa van mellékelve, ezek a ZXBC\examples könyvtárban találhatóak.
Remélem sikerült kedvet csinálnom a NEXTBuild-hez. Én magam nagy izgalommal várom az újabb verziók megjelenését. Az alap, amit az alkotók leraktak, nagyon erős, de szerintem még hosszú út van hátra addig, amíg a NEXTBuild-ből igazi, kiforrott fejlesztőrendszer lesz. Sok múlik azon, hogy a NEXT-es közösség hogy fogadja a NEXTBUILD-et: lesznek-e elegendően, akik használják, illetve a fejlesztők kitartása is töretlen marad-e. Véleményem szerint erre minden esély megvan.

Befejezésül szeretném a teljesség igénye nélkül felsorolni, hogy kik vesznek részt a NEXTBUILD, illetve az egyes komponensek fejlesztésében:

BorIDE integrált fejlesztői környezet:
Leszek Chmielewski (LCD)

XZB Spectrum Basic fordító: Boriel

CSpect emulátor: Mike Dailly

Fuse emulátor: Philip Kendal és még sokan mások

NextLibs: David Saphier

Vándorffy Tamás (Vándor)

WANTED

Az alábbi magyar fejlesztésű ZX Spectrum programokat keressük z80/tap/tzx formátumban vagy akár kazettán, természetesen keressük a hozzávaló kazettaborítót, leírást magyar vagy akár angol nyelven is.

Játék programok:

- 3D Sakk (Novotrade)
- Alquerque (InterBit)
- Betűpóker (Novotrade)
- Betűrömi (Novotrade)
- Gazdasági játékok (16K) (InterBit)
- Memória (Király Péterné; Király-Török SW)
- Színkereszt (16K) (InterBit)
- Tervezhető CHAOS (Héra Tibor)
- Tologató játék (INTEGRÁL)
- Trilog kazettaborító (Novotrade-Andromeda)
- Űrjátékok (16K) (InterBit)

Oktatóprogramok:

- Erdélyi fejedelmek, Habsburg uralkodók (Ságí György, 1986, KLTE OK.)
- Hangbűvölő (oktató, InterBit)
- Teszt-Mester (oktató, InterBit)

Segédprogramok:

- BASIC compiler (InterBit)
- HUN-CHAR (Tasword) (INTEGRÁL)
- Monitor-Disassembler (InterBit)
- Spectmusic (Rozsnyai György)

Ügyviteli programok:

- Adófejtő INTEGRÁL
- Címletezés LSI ATSZ
- GMK jövedelemadózás LSI ATSZ
- Havidíjasok bérelszámolása LSI ATSZ
- Jövedelemadó-számítás és -nyilvántartás INTEGRÁL
- Keresetszint szerinti adózás LSI ATSZ
- SZJA'88
- Üszőnyilvántartás (INTEGRÁL)

A speccyalista@sinlair.hu email címen értesíthesz bennünket, ha neked megvan valamelyik. Természetesen, ha olyan magyar fejlesztésű program van a birtokodban, ami ebben a listában nem található, akkor se késlekedj:

KERESSÜK

PROGRAMOZÁSTECHNIKA - MÉLYVÍZ

AZ ULAPLUS SZABVÁNY A GYAKORLATBAN - 1.RÉSZ

Bevezető

Az ULA történetét bemutató cikkben elolvashatjuk az ULAplus szabvány célját és legfontosabb tervezési szempontjait. Ebben a cikkben az alkotni vágyó olvasók számára próbálom összefoglalni a legfontosabb tudnivalókat arról, hogy mit és hogyan lehet az ULAplus által nyújtott lehetőségekből kihasználni. A cikk a ZX Spectrumhoz járó gépkönyvben (bármelyik változat) leírtakon túl semmilyen plusz ismeretet nem feltételez, de a történeti áttekintésnél szükségszerűen jóval technikaibb. Minimális ZX Spectrum programozási rutin elengedhetetlen a cikk megértéséhez és a benne leírtak sikeres használatához.

Az ULAplus képességek elérésének módja első ránézésre meglehetősen rossz hatékonyságú megoldásnak tűnik, de valójában egy nagyon fontos, mindent felülíró szempont miatti kompromisszumról van szó: a legtöbb ismert ZX Spectrum perifériával kompatibilisnek kellett lennie. Sajnos Sinclair már a ZX Spectrum tervezésével megalapozta azt a gyakorlatot, hogy a különböző eszközök kimeneti-bemeneti címeinek dekódolása hiányos, ezért nem csak a nekik szóló címekre reagálnak, hanem másokra is. Rossz példával elöl járva, az eredeti ZX Spectrum összes alaplapi illesztésű perifériája (billentyűzet, hangszóró, magnetofon, keretszín) elvileg a 254-es porton érhető el, de gyakorlatilag csak az A0 címvezeték 0 állapota számít, így az összes páros portra reagál, oda már más nem kerülhet. Hasonlóan „takarékos” még sok másik periféria is, akár eredeti Sinclair, akár más gyártótól származik. Ez magyarázza, hogy az ULAplus csak két bemeneti-kimeneti portot használ (teljes címdekódolással), hogy össze ne akadjon más perifériával.

A paletta

A szabvány legfontosabb része a 64 színű paletta, amit 256 színből választhatunk ki. Ezt minden, a szabványt implementáló eszköz ugyanúgy csinálja, erre bátran számíthatunk, akár emulátoron, akár valamilyen hardware megvalósításban.

Egy adott paletta színt úgy tudunk beállítani, hogy a 48955-ös (hexadecimálisan BF3B) portra kiírjuk a szín sorszámát 0 és 63 között (egyéb értékeknek más a jelentése!), majd a 65339-es (hexadecimálisan FF3B) portra kiírjuk magát a kívánt a színt.

A szín kódolása a ZX Spectrum hagyományához híven fényességi sorrendben történik, `GRB` sorrendben, ám a Spectrumnál megszokott 1-1-1 bit helyett 3-3-2 biten: `GGGRRRBB`. Azaz 8 fokozatban állíthatjuk a zöld és a piros komponenseket, és 4 fokozatban a kék komponenst, illetve fekete-fehér üzemmódban 256 szürkeárnyalattal dolgozhatunk.

Tehát például a 20-as szín teljesen telített zöldre állítását a következő BASIC parancssor éri el:

```
OUT 48955,20 :
OUT 65339,BIN 11100000
```

Hogy a 65339-es port kiolvasásával vissza tudjuk olvasni a beállított színt, opcionális része a szabványnak, ugyanakkor az ULAplus detektálásának egyetlen lehetősége. Szerencsére szinte minden implementáció támogatja, ezért nem lövünk túl nagy bakot, ha számítunk rá.

Persze ha csak ezekkel próbálkozunk, semmi látványos nem fog történni, hiszen az, hogy a beállított palettát hogyan használja a videoáramkör, egyéb beállításoktól függ, s az alapbeállítás az, hogy „sehogy”.

Átszínezett alapmód

A leghasznosabb és minimálisan támogatandó videomód az ún. átszínezett alapmód. Ez az üzemmód az eredetivel teljesen azonos képfelépítést használ, de igen rugalmas módon megváltoztathatjuk a színkódok jelentését. Ezzel lehetőségünk nyílik már meglévő játékokat és egyéb programokat átszínezni, nagyon kevés munkával. Szintén minden ULAplus-t támogató emulátor és hardware tudja.

Mind a 64 szín egyidejűleg megjelenhet a képernyőn processzormunka nélkül, ám komoly megkötésekkel. A 64 szín 4 ún. CLUT (Color LookUp Table, színtáblázat) csoportot alkot, mindegyikben 16 színnel, amelyik közül az első 8 az előtér szín, a második 8 a háttérszín. Azt, hogy melyik CLUT-ot használjuk a 4 közül az attribútum byte felső két bitje (eredetileg BRIGHT és FLASH) dönti el. A 8 lehetséges keretszín a 8 és a 15 közötti színek, azaz a 0-ás CLUT háttérszínei.

Mit lehet ezzel elérni? Meglepően sokat. A legegyszerűbb nyilván a meglévő 15 színű paletta (7

alap és 7 fényes szín, plusz fekete) átszínezése 16 tetszőleges színre. Nyilván érdemes olyan színeket választani, amik azért nagyjából közel vannak a megfelelő eredeti színhez, hogy ULAplus nélkül is hasonlóan nézzen ki, amit csinálunk. Ezzel például el tudjuk érni, hogy a Commodore 64, sőt a Commodore 16/116/+4 nagyfelbontású képeit (legalábbis, amennyi a 256x192-es felbontásba belefér a 320x200-ból) eredeti színeikben élvezhessük. Már meglévő játékok átszínezésének is ez a legegyszerűbb módja.

De ennél többet is tehetünk, hiszen immár nem 16, hanem 64 színből gazdálkodhatunk. Az eredeti Spectrum két megkötésétől szabadulhatunk meg: az emelt fényesség egyszerre hatott az előtér- és háttérszínekre, illetve az előtér és a háttér színeket ugyanazok közül a színek közül választhattuk csak ki. Ha szerencsénk van, akkor bizonyos színeket előtér- és háttérszínként konzisztensen más-más célra használt az átszínezni kívánt program, így különböző színe lehet olyasminek, aminek eredetileg ugyanaz volt a színe. Ha csak ennyit teszünk, akár 32 színben pompázhatnak a régen legfeljebb 15-színű programok. A kódhoz még mindig nem nyúltunk.

Ha képesek és hajlandóak vagyunk mélyebben belenyúlni a szoftverbe, akkor az általában nem használt FLASH bit felhasználásával összesen 32 előtér- és 32 háttérszínre teszünk szert (persze csak az azonos CLUT-ban lévőket párosíthatjuk egy attribútumon belül). Ez akár vízszintesen, akár függőlegesen lehetővé teszi, hogy 8 pixelenként váltsunk előtér- vagy háttérszínt ismétlés nélkül, gyönyörű gradienseket varázsolva a képernyőre. Azonban ha ezt a lehetőséget is kihasználjuk, elveszítjük a lehetőségét, hogy ULAplus nélküli masinán is változtatás nélkül élvezhető legyen az alkotásunk, ezért vagy detektálnunk kell az ULAplus meglétét, vagy külön verziót kell kiadnunk ULAplus számára a szoftverünkből.

Az üzemmód be- és kikapcsolását a 48955-ös portra 64-es érték írása után a 65339-es port írásával érhetjük el. A 0. bit dönti el, hogy a Spectrum eredeti palettáját (0-ás érték), vagy a fent beállított palettát (1-es érték) használjuk-e a színek kiválasztásánál. Az 1. bit dönti el, hogy fekete-fehér (1-es érték) vagy színes (0-ás érték) képet akarunk látni. Az 1. bit értelmezése opcionális, a többi bit pedig kötelezően nulla.

Tehát a megszokott Spectrum üzemmódot a 0 értékkel, annak fekete-fehér változatát a 2 értékkel tudjuk bekapcsolni. Mindkét esetben a 64-színű paletta használaton kívül marad. Próbáljuk ki!

```
OUT 48955,64 : OUT 65339,2
```

Ez nem garantált, hogy működni fog, bár én még nem találkoztam olyan megvalósítással, ami ne támogatná a fekete-fehér üzemmódot. Visszakapcsolni a színeket az `OUT 65339,0` utasítással tudjuk.

Ha használni szeretnénk a palettát, akkor 1 (színes) vagy 3 (fekete-fehér) értéket kell a 65339-es portra küldeni, miután a 48955-ös portra 64-et írtunk.

Csapjunk bele! A következő programot futtatva az eredetivel majdnem azonos palettát állítunk be, ami jó kiindulópont:

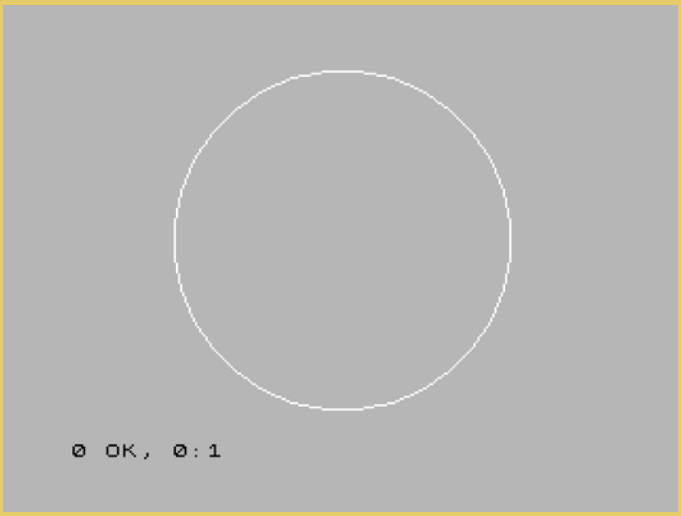
```
10 DATA BIN 00000000,BIN
00000010,BIN 00010100,BIN
00010110,BIN 10100000,BIN
10100010,BIN 10110100,BIN
10110110
20 DATA BIN 00000000,BIN
00000011,BIN 00011100,BIN
00011111,BIN 11100000,BIN
11100011,BIN 11111100,BIN
11111111
30 FOR b=0 TO 1: FOR f=0
TO 1
40 RESTORE 10*(b+1): LET
o=b*16+f*32: FOR i=o TO
o+7: OUT 48955,i: READ c:
OUT 65339,c: NEXT i
50 RESTORE 10*(f+1): FOR
i=o+8 TO o+15: OUT
48955,i: READ c: OUT
65339,c: NEXT i
60 NEXT f: NEXT b
70 OUT 48955,64: OUT
65339,1
```

Nézzük meg, mi is történt itt: a 10-es és a 20-as sorban felsoroljuk a normál illetve a fényes színeknek megfelelő 8-bites kódokat. A 30-as sorban a 4 CLUT-on végigiteráló ciklust indítunk, ami az eredetileg

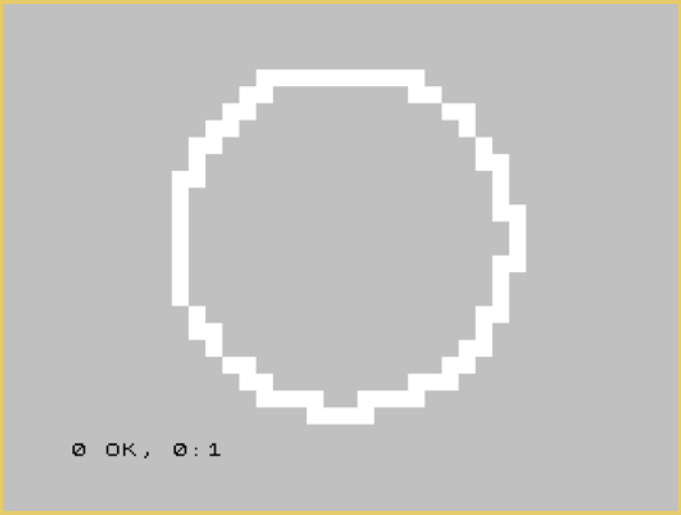
BRIGHT és FLASH attribútumbitek megfelelő értékeihez rendel színskálákat. A 30-as sorban az előtér, a 40 sorban pedig a háttérszíneket állítjuk be. Végül a 70-es sorban bekapcsoljuk a paletta használatát.

Első ránézésre nem sok különbséget veszünk észre azon kívül, hogy az eredetileg villogó kurzor már nem villog, viszont fényes. Ennek az az oka, hogy a FLASH bitet a háttérszín fényességére használtuk fel, míg a BRIGHT bit csak az előtérszín fényességét érinti. Most csináljunk valamit, ami ULA plus nélkül nem lehetséges:

```
CIRCLE INK 7; BRIGHT 1;
128,87,80
```



Azt látjuk, hogy a normál fehér (szürke) háttérre, fényes fehérrel rajzoltunk egy kört. ULAplus nélkül ez sokkal csúnyábban nézne ki. Az `OUT 65339,0` utasítással megnézhetjük, hogyan.



Ha csak annyit akarunk, hogy a Spectrum eredeti 16 színe helyett másik 16-ot használunk, akkor a fenti programban az 50-es sor `RESTORE` utasításában az `f` változót cseréljük ki `b`-re (esetleg `f<>b`-re, ha a villogást háttér-fényességváltozással akarjuk helyettesíteni), s a 10-es és 20-as sorok `DATA` utasításait követő színekkel játsszunk!

A fenti programot a megfelelő színekkel egy betöltő utasítás hozzáadásával bármelyik Spectrum program elé másolva könnyen elkészíthetjük annak átszínezett változatát.

Nézzünk egy másik példát az ULAplus paletta és az átszínezett alapmód használatára, immár assembly nyelven! Először is, vegyük észre, hogy a két port alsó 8 bitje ugyanaz, ezért ha az `OUT (C),x` utasításokat használjuk (melegen ajánlott), akkor elég a B regisztert állítgatni.

A következő példában lángra fogjuk lobbantani a képernyőt! Összesen 8 hőmérsékletértéket fogunk használni, hogy ULAplus nélküli Spectrumon is valamennyire élvezhető legyen, amit csinálunk. De természetesen ULAplus-on tüzes színeket akarunk látni. A BRIGHT és FLASH biteket most nem használjuk, tehát elég a 0-ás CLUT-ot feltölteni és feketére állítani a keretet:

```
   ORG  #8000
START:
   LD   HL, PAL      ; PAL a paletta címe.
   LD   BC, #BF3B
   LD   A, 64
   OUT  (C), A
   LD   B, #FF
   LD   A, 1
   OUT  (C), A       ; Bekapcsoljuk a színes
                       palettát.
   XOR  A
   OUT  (#FE), A     ; Kifeketítjük a keretet.
SETUPL:
   LD   B,#BF
   OUT  (C), A       ; Kiválasztjuk a megfelelő
                       színt,
   LD   B, #FF
   LD   D, (HL)      ; kiolvassuk a memóriából,
   INC  HL
   OUT  (C), D       ; majd beállítjuk.
   INC  A
   CP   16           ; Elég a nullás CLUT 16
                       színe
   JR   C, SETUPL
```

Eddig a paletta beállítása. Ha ennek a végére RET utasítást írunk, akkor a kód beállítja a palettát (már ha van ULAplus) és visszatér. Próbáljuk ki! Szükségünk lesz még magára a tüzes palettára (ezt mindig hagyjuk a kód végén):

```
..
PAL:
   ; 8 előtérszín
   DEFB  #00, #08, #10, #38
   DEFB  #7C, #BC, #FC, #FE

   ; Ugyanaz a 8 háttérszín
   DEFB  #00, #08, #10, #38
   DEFB  #7C, #BC, #FC, #FE
..
```

Ugyan a háttér- és előtér színek azonosak, ennek a redundanciának a kihasználása programkódból többe kerül, mint a 8 bájt, amit megspórolnánk, úgyhogy nem erőlködünk. Töröljük ki a RET-et és folytassuk!

Mivel véletlenszerű, hosszú ideig nem ismétlődő lángot szeretnénk, szükségünk lesz egy hosszú periódusidejű véletlengenerátorra. A beépített RND rövid periódusideje és lassúsága miatt alkalmatlan.
Használjuk tehát az RC4 véletlengenerátort. Ennek inicializálásához egy 256 bájt hosszú tömböt fel kell töltenünk a bájt minden lehetséges értékével:

```
..
  LD  HL, RND
INITR:
  LD  (HL), L
  INC L
  JR  NZ, INITR
..
```

Az RND tömböt 256-tal osztható címre kell rakni, ld. később. Csak az attribútum bájtokat fogjuk állítgatni, ezért a videómemóriát fel kell tölteni úgy, hogy minden attribútum egy háttér és egy előtérszínű 4x8 pixeles téglalapot tartalmazzon:

```
..
  LD  HL, #4000
  LD  DE, #4001
  LD  BC, #17FF
  LD  (HL), #0F
  LDIR
..
```

Magát a lángolást végeselem-módszerrel fogjuk szimulálni. Minden térrésznek egy bájt felel meg, amiben az ott lévő gáz hőmérsékletét tároljuk 0 és 63 között. A gáz alját az RC4 véletlengenerátorral „fűtjük”.

```
..
BURN:
  ; 24 sorban soronként 64 véges elem
  LD  DE, HEAT+64*24
  LD  H, RND / #100
BURNL:
   LD  L, E          ; Végrehajtjuk az RC4
                       algoritmust
   INC L
   LD  A, (RC4J)
   LD  B, (HL)
   ADD A, B
   LD  (RC4J), A
   LD  L, A
   LD  C, (HL)
   LD  (HL), B
   LD  L, E
   INC L
   LD  (HL), C
   LD  A, B
   ADD C
   AND #3F           ; A max. hőmérséklet 63
   LD  (DE), A       ; Beírjuk a hőtérkép
                       aljára a véletlent
   INC E
   JR  NZ,BURNL
..
```

Minden elem hőmérséklete az alatta lévő három és a kettővel alatta lévő egy elem hőmérsékletének az átlaga lesz. Ilyen módon mind a meleg felfelé konvektálását, mind a kondukcióval és radiációval kiegyenlítődő hőmérsékletet figyelembe vettük. Hogy ne kelljen kétszer végigiterálni az összes elemen, a szimulációval párhuzamosan táblázatból kinézve a megfelelő hőmérséklethez tartozó színt beállítjuk az attribútumértékeket is.

```
..
   LD  HL, #5800     ; Az attribútumtartomány
                       eleje
   ; Hőmérséklet - háttérszín táblázat
   LD  D, QUANTP/#100
   ; A hőtérkép második sorának eleje
   LD  IX, HEAT+64
   LD  BC, 3         ; Az attribútummező
                       3x256 elemű
FLI:
   LD  A, (IX+0)     ; aktuális elem
                       hőmérséklete
   ADD A, (IX-1)     ; + bal szomszéd
                       hőmérséklete
   ADD A, (IX+1)     ; + jobb szomszéd
                       hőmérséklete
   ADD A, (IX+64)    ; + alsó szomszéd
                       hőmérséklete
   RRA               ; /2
   SRL A             ; /2
   SBC A, 0          ; hűtés, hogy ne csapjon
                       túl magasra a láng
   ADC A, 0          ; alulcsordulás
   LD  (IX-64), A    ; a felső szomszéd
                       hőmérsékletének beírása
   LD  E, A
   LD  A, (DE)       ; háttérszín kiolvasása
   DEC D
   EX  AF, AF'       ; és elmentése
```

```
  INC   IX
  LD    A, (IX+0)      ; a fenti operáció a
                         következő elemmel
  ADD   A, (IX-1)
  ADD   A, (IX+1)
  ADD   A, (IX+64)
  RRA
  SRL   A
  SBC   A, 0
  ADC   A, 0
  LD    (IX-64), A
  LD    E, A
  EX    AF, AF'        ; elmentett háttérszín
  EX    DE, HL
  OR    (HL)           ; hozzákeverjük az
                         előtérszínt
  EX    DE, HL
  INC   D
  LD    (HL), A        ; attribútumérték
                         beállítása
  INC   IX
  INC   L
  DJNZ  FLI            ; 256 attribútumértékre
                         elvégezzük
  INC   H
  DEC   C
  JR    NZ, FLI        ; háromszor megismételjük
                         a 256-ot.
  HALT                 ; megvárjuk a képváltást
..
```

Nos, ezzel a lényeg már meg is van. Az egészet berakjuk egy végtelen ciklusba, amiből billentyűleütéssel lehet kijönni:

```
..
  XOR   A              ; a teljes billentyűzet
                         lekérdezése
  IN    A, (#FE)
  OR    #E0            ; magnóbemenet és egyebek
                         maszkolása
  INC   A              ; Ha minden bit egyes,
  JR    Z, BURN        ; jöhet a következő
                         filmkocka.
..
```

Végül helyreállítjuk a videó üzemmódot és kilépünk. A kód végére tesszük a táblázatokat és a munkaterületeket:

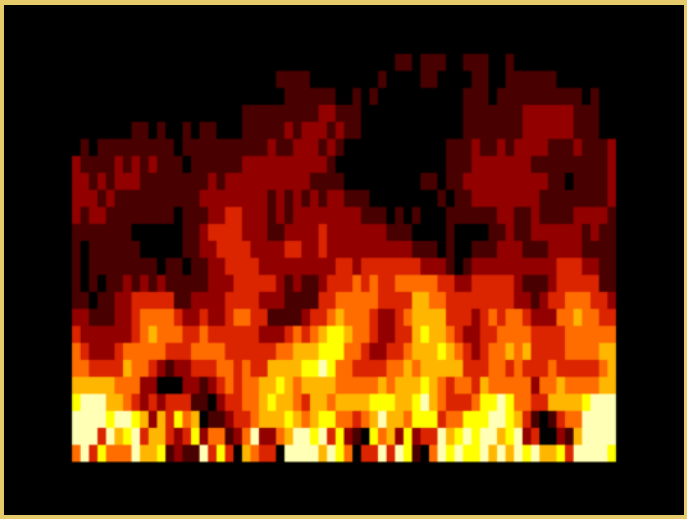
```
..
   LD   BC, #BF3B
   LD   A, 64
   OUT  (C), A
   LD   B, #FF
   XOR  A
   OUT  (C), A
   RET

; A nulladik CLUT
PAL:
   DEFB    #00, #08, #10, #38
   DEFB    #7C, #BC, #FC, #FE
   DEFB    #00, #08, #10, #38
   DEFB    #7C, #BC, #FC, #FE

; RC4 j érték
RC4J:
   DEFB    0

; 256-tal osztható címre igazítunk
   DEFS    (($ + #FF) / #100) * #100 - $

; Hőmérséklet-előtérszín táblázat
QUANT:
   DEFB    0, 0, 0, 0, 0, 0, 0, 0
   DEFB    0, 0, 0, 0, 0, 0, 0, 0
   DEFB    0, 0, 1, 1, 1, 2, 2, 2
   DEFB    3, 3, 3, 4, 4, 4, 5, 5
   DEFB    5, 6, 6, 6, 7, 7, 7, 7
   DEFB    7, 7, 7, 7, 7, 7, 7, 7
   DEFB    7, 7, 7, 7, 7, 7, 7, 7
   DEFB    7, 7, 7, 7, 7, 7, 7, 7
   DEFS    192
QUANTP:
   DEFB     0, 0, 0, 0, 0, 0, 0, 0
   DEFB     0, 0, 0, 0, 0, 0, 0, 0
   DEFB     0, 0, 8, 8, 8,16,16,16
   DEFB    24,24,24,32,32,32,40,40
   DEFB    40,48,48,48,56,56,56,56
   DEFB    56,56,56,56,56,56,56,56
   DEFB    56,56,56,56,56,56,56,56
   DEFB    56,56,56,56,56,56,56,56
   DEFS    192
RND:
HEAT:
EQU    RND + #100
..
```



A cikksorozat következő részében további videomódokat fogunk áttekinteni, és természetesen azokban is felgyújtjuk a képernyőt!

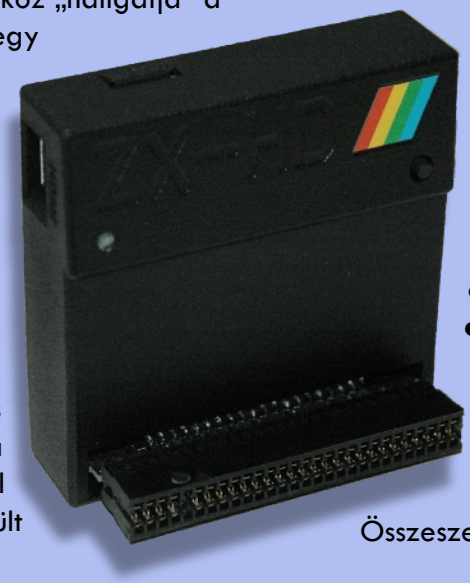
Nagy Dániel

HARDVER SIMOGATÓ

ZX-HD

Nemrégiben sikerült szert tennem egy Bytedelight ZX-HD-re, így gondoltam rögvest megosztom a tapasztalataimat vele kapcsolatban. Segítségével végre az asztalomon pöffeszkedő ZX Spectrumomat ráköthetem a 24 colos monitoromra, ráadásul a HDMI bemenetén keresztül, a korábbi kompozit-VGA konvertert kiváltva.

Nagyon leegyszerűsítve, ez az eszköz „hallgatja” a ZX Spectrum portokat, és egy Raspberry Pi Zero segítségével a ZX Spectrum képernyőjét megjeleníti a HDMI kimenetén. Az eredmény gyönyörű, éles pixelek, egyenesen a jó öreg gumigombosból, akár egy PC-s vagy Mac-es emulátorban. Ez már önmagában is felemelő, hát még, hogy ULAplus támogatással is rendelkezik. Többek között ez is szempont volt számomra a választásnál. Postaköltséggel együtt, 96 euróért sikerült beszereznem.



Az illesztő élesztése meglehetősen egyszerűen ment, eltekintve attól az apró malőrtől, hogy első próbálkozáskor elfelejtettem betenni a Raspberry Pi Zero-ba a hozzákapott SD kártyát, amin a firmware-e található. :) Egy aránylag régebbi 24”-os ASUS monitorral barátkoztattam össze, a kép elképesztően szép, tűéles és tökéletesen hozza a Spectrum színeit.
Jár hozzá egy 3D nyomtatott dobozka is, és bár én eddig nem komáltam ezeket a nyomizott dobozokat, de ez elég pofás, igényes darab.

Jellemzők:

- HDMI kimenet.
- Nincs szükség a ZX Spectrum módosítására.
- Akár 64 színű játékok a ZX Spectrum-on ULAplus támogatással.
- ZX Spectrum 128 videó memória támogatás.
- Pontosan szinkronizálva az ULA időzítéshez – a BORDER és multicolour effektek mennek, ahogy kell!
- A színeket alaposan bekalibrálták, hogy az egyedülálló Spectrum érzés ne szenvedjen csorbát!
- Teljesen kompatibilis az összes Spectrum modellel: ZX Spectrum 16K, 48K, 48K+, 128K, +2, +2A, +2B, +3, a Harlequinnel és más klónokkal is.

A csomag

Az illesztő „csináld magad” készletben érkezik, egy kis LE GO Spectrum geek-eknek. :)

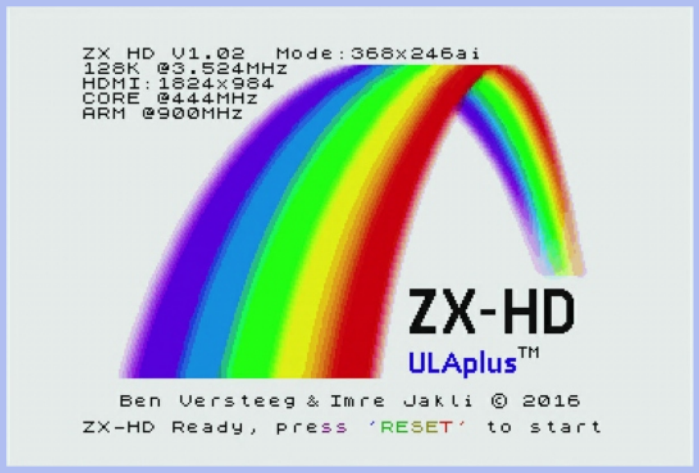
Ezt leljük a dobozban:

- ZX-HD.
- 3D nyomtatott doboz,
- mikro SD memória kártya, mikro SD - SD adapterrel,
- mini-HDMI - HDMI kábel,
- kézikönyv,
- opcionálisan Raspberry Pi Zero (én ezt is kértem, enélkül úgysem megy).

… és már csak egy kis csavarhúzóra van szükségünk.
Összeszerelési útmutató a Youtube-on.

Működés

Mindig kapcsoljuk be először a TV-t vagy monitort, mert a ZX-HD a HDMI kimenet felbontását a TV-vel fogja egyeztetni. Ezután bekapcsolhatjuk a ZX Spectrumot, amely néhány másodperc múlva az alábbi képernyővel jelentkezik be:

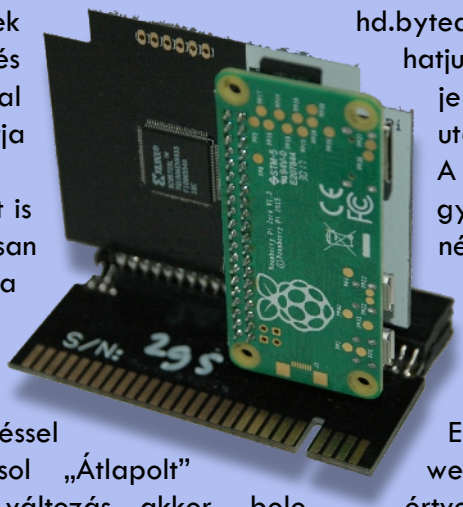


A képernyő felső részén a ZX Spectrum automatikusan felismert sebessége látható, amihez a ZX-HD kalibrálta magát, és a Raspberry Pi sebességét.
A „Mode:“ a kimeneti felbontást és az üzemmódot mutatja:

- **ai**: automatikus átlapolt mód (alapértelmezett)
- **fi**: átlapolt mód kikényszerítése
- **fp**: átlapolt mód letiltása

Ezek a beállítások a cmdline.txt fájlban módosíthatók. Alul látható, hogy a RESET gomb megnyomásával léphetünk tovább a Spectrum indításához. A ZX-HD elején található egy reset.

Átlapolt (Interlaced) videó mód



A ZX Spectrum 128-cas modellek ('toastrack', szürke +2, fekete +2A és +3) két videó memória lappal rendelkeznek. A ZX-HD támogatja ezen lapok használatát.
A ZX-HD még azokat a szoftvereket is támogatja, amelyek nagyon gyorsan váltják át ezeket a videó lapokat a két képernyőlap úgynevezett „átlapolás"-ához. Amikor a ZX-HD azt észleli, hogy ezek a lapok 4 egymást követő képernyőfrissítéssel változnak, automatikusan átkapcsol „Átlapolt" módra. Ha ezen idő alatt nincs változás, akkor visszatér a normál üzemmódba.
Az „átlapolt" módban a megjelenített sorok száma megduplázódik, így a HDMI framebuffer felbontása megduplázódik, és a páratlan vagy a páros képernyővonalak egymás után frissülnek. Ennek eredményeként a két különböző képernyőoldal egyidejűleg jelenik meg.
A ZX-HD a firmware-t tartalmazó mikro SD kártyáján a cmdline.txt fájlban az automatikus átlapolt mód felülbírálható. További információk ebben a fájlban találhatók.

A HDMI kimenet konfigurálása

Lehetőség van a ZX-HD konfigurációjának módosítására a ZX-HD microSD kártyáján található konfigurációs fájlok szerkesztésével.
A HDMI színkimenetet a felhasználó szabadon beállíthatja az igényei szerint. Az élénk színkészlet az alapértelmezett, emellett választható a ZX Spectrum kompozit videó kimenetéhez kalibrált készlet, vagy egyéni színkészlet készítésével is próbálkozhatunk. További információ a cmdline.txt fájlban található.
A config.txt fájl módosítható ezekre a beállításokra:

- megpróbálhatjuk a kijelző méreteihez igazítani a képet, ehhez megadhatjuk a TV méretarányát, pl. 16:9
- a megjelenített kép kis fekete szegélyének eltávolítása (ez azonban olyan pixeleket eredményezhet, amelyeknek nem feltétlen lesz azonos szélessége vagy magassága)

Firmware

A ZX-HD kétféle firmware-t tartalmaz:

- A CPLD chip a ZX-HD áramköri lapon
- A mikroSD kártyán lévő firmware a Raspberry Pi Zero számára

A Raspberry Pi Zero aktuálisan használt firmware verziója megjelenik ZXHD boot képernyőjén. A zx-hd.bytedelight.com webhelyen megtalálhatjuk a legújabb firmware verziót - ami jelenleg v1.2 -, beleértve a frissítési utasításokat is.
A CPLD firmware nem változik annyira gyorsan és CPLD programozó eszköz nélkül egyébként sem lehet frissíteni, ez jelenleg v1.3. A ByteDelight amúgy CPLD firmware frissítési szolgáltatást is kínál, ha lenne a jövőben CPLD firmware frissítés. Erről szintén a zx-hd.bytedelight.com webhelyen tudunk tájékozódni, beleértve a CPLD firmware frissítési szolgáltatással kapcsolatos további információkról.

Ismert hibák

- A ZX-HD 99,99%-ban szinkronizálva van a ZX Spectrum ULA-hoz, de egy nagyon gyors border effekt esetleg kibillentheti a szinkronból.
- Habár az ULAplus™ állapotregiszter olvasható, de jelenleg nem lehet kiolvasni az ULAplus paletta értékeit. Egyelőre nincs is olyan szoftver, amely ezt a szolgáltatást használná (az ULAplus-t használó játékok csak a színpalettát állítják be), azért a jövőben, később még integrálják ezt a funkciót.
- The ZX-HD ULAplus implementációja jelenleg még nem támogatja a Timex módokat, valamint HAM256 és HAM8x1 módokat (amik amúgy opcionális ULAplus jellemzők).
- A ZX-HD mutat némi inkompatibilitást az Interface 1-gyel, ezt jelenleg vizsgálják.

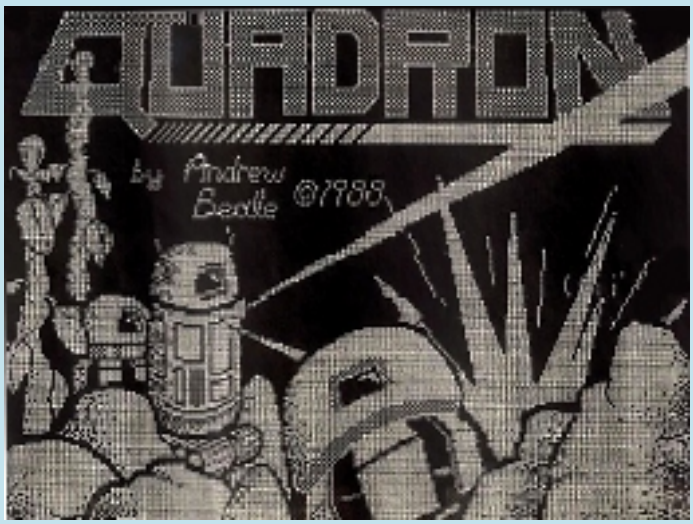
Kardos Balázs (Balee)

BYTEDELIGHT
Everything for your ZX Spectrum!

QUADRON

A Quadron egy 2018-ban megjelenő akciójáték, de a megjelenési éve ellenére mégsem mondható teljesen újnak.

Andy Beale (művésznevén Cosmium) 1986 és 1988 között azzal a céllal írta a játék eredeti változatát, hogy azt egy kiadó az akkori trendnek megfelelően kazettán, esetleg lemezen is kiadja, ő meg ha nem is meggazdagszik belőle, de a kétéves munkájáért kap némi anyagi ellenszolgáltatást. Inspirációként a korai Williams árkádokat jelöli meg, a Defendert, A Sinistart, a Robotron 2084-et. A Quadron igazán egyikre sem hasonlít, lehet, hogy csak ihletet adtak. Andy nem volt kezdő programozó, már 1983-ban több játéka megjelent a Softeknél (Megapede, Robon, Repulsar), majd 1984-ben az utód Softek International-nél (Microbot), és a szintén Softek érdekeltségű The Edge-nél (Psytraxx). Valószínűleg a



Quadron is vagy a The Edge, vagy az ACE Software (szintén Softek címke) neve alatt lett volna kiadva, lett volna, mert a kiadás nem jött össze, sőt még az eredeti forráskód is elveszett, mindössze a tervek, jegyzetek maradtak meg, melyekből Andynek újra kellett alkotnia játékát. Szerencsére sok minden megmaradt, Andy nem lehet egy kihajigálós fajta, még az eredeti betöltőképet is megtalálta, amit anno egy mátrix printerrel nyomtatott ki.

A lényeg, hogy az eredeti játék befejezése után 30 évvel szép lassan összeállt az új verzió, ami még javításokat is tartalmaz az őshöz képest, így idén piacra kerülhetett végre a Quadron, de nem ingyenesen, ahogy manapság szoktak a játékok, hanem egy cent híján öt dollárért, ami ha nem is hatalmas összeg, de kissé indokolatlannak tűnik egy eddig semmit sem bizonyító szerző teljesen ismeretlen játékáért. Szerzőnk lehet, hogy úgy volt vele, hogy már 30 éve is pénzt szeretett volna művéből csinálni, most ráadásul még dolgozott vele öt hónapot, így még inkább jár neki némi ellenszolgáltatás. A lényeg, hogy vajon megéri-e nekünk, játékosoknak durván 1500 Ft-ot áldozni egy játékra, mikor van helyette rengeteg ingyen? Határozott választ ez esetben nem fogok adni a saját kérdésemre, ugyanis a játék piszok nehézre sikerült. Sok, a játékban felmerülő helyzetre sem tudom a megoldást, így a következő leírásban akadhatnak még felfedezésre váró lyukak.

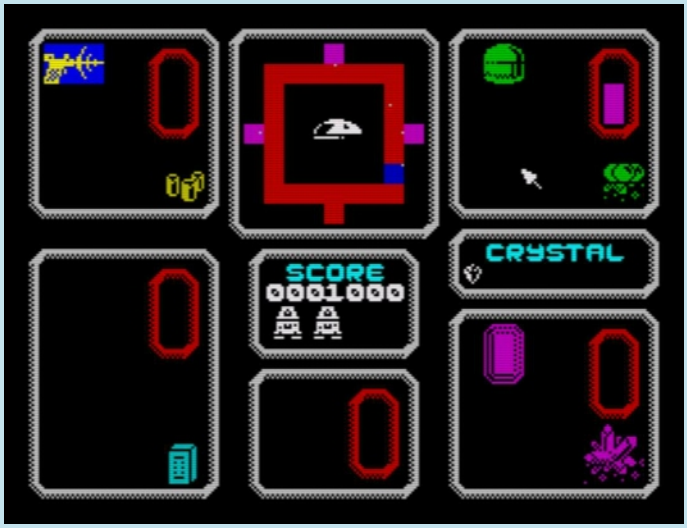
A játékban egy kis robotot vezérelsz, azzal a nem titkolt céllal, hogy felszabadítod az idegen támadók által megszállt Quadron bázist. A bázis huszonnyolc képernyőnyi területen terül el. A területet egy 7x7 képernyős négyzetet formáló folyosó, plusz mind a négy oldalon található egy átalakító helyiség. Ezek az átalakítók alapesetben egy roppant erős mezővel le vannak zárva, a nyitásukhoz kristályokat kell találnod. Az idegenek Quadron elleni offenzívája pont ezekért a helyi kristályokért van. A kristályokat a Quadron négy sarkában lévő fura sziklák termelik. Kezdetben a robotod képességei nagyon korlátozottak, csak egy rövid hatótávolságú lézerrel

van felszerelve, ami csekély kárt tud okozni az idegenekben. Azonban a játék folyamán a minimalista kis droidod olyan tárgyakkal fog találkozni, melyek növelik a tűzerejét, védelmet nyújtanak neki, transzportálják, vagy hatással vannak ellenségeid működésére. Az objektumok kategóriákba sorolhatóak, használatuk bizonyos mennyiségű kategóriájuknak megfelelő energiát igényel. Az objektumok nem használhatók e szükséges energia megléte nélkül. Ez az energia a Quadronon belül megtalálható, és átalakítható egyik kategóriának megfelelőről a másik kategóriának megfelelőre az energia átalakító segítségével az állapotképernyőn, mely átalakításhoz szintén energia szükségeltetik. Bonyolultan hangzik és nem véletlen, mert az is.

Robotod kezdetben védelmi rendszer nélkül lép a Quadronba. Az idegenekkel való ütközés és a tőlük beszedett lövések pedig gyorsan károsítják, amit a színe változása is mutat, fehér a maximális kondíciójában, sötétkék, ha mindjárt életet veszít, tehát pusztulásának mértéke Spectrum színkódos. Szerencsére kis robotodnak van egy regenerációs képessége, ha sérülést szenved, de még időben el tud vonulni a további károsodások elől, akkor öngyógyító képességét bevetve újra fehérre, tehát maximális egészségig javítja az állapotát, a gyógyulási folyamat annál tovább tart, minél nagyobb a károsodás mértéke. Három élettel kezded a játékot.

Lila a robot, tehát ideje lenne regenerálódnia, de mivel két idegen molesztálja, így valószínűsíthető a közeli életvesztés.

Az állapotképernyő a játék központi eleme, ami a Space gomb megnyomásával bármelyik pillanatban előhozható. Amíg az állapotképernyőt látod, a robotod nem sérülhet, nem történik semmi, mintha mindent kimerevítettél volna a Quadron területén. A képernyőn látsz egy egérkurzort, amit az iránybillentyűkkel tudsz mozgatni, a tűz gombbal „klikkelve” pedig bizonyos dolgokat tudsz aktiválni. A játékba a Space ismételt megnyomásával tudsz visszatérni.



A játék közben sokszor meg kell látogatni az állapotképernyőt.

Az állapotképernyő elemei:

TÉRKÉP: a képernyőn középen felül látható a Quadron négy folyosója és négy átalakító szobája lekicsinyítve, ahol kék háttérrel látod a saját pozíciódat, pirossal vannak jelölve a folyosók és a védett átalakító szobák, lilával a nyitott átalakító szobák. A felvillanó fehér kis piszkok pedig az ellenségeket és a Quadron négy sarkában megtalálható kőzeteket jelölik, a térkép középpontjában egy azonosító képen még azt is látod, hogy a felvillanó pötty mit takar a valóságban. A szkenner központi kijelzője körülbelül másodpercenként vált egyik típusú idegen/objektum azonosításáról a másikra, ezt a váltást meg tudod sürgetni az azonosító képre való klikkeléssel. Ha a tűz gombot nyomva tartod, akkor pedig ki tudod merevíteni az aktuális azonosítást, amint felengeded a tűz gombot, a folyamat automatikusan folytatódik.

PONT: a térkép alatt látod a pontszámodat.

ÉLETEK: a pontszám alatt, alapesetben robotok jelzik az életek számát, ha több mint három életed van, akkor azt számmal fogod látni.

KRISTÁLYOK: a pontszámod mellett látod, hogy mennyi kristály van aktuálisan a robotodnál. Maximum nyolc lehet, annál többet nem tud szállítani.

ESZKÖZÖK: a négy sarokban, valamint az életek alatt láthatod őket kategóriánként szétválasztva egy-egy fehér mezőben. A mező bal oldalán egymás alatt a megszerzett eszközöket látod, fekete háttérrel a még nem aktiváltakat, kék háttérrel az aktiváltakat. Aktiválni, illetve deaktiválni az eszközre való klikkeléssel tudsz. Jobb oldalon, egy piros mezőben az eszköztípushoz tartozó energiaszintet látod. A FEGYVEREK vannak a bal felső, a VÉDELMI ESZKÖZÖK a jobb felső, a TÁVIRÁNYÍTÓK a bal alsó, a TRANSZPORTEREK a jobb alsó sarokban és alul középen van az ENERGIAÁTALAKÍTÓ. Az eszközkategóriának megfelelő, Quadronban felvehető energia látható a fehér mezők jobb alsó sarkában. Tehát pl. a lila kristályoknak látszó tárgy a transzporterekhez szükséges energia, ha felveszed, akkor azzal a transzporterek energiáját növelheted.

A játék kezdetén, és az idegenek minden támadó hullámának elején három objektum jelenik meg a Quadronban. A támadóhullámok akkor indulnak, ha az előző hullám tagjait már megsemmisítetted, erről értesítést is kapsz.

FEGYVEREK: akár az összes fegyvert aktiválhatod egy időben, ilyenkor hatásuk összeadódik, mikor elfogy az energia, az első, leggyengébb fegyver akkor is aktív marad, a többi hatása pedig leáll.

Lézersugár: nem kell hozzá plusz energia, gyenge, vízszintesen lő.

Lézercsavar: nagyobb tűzerejű és erősebb, mint a lézersugár, de ugyanúgy csak vízszintesen lő.

Lövedékágyú: a legerősebb fegyver, függőlegesen lő.

VÉDELMI ESZKÖZÖK:

Normál pajzs: nem ér kár az idegenekkel, illetve a lövéseikkel való találkozáskor.

Nagyerejű pajzs: az idegenekkel való ütközés az ellenséged elpusztítását eredményezi, de ez nagyon sok energiát felemészt.

Turbo: sebességnövelő eszköz.

TÁVIRÁNYÍTÓK:

Mozgásgátló: amíg az eszközhöz tartozó energia engedi a működését, addig a képernyőn lévő idegenek nem fognak tudni mozogni.

Okos ágyú: egymás után pusztítja el a képernyőn lévő idegeneket számítógépes vezérlésű lövésekkel.

Álcázó: nem tudsz ütközni idegenekkel, illetve a lövéseikkel sem.

TRANSZPORTEREK: használatukhoz klikkelj az eszközre, majd a térképen arra a pontra, ahová jutni szeretnél.

Általános transzporter: Warp ugrást tudsz vele végrehajtani a Quadron valamelyik sarkába.

Multi-screen transzporter: a Quadron bármelyik részébe ugorhatsz vele, de ez sok energiát emészt fel.

ENERGIAÁTALAKÍTÓ:

Segítségével az állapotképernyőn átalakíthatod a különböző energiákat egy másik, neked szükséges fajtába.

Az idegenek néhány típusa:

FLETCHER: hozzád hasonlóan keresi a szabad kristályokat, majd ha talál egyet, akkor annak segítségével egy átalakító szobába jutva átalakul META-FLETCHER-é, ami számodra egy sokkal kevésbé elpusztítható, agresszív formája, ezért még időben válj meg tőle, ha jót akarsz.

MODUL: a második hullámtól jelennek meg, ha kilövöd őket, akkor SEEKER-eket bocsát ki magából.

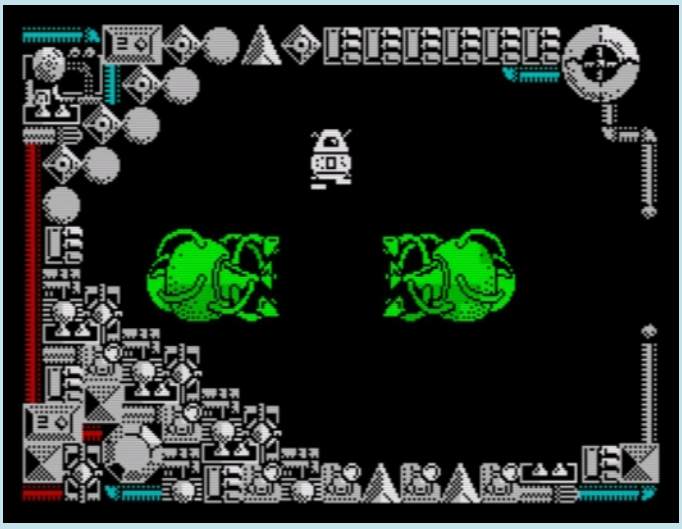
SEEKER: a legkisebb idegenek, a MODUL bocsátja ki őket, üldöznek és próbálnak elpusztítani.

WARPER: ha sokáig molyolsz egy hullám elpusztításával, akkor jelennek meg, gyorsak és általában falkában támadnak.

LURER: később jelenik meg, látszólag nem foglalkozik veled, de ha elkezded lődözni, akkor egyre jobban bepipul, öt vagy több lövéssel lehet elpusztítani.

SZONDÁK, SZUPERSZONDÁK: körbejárják a komplexumot, és képesek a kristályokat a sziklákról lebontani, így elérhetővé teszik őket a Fletcherek és számodra egyaránt.

Nekik tudtommal nincs nevük, vagy lehet, hogy ők a SZONDÁK. Elég ártalmatlanok, ellenben falkában vonulnak, tehát tömeges megjelenésükre számíthatsz.



Kristályból energia? Ahogy a fletchereknek megy a dolog, úgy a droidodnak is (elvileg). Két fontos kritériumnak meg kell felelned: kell lennie nálad kristálynak, és az egyik átalakító szobában kell lenned… Feltételezem, hogy az átalakító gépezet színe is számít, pl. a zölddel a védelmi eszközökhöz lehet energiát gyártani. De fogalmam sincs, hogy hogyan is kell… Nyomni kell a B vagy V billentyűt? Az állapotképernyőn kell klikkelgetni? Passz.

A játék folyamán, ahogy egyre ügyesebben nyúvasztod ki az idegeneket, többször találkozni fogsz különböző bónuszokkal. Például POWER bónuszt szerezhetsz, ha megölsz rövid időn belül három különböző típusú alient. Ilyenkor egy váltakozó színű POWER felirat kezd el mozogni a képernyőn, amit felvéve extra képességhez juthatsz. Sajnos a felirat bizonyos időközönként átvált REWOP-ra, majd egy idő múlva vissza POWER-re, és ha a REWOP állapotában veszed fel, akkor ahelyett, hogy gyarapodnának képességeid, pont az ellenkezőjét éred el. Másik röpködő bónusz a BONUS, amit kilőve egy pontszámot láthatsz felváltva a BONUS szóval, ha ezután felveszed, mikor a szám van kiírva, akkor megkapod a bónusz pontszámot.

Mivel a Quadron egy fizetős játék, így az ingyenes kortársaitól eltérően azt hiszem nyugodtan lehet kritizálni. A körítéssel szoktam kezdeni, ami itt szinte teljesen hiányzik, fizetni kell, de pillanatnyilag nincs a játéknak kazettaborítója. Tudom, ki másolna játékot kazettára? Hát, például én. De nem csak a borító hiánya miatt nem teszem, hanem azért sem, mert nincs a játéknak sem TAP, sem TZX változata, csak egy árva SNA (Közben TAP verzió született, már csak a borítót kell kiszurkolnom). A sztori nem egy nagy durranás, de egy sima kis lövöldözős játékhoz képest kellemesen meg lett bolondítva. A grafikával, animációval, mozgás gördülékenységével elégedett vagyok, ahogy a hangeffektekkel is. A vezérlés hagy némi kívánnivalót maga után. A billentyűkiosztás elég vacak (Q..P,A..En - fel, le, CS, X, N - balra, Z, C, M - jobbra, 0..9 – tűz, Space - állapotképernyő, V, B - kristály kibocsátása balra, jobbra), botkormányokkal (Kempston, Sinclair, Cursor) kicsit jobb a helyzet. A vezérlés más szempontokból remek. A nehézség erősen eltúlzott, sajnos ezzel le is rontja alaposan az összképet, és a hosszú távú sikertelenség megrövidíti a játékkal még kellemesen eltölthető időt.

Mezei Róbert (m/zx)

Így készült a ZX X-Mas '18

Régóta dédelgetett tervem volt egy kis karácsonyi Intro elkészítése Speccy-re. Még az előző évezredben történt, hogy hosszabb kódot írtam a kis fekete dobozra, ezért úgy gondoltam ez az intro jó kis gyakorlat lesz, hogy újra megtanuljak Z80 assembly-ben. Persze aztán lassacskán visszajöttek az emlékek, és újfent felfedeztem miért is szerettünk Spectrumon kódolni. Ez még az az időszak volt, mikor a számítógép működését tényleg értettük. Persze manapság az emulátorok is nagyban egyszerűsítik az ember életét, mert pillanatok alatt fordítják az kódot, és egyből ki lehet próbálni. Ha mellényúl az ember, és „© 1982 Sinclair Research Ltd" lesz az USR hadművelet eredménye, a korábbi állapotra másodperceken belül vissza lehet állni. Régen ez 5-10 percet is igénybe vett. Emlékszem, hogy inkább jó kódot írt az ember, és nem bízta a véletlenre, átnézte sokszor, mielőtt elengedte próbafutásra.
Az inspirációt a KKU adta a maga zsenialitásával. Az MTV 2-n, 1986 decemberében sugározták, Barát Gellért, Erdei András, Mészáros Bálint készítette.

Fogjunk neki!

Kicsi és karácsonyi. Ez a kiindulópont, ezért aztán semmi komoly dolgot nem képzeltem bele. Legyen egy jókívánság, ami izeg-mozog, zene, és mivel az ULA Plus a mostani Speccyalista Világ témája, legyen színes, ULA Plus grafika is.

Scrollozott felirat

Tehát vegyünk egy „Kellemes karácsonyi ünnepeket!" feliratot, ami szinuszosan balra, jobbra scrollozódik. A scrollozást pixel pontossággal jó végezni, ebből következően nagyon ne legyen színes sem, de legalábbis egy attribútum soron belül egyformák legyenek a színek. Magassága azért legyen szemmel látható. Mondjuk 32 pixel magas felirat pont jó lesz. Széltében meg, ahogy kifér. Próbáltam kicsit optimalizálni úgy, hogy karakterképenként tárolom csak, és kiíratom egy pufferbe, mivel így 30 betű helyett csak 17 kellene. De nyolccal osztható betűszélességgel csúnya lett volna, és lusta is voltam külön kódot írni a változó betűszélességnek. Ezért miután „rájöttem", hogy az M nagyon széles, a k, e, á, stb. közepes, de azért nem egyforma, annál a megoldásnál kötöttem ki, hogy legyen meg a grafika egyben. Ehhez ma már sok eszköz van, ami egyszerűsíti az egyszeri grafikus életét, ezért ennyit még be mertem vállalni kóderként is. Például itt van a GIMP, amivel készítettem egy 32 pixel magas rajztáblát, ráírtam, amit kellett, körvonalat adtam neki, meg egy kis szürke átmenetet alulról, fekete fillt felülre, és már kész is volt. Másik jó cucc a ZX-Paintbrush, amibe be lehet tölteni PC-s képeket, pl. BMP-t. Kell még hozzá egy csipetnyi IrfanView, mert azzal könnyebben boldogul az ember, ha 1 bitesre szeretne konvertálni egy képet. Az eredmény ez lett:

688*32 pixeles, szürkeárnyalatos kép, amiből 2 színre konvertálással ezt készítette az IrfanView:

Egészen használható lett. Volt egy kis küzdés a szürke színátmenettel, hogy mennyire világosból kezdődjön, de kb. 10 perc alatt azért sikerült eltalálni.
A 688 pixel szélesség nem véletlen, a nyolc pixelre igazított grafikát jó Spectrumon kezelni.

Hozzá a kód…

Kezdjük a scrollal! Úgy döntöttem megint lusta leszek. Egy régi tétel szerinti igazság, hogy a processzoridő és a memória átválthatók. Ezt érdemes megjegyezni, mert a mai szoftverfejlesztésben is igaz. Ez a tétel úgy materializálódott a bitekben, hogy előzetesen kiszámíttatom a számítógéppel az 1, 2, 3, … 7 pixellel jobbra görgetett változatait a KKU szövegnek, így majd elég lesz ezekből kiválasztani a szükségeset, és egy halom LDI vagy LDIR utasítással kiírni a képmemóriába.

Előzetes számítások, magyarul precalc

Adtam hozzá egy kis margót bal és jobb oldalon, 48-48 pixel szélességben, azaz 6-6 bájtnyi nullát. Így a felirat grafikájának teljes szélessége 98 bájt (788 px) lett. Egy fázis 98*32=3136 bájtot foglal, ebből van 8 darab, így 25088 bájtból kijön. Jól átváltottam a CPU-t, mert ez nem kevés memória, de legalább hexában kerek érték: $6200.
A görgetett verziók nagyon egyszerűen készíthetők. Venni kell az előző fázist, lemásolni és eggyel jobbra shiftelni.
A precalc görgetést végző kód lényege ennyi:

```
KKU_LINE_LEN         EQU   98
   ld    bc,32*KKU_LINE_LEN
   ccf
PCT_ShiftRight:
```

```
rr   (hl)
inc  hl
dec  c
jr   nz, PCT_ShiftRight
djnz PCT_ShiftRight
```

Kihasználja, hogy a jobb oldalon az üres margó miatt úgyis 0 bitek vannak, így nem foglalkozik külön a sorokkal. Az előző sorból kigörgetett 0 bit pont megfelelő a baloldalon. Senki sem fogja felismerni, hogy ez egy recycled 0, olyan jól álcázza magát.
Tehát eddig úgy néz ki a memóriánk, hogy $8000-$E1FF között van a szövegünk. Generálás közben a kezdő memóriacímeket eltettem egy lookup listába, jól fog majd az jönni.

Szinuszos mozgás

A scrollozáshoz kell egy SIN/COS táblázat, amit előre számoltatok. Tényleg lusta vagyok kódot írni ☺… de legalább azt a keveset, ami kell hozzá, Speccy BASIC-ben készítettem. Tudjuk, hogy a mozgatandó grafika 788 px széles. A képernyőn egyszerre ebből csak 168 px fog látszani. Baloldalra tervezek egy karácsonyfát, ezért 21 bájtnyi szélességben teszem csak ki a feliratot. Tehát a szinusznak úgy kell amplitúdót választani, hogy 788-168=620 legyen a két végpont között, azaz 310*COS(x). A COS vagy SIN kb. mindegy, mert egyformák a mi szempontunkból. Azért szokták inkább a COS-t használni, mert így a kezdőérték egyben szélsőérték is, nem kell a lookup táblázat közepéről indulni. x lesz a változó, a táblázat generálásakor, ami 0 és 2*π között fog szaladni. Annyi értéket kell felvennie, amennyi mozgásfázist szeretnénk kezelni. A teljes animációra 6 mp kb. elég, ebből 3-ig balra, 3-ig jobbra fog mozogni a felirat. Mivel a képernyőt másodpercenként 50-szer rajzolja ki a fekete doboz, így 50*6=300 fázist kell kiszámolni, azaz FOR x=0 TO 2*PI STEP 2*PI/300 lesz a ciklus. Egy kicsit kell még tuningolni. A 2*PI/300-ból PI/150-et csináltam, csak, hogy kevesebbet kelljen számolni feleslegesen, és eltettem egy step nevű változóba az eredményt. Ez a változó jól jön, mert a FOR ciklus úgy fut le, hogy felveszi a záró értéket, azaz a 2*PI-t is, ami viszont nekünk nem kell, mivel az ugyanaz, mint az x=0-nál, és így nem 300, hanem 301 fázist generálna le. Ezért így fog kinézni a BASIC programunk:

```
10 LET s=PI/150
30 FOR x=0 TO 2*PI-s STEP
s
40 REM itt csinálunk
valamit
120 NEXT x
```

A táblázatba írandó értéket úgy számoljuk, hogy a COS függvényt 0 és 2 közé szorítjuk, mert pozitív számokkal akarunk dolgozni, hiszen arra van szükségünk, hogy a KKU grafikát honnan kezdjük el rajzolni, melyik a baloldali pixele, ahonnan kezdve egy 168*32 px-es ablakot kirajzolunk belőle a képernyőre. Így az 1-COS(x) lesz a belseje a függvénynek. Szorozva 310-zel, és annak az egész része. Lássuk csak!

```
40 LET c=INT (310*(1-COS
x)+.5)
```

Ez volt a lényeg, innentől jöhet a memóriába írás. Úgy jó a későbbi használat során, hogy az alsó 3 bitjét külön tesszük el, mivel az mondja meg, hogy a 8 széles bájton belül mennyivel kell jobbra shiftelni a grafikát. Emlékszünk még rá, hogy ezt precalc-kal már előre kiszámíttattuk a géppel, ezért csak a megfelelő fázisú grafikát kell kiválasztani. Eggyel kevesebb gépi kódú utasítás kell majd, ha egyből a kétszeresét tároljuk az alsó biteknek, mert akkor a memóriacímeket tartalmazó táblázaton belüli helyet tárolja instant. A negyediktől bittől felfelé pedig 8-cal osztva tároljuk a számított értéket, hogy bájtpozíció legyen. Ez assembly-ben egyszerű, BASIC-ben már kevésbé. Ha már itt tartunk, írjuk meg a POKE-ot is!

```
20 LET a=40000
50 LET l=c/8
60 LET h=INT l
70 LET l=(l-h)*8
80 POKE a,2*(7-l)
90 POKE a+1,h
100 LET a=a+2
```

A figyelmesebbek már tuti kiszúrták, hogy a 110-es sort üresen hagytam. Természetesen ez nem maradhat így. Legyen mondjuk ez:

```
110 PLOT (8*h+l)/4,28*x
```

hogy lássunk is valamit a képernyőn.
Már csak futtatni kell, és kimenteni a 40.000-es címtől a memória tartalmát 600 bájt hosszan. Ezt a ZX Spin gyönyörűen elvégzi, és már mehet is az assembly forrásba.

A KKÜ szöveg rajzolása

Miután a szöveghez van egy 600 hosszú COS táblázatunk, megfelelően elkészítve, és mind a 8 shiftelt grafika betárazva, jöhet a rajzolás. Van egy mutató, ami a COS táblán belüli aktuális fázisra mutat, minden frame rajzolása után ugrik egyet előre, és ha elfogy a COS táblázat, akkor visszaugrik az elejére. A táblázat végét $FF-fel jelöltem, mert a bájtpár első tagja legfeljebb 2*7=14 lehet, így

$FF-et nem vehet fel. Ez a bájt mutatja ugyebár, hogy melyik shiftelt grafikára van épp szükség.
Az volt a cél, hogy az IM 2-es interrupt kezdetétől a felső border rajzolásának a végéig beleférjen az időbe a grafika kirajzolása, ezért volt némi küzdelmem a T időkkel. 3,5 MHz nem egy szuperszonikus repülőgép, no!

Be kellett vetni pár trükköt. Az első, hogy a képernyősorok memóriacímeit eltároltam egy táblázatba (processzoridőből memória), így ezeket gyorsan ki tudtam olvasni, nem kellett számolni. Mivel az LDI-khez szükség van a BC, DE, HL regiszterpárokra, így nem sok regiszter marad a táblázat címzésére. De azért maradt egy SP-nk. Mivel épp interruptot hajtunk végre, DI-vel párosítva, így nem áll fenn a veszélye, hogy valaki megszakítsa a futást, így ha nincs szükség a veremre, használhatjuk az SP-t. Ezzel a trükkel elég jó optimalizálásokat lehet végezni. Az rajzoló rutin elején el kell menteni az SP-t, ami praktikusan a rutin végén található LD SP,nnnn utasítás felülírásával elvégezhető.

```
  ld    (DKT_SavedSP+1), sp
  ld    sp, SCRPOS_LOOKUP_KKU
; itt majd rajzolunk
DKT_SavedSP:
  ld    sp, $1313
  ret
```

Így hát az SP-ben benne van a rajzolás cél memória címeinek táblázata, amit egy POP DE-vel meg is kapunk, és még a táblázatra mutató regiszter, azaz az SP is ugrik a következő táblázat elemre. Mindezt 11T időből. Ez azért nem rossz. ☺

A HL-be beletesszük a korábban kiszámolt COS táblázatban szereplő byte-ra mutató értéket (a 8-cal osztottat), hozzáadjuk a COS táblázat első értéke alapján kapott shiftelt grafika kezdőcímét, és már csak rajzolni kell. Itt jön a következő trükk. Az LDIR 21T ciklusig fut, mert vizsgálja, hogy a BC elfogyott-e, és ha nem, visszaállítja a PC-t 2-vel (2 byte-os az utasítás). Ezt az LDI nem végzi el, ezért az 16T ciklus alatt végez. Ez bájtonkét 5T, és mivel 21 bájt szélességszer 32 pixel magas grafikát kell kirakni a képernyőre, ez elég tetemes mennyiségű idő: 21*32*5T=3360T. Persze több memória kell a kódnak (ld. még processzorból memória elve). Lényegében 32-szer lefutó ciklusban 21 db LDI fog lefutni. Ugyan a BC-t az LDI módosítgatja, ez veszteség, de az LDIR is megtenné ugyanezt, szóval nem vesztettünk semmit, a BC regisztert mindenképp elbuktuk. Mivel az A regiszterre szükségünk lesz, - mindjárt leírom miért -, ezért a ciklusváltozónak az IX regiszter alsó 8 bites részét, azaz az IXL regisztert fogom használni. Ez pont úgy viselkedik, mint pl. az L regiszter, csak lassabban működnek az utasításai, mivel $DD prefixes op.kódja van. Ennyi még pont belefér az időzítésbe.
És, hogy mire is kell az A regiszter? A grafika bal oldalán ott lesz a karácsonyfa, ami fentről lefelé szélesedik. Bután festene, ha a scrollozott grafika függőlegesen el lenne vágva. Jobban mutat, ha kicsit ferdén végződik, azaz maszkolni kell a baloldali byte-ot.

Az első LD HL utasításba írja be a számolást végző rutin - ami az interrupt kezelő végén fut le - azt a memóriacímet, ahonnan ki kell másolni a scrollozott szöveget. A HL'-be kerül a baloldali maszk, ami 2 bájt / sorból áll, első bájton a maszk értéke, másodikon az állókép, amit rá kell OR-olni, merthogy a fa kicsit belelóg a területbe. A végén JP-t használok, mivel az 10T, ellentétben a JR 12T/7T idejével, így spórolok 59T-t a 32 lefutás alatt. 31 lefutás alatt 2T-t spórolok, és az utolsó alkalommal, amikor kilép, elvesztek 3T-t.

Így utólag átnézve annyit lehetne még gyorsítani a kódon, hogy az IXL helyett a B regisztert használja. Azaz a DKT_Loop: POP DE után egy LD C,D kéne, ez 4T, a DEC IXL + JP NZ helyett DJNZ lenne, ami 13T/8T. Egy kis magyarázat hozzá. Az LDI csökkenti a BC-t eggyel. Mivel B-t használnánk „ciklusváltozónak", így az nem módosulhat, tehát minden lefutáskor C-nek legalább 19-et kell tartalmaznia (19db LDI van a ciklusban). Mivel DE képernyőre mutató memóriacímet tartalmaz, így D értéke legalább 64, ez „pont több" 19-nél. Az LD C,$FF utasítás 7T lenne, az LD C,D-vel 3T-t spóroltunk. A DEC IXL + JP NZ 18T időt igényel, ezért a DJNZ-vel itt spórolható 5T idő, azaz lefutásonként 1T-vel gyorsabb lenne a ciklus, az utolsó alkalommal pedig még nyernénk 5T-t, így összesen 36T-t lehetne spórolni; nomeg 5 bájtot. De igazából nincs rá szükség, mert ez a rajzolás lefut, amíg a raster eléri a képernyő területet, azaz a border felső részének rajzolása alatt végez. Ez azért lényeges, mert a képernyő rajzolása alatt a raster előtt minden sorban meg kell rajzolni 11 attribútumot a multicolor karácsonyfához, így ott más nem végezhető.

És még egy fontos dolog. Ez a rutin mindig ugyanannyi idő alatt fut le, így pontosan lehet tudni, hogy kilépéskor hol tart az elektronsugár.

```
DRAW_KKU_TEXT_PROC:
DKT_KKU_STARTADDR_HL:
  ld   hl,$1313      ; 10T - #selfmod: KKU gfx
```

```
                        to put start addr
  exx                 ; 4T
  ld   hl, KKU_GFX_LEFT_MASK      ; 10T
  ld   ixl, KKU_HEIGHT_PX         ; 11T
  exx                 ; 5T
DKT_Loop:
  pop  de             ; 10T - DE=screen address
  ld   a, (hl)        ; 7T  - A=1st byte of
                        KKU's line
  inc  hl             ; 4T
  exx                 ; 4T
  and  (hl)           ; 7T - mask the left side
                        of KKU gfx
  inc  l              ; 4T
  or   (hl)           ; 7T
  inc  l              ; 4T
  exx                 ; 4T
  ld   (de), a        ; 7T - put the line's 1st
                        byte to screen
  inc  e              ; 4T
  ldi                 ; 19*16T=304T
  ldi                 ; put the next 19 bytes
                        to screen
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ldi
  ld   a,(hl)         ; 7T - mask the right side
                        of KKU gfx.
  and  %11111100      ; 7T
  or   %00000001      ; 7T
  ld   (de), a        ; 7T

  ld   de, KKU_GFX_WIDTH_BYTE-
SCREEN_WIDTH_BYTES+KKU_GFX_X_BYTE_POS+1
                      ; ^^^ 10T
  add  hl, de         ; 15T

  dec  ixl            ; 8T
  jp   nz, DKT_Loop   ; 10T
; sum: 11764T
```

Multicolor, nagyon színes

Azért nevezték el multicolornak ezt a képernyőmódot, mert attribútumonként több színt jelenít meg, mint az alap papír+tinta. Igaz annyira nem jó név, de elterjedt, így én is ezt használom.

A Speccy valójában nem rendelkezik több képernyőmóddal, csak az ismert 256*192 pixel, 32*24 attribútum van. De lehet trükközni, amivel kitolhatók ezek korlátok. A multicolor mód azzal játszik, hogy a raster lefutásával szinkronban módosítja az attribútumot, így a 8*8 pixeles attribútum helyett 8*1, 8*2, 8*4 pixeles attribútumokat lehet megjeleníteni. Mivel ez maxra tolja a procihasználatot, mást nem lehet mellette csinálni. Egészen pontosan a 8*1 módban totál pörög a Z80, és még így sem érhető el, hogy teljes sort rajzoljon ki a gép, a többi esetben marad némi idő. 8*1-es módban max. 20 attribútumot lehet módosítani egy scanline alatt, azaz 160*192 px lehet a multicolor terület. Én csak 11 attribútumot módosítok soronként, cserébe tömörebb a kód, ami rajzolja. Nem túl bonyolult az egész. Az alábbi kód ismétlődik 192-szer:

```
   LD DE, nnnn        ; itt mindig az attribú-
                        tumsor eleje szerepel
   LDI
…                     ; összesen 11 db LDI
```

Ezt a kódot nem is kell kimenteni a programban, inkább egy egyszerű kóddal beírom a memóriába. És persze egy RET a végére. A belépéskor megkapja a HL-ben, hogy honnan kell rajzolni az attribútumokat és kész is.

Ha ennek a kódnak az idejét számolgatja az ember, akkor rájön, hogy az nem 224T. 224T ideig tart a 48k-s gép ULA-jának egy sor kirajzolása a tévére. Ez így csak 186T-re jön ki. A maradék részt az teszi ki, hogy erősen együtt kell írni a képmemóriát az ULA olvasgatásaival. Azaz contended memory-ba írunk, ami lassabb, mivel amíg az ULA olvassa a képmemóriát, addig a Z80 kussol. Az történik, hogy amíg az ULA kiolvassa a pixel és az attribútum bájtokat, amiket épp ki kell rajzolnia, addig a Z80 órajelét leállítja (hosszabbítja az órajel magas impulzusát). Ezért lassabb az alsó 16kB RAM írása vagy olvasása. A mókás, hogy az I/O műveletek is lassabbak a 16k-32k tartományban, ugyanis 1 címbusz van, mint minden rendes helyen. Az, hogy a contention, azaz a lassulás mennyi is valójában, számolható, de nem triviális, viszont a jobb emulátorok elég jól mutatják, így inkább kísérletezéssel érdemes ma már belőni pontosan egy-egy raster lefutáshoz igazított rutint. Azért sem egyszerű az élet, mert, mikor a bordert rajzolja a gép, nincs késleltetés. Mikor a pixeleket, akkor pedig egészen változatos, attól függően, hogy épp hol tart a Z80 az utasítás feldolgozásában. Annak idején a Shock Megademo ezért volt zseniális, mert ott ezek számítottak. Ha ez a demó jól fut egy emulátoron,

akkor már elég jó az emuláció. Az ULA Plus fejlesztésekor is ez a demó volt az egyik tesztalany.

Van egy kis variálás itt is. Ez amolyan Colombo effektus: ennyi lenne, de azért még egy kérdés… ☺ Úgy gondoltam, hogy imitálva a szélesvásznat, kicsit vagányabban néz ki az intro, így border színt kell váltani, mikor a raster a 256*192-es rész első sorához ér. Viszont az elektronsugarat, vagyis az ULA-t meg kell előzni a multicolor sor attribútum értékeinek kiírásával. Ezért a multicolor rutint még az első sor feletti rastersor közepén el kell indítani, kb. a 11. attribútumnál, azaz a 88. pixelnél. Viszont itt még a bordernek sötétnek kell lennie. Ezért az első sor attribútumait kiíró LDI-k közé be van csempészve egy LD A,*világos border* színe és egy OUT ($FE),A utasítás. Ez a két utasítás 7T+11T=18T ideig fut, azaz 2 LDI-t kiütött, mert azok 2*16T=32T ideig tartanak. Viszont még illendőségből ki kéne tölteni valamivel a kódot, hogy a 32T-nyi idő elteljen, ezért egy DEC BC és 2 db NOP is bekerül az első sorba, amik elszórakoztatják a Z80-at 14T ideig. Az első multicolor sor így csak 9 attribútum széles, de ott pont nincs semmi baloldalon, ezért ezt nem lehet észrevenni. 48k-s gépen egy scanline 224T ideig rajzolódik. 128-ason 228T-ig. Egy NOP a különbség, de lustaságból inkább úgy lőttem be az időzítést, hogy ez ne számítson. Azzal elég volt variálni, hogy az IM2 rutin kezdete után mikor induljon el a multicolor rutin.

Az újabb 128-asokon viszont már nem működik, mert ott a contended memória kezelése változott. Játszásiból megírtam a rajzolót úgy is, de ezt a végleges változatba nem tettem bele, mert akkor még a géptípust is detektálni kéne. Ennyi plusz kóddal soronként az Amstrad gépeken is jól futna:

```
    ld  b, 9
GMC_NopLoop:
    ld  (hl), 0
    inc hl
    djnz GMC_NopLoop
```

Mikor az induláskor az intró széthúzza a bordert, és azzal együtt lerolózza a multicolor képet a karifára, csak annyi történik, hogy bedob egy RET-et, először az első sor után, majd a 2. sor után, stb. azaz korábban kilép ebből a rutinból. Faék egyszerű megoldás.

Még csak elméletben van meg, és lehet, hogy nem működik, de… A szélesebb területű multicolort úgy lehetne megoldani, hogy az SP-t rá kéne állítani a rajzolandó attribútum sor végére, és LD HL,nnnn + PUSH HL-ek tömkelegét betolni. Így LD/PUSH páronként kitehető 2 attribútum, és csak 21T időt venne igénybe, ami jóval barátibb, mint 2 db LDI 32T-s ideje. Plusz kell még soronként egy 10T ideig tartó LD SP,nnnn. Ezzel a módszerrel egy gond van csak: jobbról, balra rajzol, és szembe megy az elektronsugárral. Ezért csak 128k-s gépen a kilapozott képernyőn lehet jól használni. Ezért soronként még egy memórialapozást is el kell végezni, de még ezzel együtt is gyorsabb lesz az LDI-s módszernél.

Ha pedig 2*8-as vagy 4*8-as multicolort készít az ember, bőven van idő attribútumokat módosítgatni, és még marad is némi T kapacitás egyéb dolgok elvégzéséhez. De itt sincs elég idő egy teljes sornyi attribútum módosítására egy vízszintes sor alatt, ezért inkább a 128-as képmemória váltogatásával érdemes operálni, és a váltást akkor elvégezni, mikor épp a border rajzolása folyik.

Grafika és zene

A grafikát kóderként csak-csak bevállalom, mert Spectrumon a szín és a felbontás korlátaiból adódóan könnyebb egy képet megrajzolni, mint későbbi hardvereken. A zene már más tészta, ahhoz végképp nincs tehetségem.

Rajzoljunk egy kicsit…

Azért egyszerűsítettem az életemen, és a grafikai alapot levadásztam a netről. Találtam egy weboldalt, ahol garmadával voltak karácsonyi témájú clipartok.

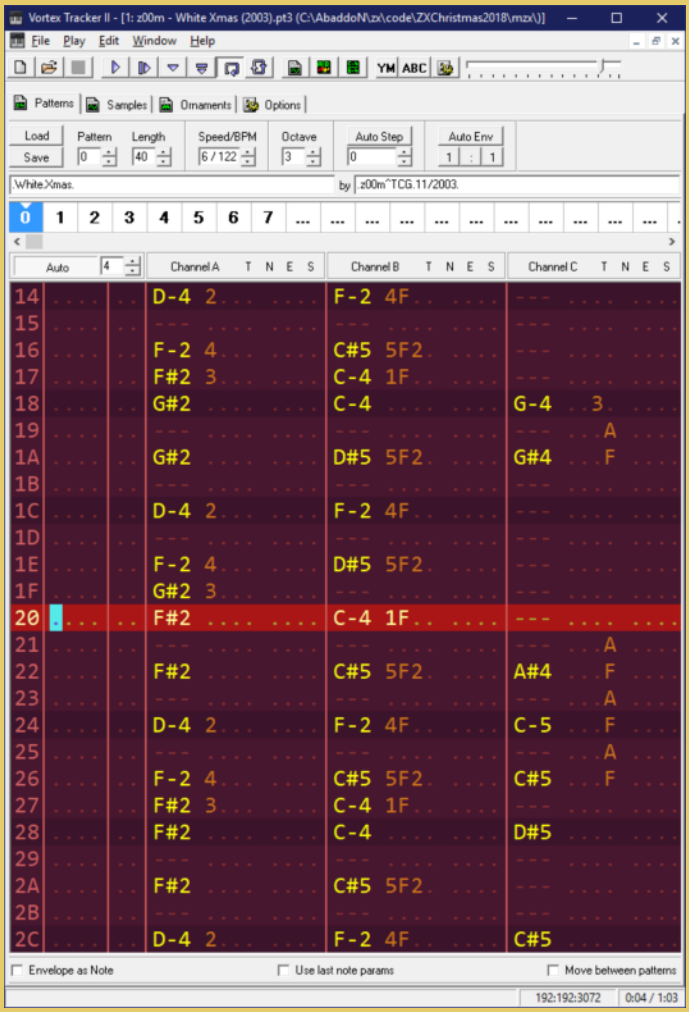
A karácsonyfát, a zoknikat és a masnikat összeszedtem, majd GIMP-pel készítettem egy-két koncepciót, aztán jöhetett a konvertálás. Olyan képet kerestem, ami stilizált, mert alacsony felbontásban elvesznek a részletek, leginkább a tűlevelek. Készítettem IrfanView-val egy 256*192-es felbontású képet és a színeket is levettem 64-re. Ezt odaadtam az Image2ULAplus-nak, amit a ZX-Paintbrush mellé javasol a szerzője PC-s képek konvertálásához. Elsőként kézzel optimalizáltam a palettát, mivel az Image2ULAplus elég nagy katyvaszt számol magától. Az ULA Plus palettának van egy tap formátuma, ami kvázi szabvány, ezt ismerik a szerkesztő programok. Itt van róla leírás: https://faqwiki.zxnet.co.uk/wiki/ULAplus

A 4 palettát úgy konstruáltam meg, hogy mindegyik 0-s színe a háttérszín legyen, és legyen egy, amelyikben csak zöldek vannak a tinta és a papír színekben, egy ahol csak a többi (piros, lila, sárga), és egy ahol keverve vannak a színek. Végül egy palettát meghagytam a többi rész megrajzolásához. Erre a strukturálásra azért volt szükség, mert a konvertálás után még akad bőven pixelezési utómunka, és így sokkal könnyebben lehetséges változtatni az attribútum értéken. Össze-vissza színeknél tuti nem lesz pont olyan papír+tinta páros, ami kell az embernek. A paletta szerkesztésére a ZX-Paintbrush tökéletes. A kész palettát el lehet menteni, odaadni az Image2ULAplus-nak, ami szinte ugyanolyan jó - avagy rossz - képet konvertál fix palettával is. Hogy mennyire jó a konverzió, azt döntse el mindenki a képek alapján.

Baloldalon a nyers konverzió, jobb oldalon az átrajzolt. A piros vonal a multicolor terület határa, attól jobbra már csak 8*8-as attribútumok vannak a jobb oldali képen. A baloldalin mindenhol 8*1 pixelesek vannak, így lehet csak konvertálni. Vagy 8*8-as vagy 8*1-es a teljes kép. Pár napnyi pixelvadászat után lett grafika a háttérhez, és a töltőképernyőhöz is. És mivel illendő igazi Speccy-n is futni, ezért a színeket normál palettára is elkészítettem. Az persze közel sem ennyire szép, viszont futott volna ‘82-ben is.
A pixelek mindkét esetben ugyanazok, csak az attribútumokat változtatja az intro hagyományos ULA és ULA Plus szerint.

...és lőn világosság, de most már legyen hang is!

Ahogy említettem a zenéléshez fikarcnyit sem konyítok. Mit lehet ilyenkor tenni? Idézve egy klasszikust: „Lopni, megyünk lopni / A Pityinger, meg a Váradi / És lopni, megyünk lopni…” Speccy-n a sok tracker közül az idők során a Vortex Tracker II vált egyeduralkodóvá. Ez egy PC-s program, és nagyon könnyen használható.



A trackereket az Amiga korszakában találták fel; vagy legalábbis akkor élték fénykorukat. Lényegük, hogy vannak a hangszerek, amiket egyesével meg lehet szerkeszteni, vagy ha digizene-képes számítógépünk van, akkor lehet digitalizált hangot is használni. AY-on ez nem játszik, mert túl sokat enne a prociból a digizene lejátszás, így marad a szokványos AY használat. Van 3 csatornánk. Ezt a három csatornát lehet szerkeszteni, egy-egy úgynevezett patternben megírni. Egy pattern többnyire 64 sorból áll, soronként lehet változtatni a hangszert, hangmagasságot, és egyéb effekteket, pl. arpeggiot mellé tenni, hangerőt változtatni. A patterneket egymás után fűzve, újra lejátszva stb. áll össze a track, azaz a teljes zeneszám. A sebesség is változtatható, ahogy a sorokat görgeti a lejátszó. Legjobb, ha megnézitek élőben a VT II-t, úgy lényegesen könnyebben megérthető a filozófiája.
Szóval jó pár karácsonyi témájú Speccy zenét végighallgatva, végül z00m: White Christmas című, 2003-as alkotása mellett tettem le a voksomat.
A Vortex Tracker van olyan kedves, és készít egy bináris fájlt, amiben benne van a lejátszó és a zene is. Ezt csak be kell INCBIN-elni az assembly forrásba,

és már mehet is a muzsika. A helye őneki a $6000-től kezdődő memóriatartomány lett. Pofonegyszerű kezelni. CALL $6000 az inicializálás, CALL $6005-öt minden 1/50-ik másodpercben meg kell hívni, és már zenél is a Speccy. CALL $6008-cal hallgattatható el a zene. A $600A címen, a 0. bittel szabályozható, hogy kezdje-e elölről a zenét, ha véget az ért. A $600B címről kiolvasható, hogy épp hol tart a zenében a lejátszás. Ez utóbbi egyébként komolyabb demóknál jól tud jönni a zenéhez való időzítéskor. Egészen barátságos a lejátszás. A legtöbb esetben kb. 3500-4000T időből kihozza a CALL $6005-öt, de időnként - tippre pattern váltáskor -, 8400T-ig is felszalad. Ezért a nagyobbik értékkel számoltam minden időzítést.

Hull a hó és hózik…

De mielőtt elkezdene havazni, még egy kis trükk… Mivel nem foglalkoztam a 48k-s és a 128k-s Spectrum közötti különbség kezelésével, ezért a multicolor úgy ér véget, hogy az utolsó pixelsor alatt valahol végez, de nem egyformán a két gépen. A két gép eltérő időzítése miatt nem lehetett úgy belőni, hogy az utolsó sor border rajzolásának végén fejeződjön be. Tehát a szélesvászon imitáció alsó borderszín váltása a sor közepén van, de hogy kevésbé legyen feltűnő, a jobb alsó sarokba került pár pixel, ami átmenetet képez a világosabb, lilás és a sötétkék terület között. Lustaság fél egészség. ☺

És akkor a hó… Maradt egy kevés idő még az IM 2-es rutinból, amit vétek lett volna nem felhasználni. Így született meg a havazás. 32 hópihe, azaz magyarul 32 pixel rajzolása fért bele az időzítésbe. Ez kb. elég is lett arra a minimális, 20 pixel magas területre. De azért nem volt triviális elkészíteni. Először is 3 részre szedtem a kódot. Egyik számolja a hópelyhek esését, balra-jobbra mozgását, vigyáz rá, hogy ne szaladjanak el nagyon oldalra. Egy másik rész az X-Y koordinátákat számolja át képernyőmemória címekké, és végül a harmadik törli az előző fázist, és rajzolja meg az újat. Ezek így darabonként belefférnek 4-5000T-be. A mozgás sebessége pedig 3/50 mp lesz, azaz kb. 17* mozdulnak másodpercenként a hópelyhek, ami még látványra is jó. Ez alatt az idő alatt pont 1 pixelt esnek, és kvázi random balra, jobbra mozognak.

Csak egy-két érdekességet emelek ki a kódból. Az egyik, hogy a véletlenszerű balra/jobbra mozgáshoz egy 256 bájt hosszú táblázatot használok, amiben $00, $FF, $01 bájtok vannak. Ezt pont hozzá lehet adni az X koordinátához, és akkor már csak azt kell figyelni, hogy balra vagy jobbra túllóg-e azon a területen, ami ki van jelölve a hóhullásra. Mivel +/-1 a maximális mozgás, így a visszakorrekció a carry flages összeadással vagy kivonással elvégezhető:

```
  cp   80        ; left move can be max. 1 px,
  adc  a,c       ; CF=1 if left overflow

  cp   239       ; right move can be max. 1 px,
  ccf
  sbc  a,c       ; CF=1 if right overflow
```

Ez elég tömör és gyors, mert nincs szükség ugró utasításra.
Egy másik mókás megoldás, hogy a képernyőcímeket számoló eljárás 2 puffert használ felváltva. Egyikbe kiszámolja 2 bájton a memóriacímet, és 1 bájton tárolja, hogy hányadik pixelt kell kigyújtani. A másik pufferben a korábbi számolás eredménye van, ami egyébként a képernyőn is kint van. Utána jön a rajzoló rutin, ami először fogja az aktuális adatokat tartalmazó puffert, lenullázza a 32 memóriacím tartalmát, így törli a korábbi fázist. Ezután az új pufferben szereplő értékek alapján „kitalálja”, hogy a memóriacímre milyen bájtot kell OR-olnia a pixel rajzolásához. Van némi SP használat, mert így gyorsan lehet írni és olvasni a memóriát, ráadásul egyből 2 bájtonként, ami itt igencsak hasznos. Meg van néhány 256 bájtos memóriahatárra igazított táblázat is, hogy a H-ba lehessen tölteni a cím felső bájtját, az L-be pedig az index értéket. Például a képmemóriába írandó bájtot ennyiből elő lehet állítani:

```
  pop  hl           ; L=draw X position
  ld   a, l
  and  $07
  ld   l, a
  ld   h, CSS_PlotBit>>8
  ld   a,(hl)       ; A=mask for plotting
                      (1 bit is set)

ORG    CSS_PlotBit
                    ; must be $100 aligned
  defb $80,$40,$20,$10,$08,$04,$02,$01
```

Volt még egy furcsa érdekesség. Első változatban úgy működött, hogy ahol lent kiesett a hópehely, ott jött be újra felül, azaz az X koordinátáját a felső pozícióba helyezéskor nem változtattam meg. Ez azzal járt, hogy balra, vagy jobbra összetömörültek, úgy 5 perc elteltével. A bal-jobb random táblázattal addig játszottam, amíg pont annyi $01 és $FF lett benne, hogy már nem rendeződtek egy oldalra azok a fránya pelyhek… nem egyre, hanem kettőre, és középen nem maradt senki. Végül az lett a megoldás, hogy készítettem egy 31 hosszú random számokból

álló start X pozíciós táblázatot, abból választ új kezdő X értéket a kihullott pihéknek a kód. Azért 31 hosszú ez a számsor, mert körbe-körbe olvasom, és így a 32 pihe nem mindig ugyanoda kerül, jóval természetesebbnek hat a mozgásuk.

ULA Plus vs sztenderd színek

Az ULA Plus működése zseniálisan egyszerű. A bright és a flash bitet nem az eredeti céljukra használja, hanem arra, hogy megcímezzen vele egyet a 4 lehetséges paletta közül. Egy paletta a szokásos, 8 papír és 8 tinta színből áll össze. Így van összesen 4*16, azaz 64 színünk. A slusszpoén, hogy ezek a színek mind átdefiniálhatók. 8 bites RGB-ben lehet megadni, hogy melyik színkód milyen színű legyen. Az R és a G 3 bites, azaz 8 érték közül lehet választani, a B 2 bites. A 0 a fekete, az $FF a bright 1 fehér. És még az sem kötelező, hogy az ugyanolyan értékű tinta és papír egyforma színű legyen. Vagyis mondjuk a 0. paletta 1-es papír színe lehet kék (RGB=%00000011), és az 1-es tinta sárga (RGB=%11110100). Így, ha az attribútum bájt értéke %00001001, akkor kék alapon sárgával lehet rajzolni. Ez azért kiváló ötlet, mert a papír színe 1-2 színnel letudható a legtöbb grafikánál, viszont így a többi érték is kretívan kihasználható, sokkal színesebb grafikák születhetnek.

Az ULA Plus jelenlétét egyszerű detektálni. Ki kell választani egy regisztert, ami I/O műveletekkel történik. Ha kis várakozás után visszaolvassuk a porton lévő értéket, és visszakapjuk a kiválasztott regiszter értékét, akkor találtunk egy ULA Plus-t a gépben. Ezt összevontam a paletta beállításával, valahogy így:

```
SET_ULAPLUS_PALETTE:
  ld    bc, $BF3B     ; register select
  ld    a, $40        ; mode group
  out   (c), a
  ld    a, 1
  ld    b, $FF        ; choose register port
  out   (c), a        ; turn palette mode on

  halt
  in    a, (c)
  cp    1
  jr    z, SUP_ULAPlusDetected
  ; normal Speccy detected
  xor   a
  ret
SUP_ULAPlusDetected:
  ; ULA Plus detected
  ld    hl, ULAPLUS_PALETTE
```

A későbbi rész már csak annyit csinál, hogy végigírja a 64 színregisztert a paletta RGB értékeivel.

Start Me Up

Az intro indulásakor kicsit bohóckodtam a képernyővel, mégse legyen csak úgy „in medias res”. Ez tényleg nem nagy művészet, csak igen pepecselős. A készítés idejének legalább a harmadát ez a rész vitte el.

Például van benne grafika elporlasztó effekt. Ezt, ha törlésre használja az ember, akkor még csak extra memória sem kell neki, elég ha a képernyő adott területén lévő bájtokat egyre kevesebb bitet tartalmazó maszk bájttal AND-elgeti. Készítettem BASIC random generátorral 20 db 8*8 pixeles maszkot, egyre kevesebb egyes pixel tartalommal. Az utolsó egy 8 db 0 értékű bájtot tartalmazó maszk, ami véglegesen törli a pixeleket. Az effekt lényege ennyi: LD A,(DE) / AND (HL) / LD (HL),A. DE-ben van a maszk memóriacíme, HL-ben a képernyő. De ez nem lesz valami gyors, ezért csak a szükséges területeken végzem el. 11 téglalapot porlasztgat a kód, de minden 1/50-ed másodpercben csak egy kicsi területnyit. 8 lefutásonként lép a következő fázisra, azaz a sebesség 50/8, azaz 6,25Hz lesz. Ez már szemmel felfedezhető frissítés. Amúgy érezhetően lassú is.

Ugyanezt a módszert visszafelé eljátszva lehet felépíteni egy grafikát. Itt annyival nehezebb dolgunk van, hogy valahol hátul a memóriában még el kell helyezni a kirajzolandó képet, és annak már nem maradt könnyűsúlyú címregiszter, így kénytelen voltam az IX-et használni. Ez esetben ennyi a lényeg: LD A,(DE) / AND (IX+0) / LD (HL),A. Nem tűnik sokkal lassabbnak, de ha hozzávesszük, hogy már a DE és a H mellett az IX-et is növelni kell minden ciklusban, kezd kellemetlenné válni a dolog. Nem is fért el annyi T-ben, mint a szétporlasztás.

Ezért varázsolgattam a karifa ki-/bescrollozásával. Mindjárt elmondom, hogy mit is, de előbb elmagyarázom a scrollozást. Az volt az alapötletem, hogy a fát a töltőképernyőről csúsztatom balra. Igen ám, de közben át is kell kicsit alakítani, mert nem pont egyforma a töltés alatt látható és a futás alatti fa. Ezért megy ki teljesen a képernyőről, és jön vissza. És azért ilyen furcsamódon eltolt sorokkal, mert a teljes magasságú scroll nem fér bele az időzítésbe, ...és még kettőben sem fér el, mert ott van még a porlasztó rutin, ami 19000T körül zabál, a zene is 8600T, némi vezérlés is kell az egésznek, úgyhogy el kellett férni 29000T-ben. Ráadásul kifelé scrollkor könnyen

megoldható, hogy csak olyan széles sort mozgasson az ember, amennyin grafika is van. Itt jól jött, hogy háromszögszerű a fa. De visszafelé már túl bonyolult lett volna számolgatni, és vészesen közel volt a december 24-edike, így befelé már szép széles, 15 karakternyi szélességű területet mozgatok. Az meg már még több, 31800T-ben fér el. Amúgy van egy karaktersort bájtonként scrollozó rutin, amit szép sorban fentről indítva egyre tovább haladva lefelé hívogatok, ez adja a ferde scrollozást. És képharmadonként dolgozik, vagyis 1 frame-ben csak 1/3-ad képernyő mozog. Ezt annyira nem szúrja ki az ember, ha direkt nem figyeli. A végén, ahogy fogynak a mozgatandó sorok, úgy lesz egyre gyorsabb a fascroller. Ezért van a „beporlasztás" a befelé scroll vége felé indítva, mert így már elférnek egymás mellett 1 frame-ben. A porlasztásoknál az attribútumok írását kb. teljes effekt harmadánál jó elvégezni, úgy a legkevésbé zavaró a színváltás.
Az indulás vége felé még van némi precalc, itt másolja végleges helyére a multicolor színeket attól függően, hogy milyen ULA-t talált, generálja a multicolor kódot, előállítja a KKU szöveg mind a 8 pixellel eltolt fázisát. Ugyanis a végén már jó sok memóriaterületet felül lehet írni, hiszen a lefutott startup effektekre továbbiakban nincs szükség. Szokás szerint ennél a programnál is vadásztam a végén a szabad memóriát, de aztán egy kis átrendezéssel maradt is néhány kB-nyi belőle.

Az egész indítási procedúrát egy IM 2-es rutinból intézem, ami egy listából veszi, hogy épp miket kell csinálni.
Ilyesmi táblázattal dolgozik. 1 word, hogy melyik frame-től, a következő word, hogy melyik frame-ig, és még egy word, hogy melyik szubrutint kell hívni. Ezzel szerencsére elég egyszerű volt dolgozni, játszadozni az effektekkel. Aztán miután végzett az összes effekttel, átvált egy másik IM 2-es rutinra, ami a fő részt futtatja.

```
; vaporize out loading screen's title and ribbon
(12600T-18700T)
defw   $0000, $00A0, DO_VAPORIZE_OUT

; play music in every step (4600T-8145T)
defw   $0000, $FFFF, MZX_PLAY_PROC
```

Csak egy kis értekezés még az IM 2-ről, és a low budget tervezésről. A Z80-nak úgy kéne működnie 2-es megszakítási módban, hogy az adatbuszon kap egy páros számot, ami egy 128 memóriacímből álló táblázatba címez, és az abban található címre ugrik a futás. Az adatbuszon a Spectrum esetében egy random érték van. Hurrá, máris ellőttünk 257 bájtnyi memóriát tök feleslegesen. Azért 256+1 bájt, mert, ha az adatbuszon 255 van, akkor az ugrótáblázat 255. és 256. helyéről olvassa ki a címet. És hogy ez ne legyen elég, mivel páros és páratlan számok is lehetnek az adatbuszon, ezért az interrupt táblázat csak azonos bájtokat tartalmazhat. Ezért úgy néz ki a memória teteje, hogy $FDFD lett a belépési címe az IM 2 rutinnak. Ott csak egy JP nnnn szerepel, ami a ténylegesen futtatandó rutinra ugrik. Így könnyen módosítható, hogy épp melyik rutin fusson IM 2 megszakításkor. A JP nnnn pont 3 bájt, tehát $FE00-tól van hely a 257 bájtos táblázatnak. Felette pedig pont elfér a stack.

Tömörítés, TZX gyártás és loader

A végén még formába kellett önteni az intrót, hogy késznek nyilváníthassam. Írtam egy kis BASIC programot, ami egy REM-ben elhelyezett assembly kódot futtat. Ez a kód megnézi van-e ULA Plus, kiír pár dolgot a képernyőre, és töltöget, meg kicsomagolgat.

A gyári loadert csöppet átalakítottam, hogy ne lehessen BREAK-kel megállítani. Ha már módosítottam, a VERIFY részt is kidobtam. A sarkok lekerekítéséhez kitettem az ívet tartalmaz 8*8 pixelt a sarkokba, és mikor a loader a bordert színezi, a 4 sarokban lévő attribútumot is átírja. Látszik is néha, hogy egy-egy sornyit elcsúszik a borderhez képest. Az LD-EDGE_1 ROM rutin egyébként tartalmaz elég sok szabad időt. Van benne eleve egy ilyen rész:

```
LD_EDGE_1:
  ld    a, $16    ; Wait 358T states before
                    entering the
LD_DELAY:         ; sampling loop
  dec   a
  jr    nz, LD_DELAY
```

Ennek helyére bőven lehet például egy visszaszámlálót tenni. Vagy lerövidítve az időzítéseket, az 1500 baud helyett, úgy emlékszem jó kazettára, jó magnóval, lehetett 3000 baudos turbót is készíteni. Persze arra már nem bíznám rá a féltve őrzött programokat, mert esélyes, hogy pár betöltés után nyúlik annyit a szalag, hogy már nem lehet betölteni a programot.
Tehát írom az attribútumokat a sarokban, így kb. 100T plusz időt felhasználok az éldetektáló rutinban. Ezért kicsit módosítani kellett az időzítésen. Mivel a késleltető ciklus (az egy másik, nem a fenti) magja, 59T ideig fut, az azt jelenti, hogy az éldetektálás konstansait 2-vel emelni kell. Annyira túl van méretezve a kazettás betöltő, hogy pár T eltérés abszolút nem zavarja meg. Még az sem érdekes, hogy a módosított loaderem ciklusmagja csak 58T

hosszú. Ügyesen kitalált dolog volt a Speccy ROM programja.

Tömörítsünk, mert a méret igenis számít

Létezik néhány elég jó tömörítő program Spectrumra. A legjobb a ZX-7, az Exomizer és a MegaLZ. Én ezek közül az utolsót vettem elő, mert nagyon könnyen használható. Van egy megalz.exe, ami egy bináris fájlból egy kisebbet gyárt. Adnak hozzá egy kicsomagoló Z80 kódot, ami kicsi is, gyors is, könnyen lehet használni. Jó rátával tömörít, ami egyébként a másik kettőre is igaz. Ebben a táblázatban látszik, hogy mit sikerült összehoznia.

	Eredeti méret	Tömörített méret	Arány
Töltőképernyő	6144+2*768		
	=7680 bájt	4562 bájt	0,594
Program	23 185 bájt	12 765 bájt	0,551

És még a töltési idő is jelentősen csökkent, mivel nagyjából fele annyi bájtot kell betölteni. Én a végére nagyon megkedveltem. Úgy is használható, hogy az épp futtatandó részt csomagoltatja ki vele az ember, és ezzel jobban el lehet férni a memóriában. Pl. játékokhoz háttérgrafikákat elég tömörítve tárolni; 3 tömörítetlen kép helyén elfér 5 tömörített.

Legyél kazetta

Vagy legalább legyél TZX fájl. A zx-modules.de oldalon lévő programok közül ez a darab is igencsak jól sikerült. Összeépül a ZX-Editorral és lehet egyből hozzáadni BASIC programot a TZX image-hez, amit egyből meg is lehet szerkeszteni. Igencsak kényelmes eszközkészlet. Megírtam a BASIC programot, tettem egy REM-et a végére, és kellő mennyiségű szóközt. Aztán a szóközökre beolvastattam a ZX-Spinnel lefordíttatott assembly loader-t, és ennyi. Kész az első blokk. A másik kettőt elkészítettem MegaLZ-vel binárisba, betöltöttem a ZX-Blockeditorba, és lett egy jó kis kazettám … helyett egy kazettát helyettesítő fájlom.

Mi nem fért bele?

Először is, ami még belefért: egy easter egg. Ezt találja meg mindenki magától, nem árulok el többet. ☺ Viszont volt, ami már nem fért bele, leginkább időhiány miatt. Nem túl szépen rolózza le az intro a multicolor területet. Mikor egy 8*1-es attribútumsornál véget ér a kirajzolandó terület, - mert ugye indul fentről, és 192 lépésben halad lefelé - tehát véget ér, és onnantól lefelé már nem multicolor, akkor az attribútum értéket nem a 8*8-as attribútumokkal megrajzolt színnel tölti fel, hanem, az marad, ahol épp abbahagyta. Ez nem annyira szép. Elvileg elférne a memóriában még ez a kód, de majd egy másik alkalommal, egy másik programban fogom inkább ezt megírni.

Későn jöttem rá, hogy a start effekteket felesleges volt IM 2 rutinba tenni. Ezeket simán normál kódból lett volna jó futtatni, sokkal egyszerűbb lett volna az életem. Pl. nem kell bontani a rutinokat, és elég lett volna csak a zenének mennie IM 2-ben. De már majdnem kész voltam, mikor észbe kaptam.

El lehetne készíteni a későbbi ZX 128-asokra, Pentagonra, Scorpionra, Didaktikra, és még jó ég tudja milyen klónokra. De majd inkább egy következő alkalommal játszadozok ezeknek a gépeknek időzítésével.

Hát ennyi, így készült…

Meglepően hamar visszatértek az emlékeim a Z80-ról és a Spectrumról. Persze a PC-k, emulátorok és internet korában sokkal könnyebb infót találni, tesztelni, fejleszteni. De ez inkább hozzáad, mert régen szenvedtünk a lassú töltésekkel, a gumibillentyűvel, az áramszünettel. Megvolt a maga bája, de azért nem sírom vissza. Így azért könnyebb az élet, és jobban lehet koncentrálni az alkotásra.

Felhasznált eszközök

GIMP - GNU IMAGE MANIPULATION

IrfanView

ZX-Paintbrush
és hozzá építve Image2ULAplus

ZX-Blockeditor

ZX-Editor

MegaLZ: sajnos nehezen lelhető fel; egy Linux-ra fordított verziót tartalmazó, GitHub-os project található csak. Feltesszük a sinclair.hu-ra.

Vortex Tracker II
Update (csak .exe)

ZX-Spin 0.7

Taletovics Dávid (G.o.D)

PROGRAMOZÁSTECHNIKA - NEXT BASIC

KÉREM A KÖVETKEZŐT! - 1. RÉSZ

avagy NextBASIC újdonságok...

A sztárfellépő késik, de azért belesünk a függöny mögé és felvázoljuk, hogy mi újat hoz majd a Next BASIC.

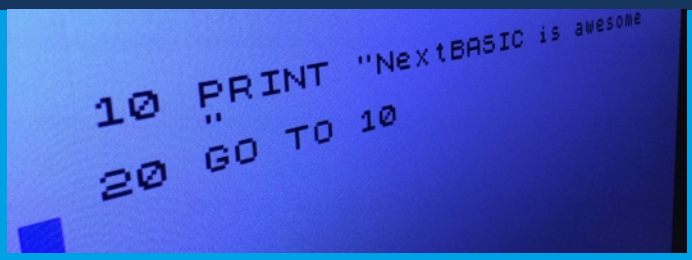
Számos meghatározó hardveres újdonsággal áll elő a ZX Spectrum Next, és a legtöbbhöz szerencsére BASIC nyelvi támogatás is társul majd, hogy ne csak azok aknázhassák ki a vas lehetőségeit, akik a kontinentális talapzat mélységéig ismerik a gép felépítését és a gépi kódú programozást. Nagy a csábítás, hogy a grafikus képességek vagy a hang irányából kezdjük a lehetőségek tárgyalását, de indításként maradjunk inkább a legalapvetőbb erőforrás, a memória kihasználásánál.

A ZX Spectrum Next 1 vagy 2 Mbájtosos RAM kapacitása a 80-as években bármilyen háttértárnak is becsületére vált volna, memóriaként pedig kifejezetten elképzelhetetlen volt, legalábbis az otthoni számítástechnikában. A Z80 azóta is csak 16-bites címbusszal dolgozik, ami a tudomány mai állása szerint is csak 65536 memóriarekeszt enged közvetlenül címezni, így ha el szeretnénk érni az extra memóriát, ahhoz trükköznünk kell. Szerencsére ezt elintézi helyettünk a NextZXOS és a Next BASIC. A BANK utasítás segítségével a tár bármelyik 16K-s blokkját (bankját) használhatjuk memória- és fájlműveletekhez.

Persze ez sem úgy van, ahogy mi mórickásan elképzelnénk, nem cím szerint szekvenciálisan osztották ki a bankok sorrendjét, hanem a korábbi 128-as Spectrum modellekkel való kompatibilitás érdekében ragaszkodtak a hagyományokhoz, így a NextZXOS használja az első 9 bankot:

Srsz. Foglalás

0 Standard ZX Spectrum memória (címtartomány: 49152-65535)
1 RAMdisk
2 Standard ZX Spectrum memória (címtartomány: 32768-49151)
3 RAMdisk
4 RAMdisk
5 Standard ZX Spectrum memória (címtartomány: 16384-32767)
6 RAMdisk



7 A NextZXOS számára fenntartott munkaterület
8 A NextZXOS számára extra képernyőmódokhoz (lo-res, Timex hi-res és Timex hi-colour módok) és egyéb adatokhoz fenntartott munkaterület

A bankok indexelése 0-tól indul, bank 9-től kezdve a memória a programozó rendelkezésére áll, 1 MB-os RAM méret esetén 47, 2 MB-os RAM esetén 111 a legnagyobb használható bank index. (Ezt adja vissza a MAXBNK rendszerváltozó értéke is.) Lehet garázdálkodni a 48K-s Spectrum standard területét képező 5, 2 és 0 sorszámú blokkokban is, de óvatosabban: csak az 5-ös bank 0-6911-ig terjedő képernyőtára biztonságos és a RAMTOP rendszerváltozó feletti részek, egyébiránt könnyen belerondíthatunk a BASIC munkaterületbe és a standard rendszerváltozókba, melyek a képernyőtár felett sorakoznak.

Ahogy a táblázatban látható, az 1, 3, 4 és 6-os számú szeleteket a jó öreg RAMdisk használja, a 128-as Spectrum tulajdonosok BASIC-ből leginkább ennek használatával, azaz fájlműveletekkel tudták kihasználni az extra memóriát. Ha valakinek nem lenne elég az, ami a 9-es banktól felfelé rendelkezésére áll, akkor a RAMdisk területeket felszabadíthatja a BANK 1346 USR paranccsal. (Ekkor persze RAMdisk balra el.) A 7-8-as bankokat a BASIC BANK parancsok segítségével nem lehet elérni.

A következő parancsok használhatók a BANK-kal kombinálva:

```
POKE, PEEK, USR, COPY TO,
ERASE, FORMAT, CLEAR,
LAYER.
```

Az alábbi példák segíthetnek megérteni a BASIC adta memóriakezelési startégiát:

```
BANK 9 USR 49152
```

Elindítja a bank 9-ben elhelyezett gépi kódú

programot az adott címtől. Előtte belapozza a megadott indexű bankot a 48K-s memória legfelső 16K-s szegmensébe, így a címnek mindig 49152 és 65535 között kell lennie. Ügyelj a RAMTOP-ra és az UDG-re!

```
BANK 10 POKE 0,255
```
Elhelyezi a 255-ös decimális értéket a 10-es bank legelső (nulladik) byte-jában. Egy-egy érték beszúrására célszerű utasítás, de aki egész blokkokat másolna FOR ciklusba ágyazott BANK … POKE utasítással, az legyen szíves ne tegye, mert alább jön még ugyanerre a célra sokkal alkalmasabb eszköz.

```
BANK 9 PEEK 15000 TO numvar
```
Elhelyezi a 9-es bank 15001-edik memóriarekeszének tartalmát a numvar nevű numerikus változóban. Mivel a bank 16K hosszú, ezért a PEEK paramétereként legfeljebb 16383 adható meg. Hát igen, nem szép megoldás a 21. században, hogy nem egy függvény visszatérési értékével adják át a művelet eredményét, de egyrészt nyelvtől függ, hogy mikor történik valóban érték vagy cím szerinti átadás, másrészt 8 biten azzal gazdálkodunk, amit találunk.

```
BANK 11 COPY TO 9
```
Az egész 16K területet átmásolja a BANK 11-ből a 9-be. Gyorsabb, mint FOR és BANK … POKE utasításokkal, hidd el.

```
BANK 11 COPY 4096,256
TO 9,8192
```
Átmásol 256 bájtot a 11-es bank 4097-edik (nullától számozunk, ugye) memóriarekeszétől a 9-es bankba, annak 8193. (azaz 8192-es számú) rekeszétől kezdődően.

```
BANK 12 ERASE 255
```
Teletölti a 12-es sorszámú (nem számít, de sorrendben a 13-ik) bankot 255-ös decimális értékekkel. Ha az értéket nem adod meg, akkor 0-t használ.

```
BANK 12 ERASE 9999,1024, 255
```
Opcionálisan két paraméterrel bővíthető a BANK … ERASE (eggyel nem!). Példánk feltölt a 12-es sorszámú bank 10000. byte-jától kezdve 1024 memóriarekeszt a 255-ös értékkel.

```
BANK 47 CLEAR
```
Eddig nem volt róla szó, de ha egy bank tartalmát bármilyen utasítással (BANK PEEK/POKE/COPY/ERASE/LOAD) megváltoztatod, akkor a rendszer foglaltként jelöli azt. Ha nincs rá szükséged, akkor érdemes felszabadítani, hogy a rendszer saját céljaira használhassa.

```
BANK NEW newbankvar
```
Lefoglalja a következő szabad bankot és sorszámát eltárolja a paraméterként megadott newbankvar változóban. Ugyan a fentebb említett parancsok lefoglalják az érintett bankokat, de olykor a BANK NEW jól jöhet, ha fut egy rezidens program, melynek memóriaigényeit nem ismerjük.

Összegzésként a következőket állapíthattam meg:

- A NextBASIC lehetővé teszi, hogy akár byte-onként, de legfeljebb 16K-s blokkonként elérd a memória egészét, eltekintve a rendszer által fixáltan használt részektől.
- A bankok sorrendjének semmi köze a 48K-s Spectrum címkiosztásához, attól független.
- A NextZXOS nem teszi lehetővé, hogy összevissza kioszd a bankokat az általad jónak tartott címtartományokba, mert ezt a rezidens programok és az OS nem tudnák hatékonyan követni, valamint a kompatibilitásnak is annyi lenne. A kiosztás fix és a legtöbb hozzáférés blokkmásolásból áll, de ne aggódj, nem lesz ez olyan lassú.
- Amit BASIC-ben lefoglalsz, arról a NextZXOS tudni fog, nem rondít bele és a rezidens programok is tiszteletben tudják tartani (ha megfelelően írják meg azokat)
- A képernyőtartalom és a sprite-ok minden módban egyszerűen menthetőek (részletesen lásd majd a cikk folytatásában)
- Az ESX DOS fájl műveletek a megadott terület tartalmát (pl. screen vagy tömb) is tiszteletben tartva képesek dolgozni, igazi 8-bites Kánaánt nyitva a 48K-ból kiszabadult programozók számára.

A következő részekben még tovább vesézgetjük a NEXT BASIC újdonságait, a fájlkezelő parancsokat és egyéb grafikai lehetőségeket.

Egri Imre (Zimi)
Folytatjuk…

sinclair
ZX Spectrum Next

HOGYAN ÍRJUNK JÁTÉKOT ZX SPECTRUMRA - 9. RÉSZ

Háttér grafika

Blokkok megjelenítése

Tételezzük fel, hogy szeretnénk írni egy egyképernyős labirintus játékot. Ehhez meg kell jeleníteni a falakat, amelyek között bolyongani fog a játékos sprite-ja. Erre a legjobb módszer, ha létrehozunk egy blokk-táblát, amelyet soronként kirajzolunk a képernyőre. Ahogy haladunk a táblában, kiszámoljuk az aktuális képernyő címet, és kirajzoljuk azt a karaktert, amelyre a táblázat aktuális helyén lévő cím mutat.

Kezdjük is a karakter megjelenítő rutinnal! A sprite kirajzoló rutinnal ellentétben, itt karakterpozíciókkal kell dolgoznunk - szerencsére, ugyanis könnyebb a karakter pozíciók címeit meghatározni, mint az egyes pixelekét.

A Spectrum képernyőjén 24 függőleges és 32 vízszintes karakter pozíció található. Ennek megfelelően a koordinátáink (0,0) és (23,31) közé fognak esni. A 0-7-ig terjedő sorok a felső szegmensbe tartoznak, 8-15-ig a középsőbe, 16-23-ig pedig az alsó részhez. Szerencsére a szegmensekhez tartozó képernyő cím magas helyiértékű bájtja 8-cal nő szegmensenként, így ha a függőleges sorszámon végrehajtunk egy AND 24 utasítást, máris megkapjuk a hozzá tartozó képernyő szegmens kezdőcímét! Adjunk hozzá ehhez 64-et (ami a Spectrum képernyőjének a kezdőcíme), és máris megvan a címünk magas helyiértékű bájtja. Ezek után meg kell keresnünk a megfelelő karaktercellát az aktuális szegmensen belül, úgyhogy ismételten vegyük a függőleges sorszámot, és alkalmazzunk rá AND 7-et, hogy megkapjuk, a szegmens melyik soráról van szó a hétből. Ezt szorozzuk meg a soronkénti karakterek számával - ami 32 -, majd adjuk hozzá a keresett cella vízszintes sorszámát, hogy megkapjuk a képernyő címhez tartozó alacsony bájtot! Lássunk erre egy példát:

```
; Visszaadja a (b, c) címen található karakter
cella címét.

chadd:
   ld   a, b          ; függőleges pozíció.
   and  24            ; melyik szegmens, 0, 1
                        vagy 2?
   add  a, 64         ; 64*256 = 16384, a
                        Spectrum képernyő
                        memória címe.
   ld   d, a          ; ez a magas bájt.
   ld   a, b          ; mi is volt a függőleges
                        pozíció?
   and  7             ; a szegmensen belül
                        melyik sor?
   rrca               ; szorzás 32-vel.
   rrca
   rrca
   ld   e, a          ; alacsony bájt.
   ld   a, c          ; y koordinátát is
                        hozzáadjuk.
   add  a, e          ; hozzáadjuk az alacsony
                        bájthoz.
   ld   e, a          ; a de-ben a keresett
                        képernyőcím.
   ret
```

Miután megvan a képernyő címünk, már egy egyszerű történet a karakter kirakása a képernyőre. Mindaddig, amíg nem lépünk át karaktercella határt, a következő sor 256 bájttal követi az előzőt, szóval elegendő a magas bájt értékét növelnünk, hogy a következő sorra ugorjunk.

```
; A hl címen lévő karakter megjelenítése a (b,
c) koordinátán.

char:
   call chadd         ; megkeressük a cél
                        képernyő címet.
   ld   b, 8          ; 8 pixel magas.
char0:
   ld   a, (hl)       ; forrás grafika.
   ld   (de), a       ; átvitel a képernyőre.
   inc  hl            ; következő adat.
   inc  d             ; következő pixelsor.
   djnz char0         ; ismétlés
   ret
```

A blokkok kiszínezését illetően, az egyszerű attribútumütközés-detektálásáról szóló részben már érintettük a témát. Az *atadd* rutin megadja egy attribútumcella címét a (b, c) karakterpozícióhoz. Végezetül meg kell határoznunk, hogy melyik blokkot jelenítsük meg az egyes pozíciókban. Mondjuk 3 féle blokkra lesz szükségünk a pályához: legyen a 0-ás blokk az üres hely, 1-es a fal és a 2-es egy kulcs. A grafikát és az attribútumértékeket két külön táblázatban helyezzük el, de azonos sorrendben:

```
blocks  equ $
; block 0 = üres karakter.
   defb 0,0,0,0,0,0,0,0

; block 1 = fal.
   defb 1,1,1,255,16,16,16,255

; block 2 = kulcs.
   defb 6,9,9,14,16,32,80,32

attrs   equ $
; block 0 = üres hely.
   defb 71

; block 1 = fal.
   defb 22

; block 2 = kulcs.
   defb 70
```

Ahogy haladunk végig a 24 sorból és 32 oszlopból álló labirintus táblázatunkon, elhelyezzük a blokk sorszámát az akkumulátorban, majd meghívjuk az *fblock* és *fattr* függvényeket, hogy kiderítsük a megfelelő grafika és attribútumérték címét.

```
; Grafikus cella meghatározása.

fblock:
   rlca              ; nyolccal szorozzuk a
                       blokk sorszámát.
   rlca
   rlca
   ld    e, a        ; eltolás a kezdőcímhez
                       képest az e-be.
   ld    d, 0        ; magas bájt 0.
   ld    hl, blocks  ; karakterblokkok
                       kezdőcíme.
   add   hl, de      ; a keresett block címe.
   ret
```

```
; Attribútum meghatározása.

fattr:
   ld    e, a        ; eltolás a kezdőcímhez
                       képest az e-be.
   ld    d, 0        ; magas bájt 0.
   ld    hl, attrs   ; attribútumblokkok
                       kezdőcíme.
   add   hl, de      ; a keresett block címe.
   ret
```

Ezzel a megoldással a labirintusunk tárolásához 1 bájt RAM-ra van szükség karaktercellánként. Egy 32x16 blokkos játéktérhez minden képernyő 512 bájt memóriát foglal el. Így a memória elegendő egy 20 képernyőből álló platformer játékhoz, mint például a Manic Miner, de ha száz, vagy annál is több képernyőt szeretnénk összerakni, akkor érdemes megfontolni nagyobb méretű blokkok használatát, hogy egyetlen képernyő kevesebb memóriát emésszen fel. Amennyiben inkább 16x16 pixeles blokkok mellett döntünk a példában lévő 8x8-as helyett, az egyes képernyőket tartalmazó táblázatok mindössze 128 bájtot foglalnak majd a Spectrum memóriájából.

Fordította:
Tanács Imre (Kapitány)
Folytatjuk…

HARDVER SIMOGATÓ - ZX81

CHROMA

A Chroma interfész egy jó kis több-az-egyben eszköz ZX81 felhasználóknak, mely valójában már 2014 október 20-án napvilágot látott és Paul Farrow nevéhez fűződik. Az elsődleges funkciója mindenképp az, hogy SCART csatlakozón keresztül köthessük a kis fekete szörnyet a TV-re vagy monitorra, így élvezve a tűéles kép előnyeit. A bővítő portra csatlakozik, így nem szükséges kis barátunkat macerálni, valamint egy teljesen szabványos SCART kábel szükségeltetik.

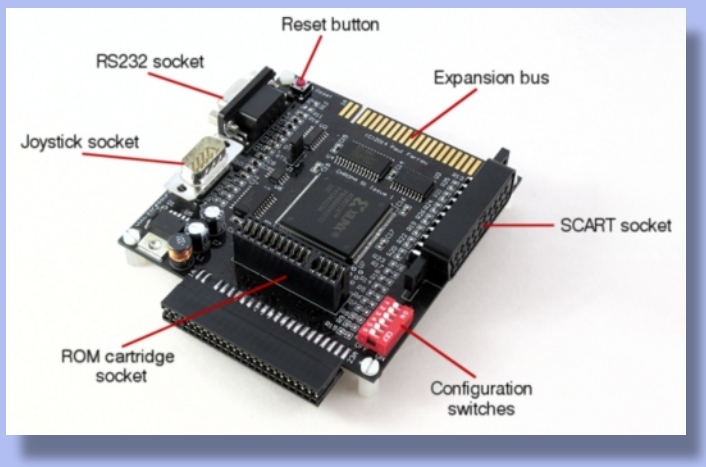
Egy másik lenyűgöző újdonságával megszínesíthetjük ZX81-ünk életét. Két színes üzemmódja létezik, mindkettő 15 színt tesz elérhetővé. Az egyik mód - a ZX Spectrumhoz hasonlóan - attribútum fájl jellegű megoldást biztosít, a másik módban egy színkereső (colour lookup) táblázattal mind a 128 különböző karakter (64 normál és 64 inverz) színét leképezi, amit csak a ZX81 képes megjeleníteni.
Ez utóbbi mód teszi lehetővé a meglévő játékok utólagos színezését.
Ezek mellett még egy csomó jópofa lehetőséggel vértezhetjük fel kedvencünket:

- 16K RAM bővítő
- 8K RAM, az alábbi cím tartományban található: $2000-$3FFF, ami a felhasználói karakterek definiálását teszi lehetővé (UDG)
- 16K RAM bővítés, ami a 48K-64K részben van és adattároláshoz használható
- Attribútum színes üzemmód: 15 ink és 15 paper szín karakter pozíciónként
- Karakter kód színes üzemmód: 15 ink és 15 paper szín soronként minden normál és inverz karakterhez
- LOAD/SAVE hang a SCART csatlakozón keresztül a TV-n
- Quicksilva Character Board emuláció, amivel azok a Quicksilva játékok futtathatóak, amelyek támogatják az eredetit
- WRX nagyfelbontású grafika, melyet az eszköz belső RAM-ja támogat
- RS232 csatlakozó, számos soros eszközhöz, például nyomtatóhoz vagy PC-hez való csatlakozást tesz lehetővé
- ZX Interface 2 jellegű ROM kártya csatlakozó, így a ZX81 ROM teljesen felülírható
- Cursor joystick csatlakozó, támogatja az auto-fire joystick-kokat
- Reset gomb
- Aranyozott továbbmenő élcsatlakozó
- Rengeteg konfigurációs lehetőség

Konfiguráció

1. kapcsoló: ON = Onboard 16K RAM [$4000-$7FFF] engedélyezve
2. kapcsoló: ON = WRX grafikus támogatás a fedélzeti RAM számára.
3. kapcsoló: ON = Onboard 8K RAM [$2000-$3FFF] engedélyezve
4. kapcsoló: ON = a Quicksilva karaktertábla emuláció engedélyezve
5. kapcsoló: ON = be van kapcsolva az RS232 kapcsolat
6. kapcsoló: ON = Színes mód engedélyezve / további 16K RAM [$C000-$FFFF].

A Chroma interfész 10x10 cm méretű. A kezelőfelület kulcsfontosságú elemei a következő képen láthatóak:



Kompatibilitás

A ZX81 két ROM verzióval került kiadásra. Az első kiadásnak hírhedt hibája volt, hogy az SQR 0,25 a 0,535 helyett 1,3591409 eredményt adott. A hibát 3 szükségtelen utasítás okozza. Ahelyett, hogy eldobták volna az összes legyártott hibás ROM-ot, Sinclairék egy kis hugi lapot készítettek, melyet ráforrasztottak a Z80 CPU-ra, ez elfogadta az egyik hibás utasítás végrehajtását és megváltoztatta azt

úgy, hogy az utasítások már nem befolyásolták a lebegőpontos számításokat. Ennek köszönhetően elkapja és felülírja a ROM kártya megfelelő helyén lévő utasításokat is. Emiatt korlátozza a ROM kártyáról futtatható programokat, amikor ilyen hardverjavító lapkával patkolt ZX81-hez csatlakoztatjuk.

Sinclair ZX 16K RAM bővítő
A Sinclair 16K RAM bővítő kompatibilis a Chroma interfésszel, de megköveteli, hogy a 16K-os RAM-ot kapcsoljuk ki az 1. konfigurációs kapcsoló kikapcsolásával. Ha a kapcsoló be van kapcsolva, akkor felülírja a Sinclair 16K RAM memóriájának hozzáférését.

Sinclair ZX Printer
A Sinclair ZX nyomtató teljesen kompatibilis a Chroma interfésszel. A ZX nyomtatót a Chroma interfész mögé kell csatlakoztatni.

ZXpand
Charlie Robson ZXpand SD kártya interfésze is teljesen kompatibilis a Chroma interfésszel. A ZXpand 8K ROM-ot és 32K RAM-ot tartalmaz, ezeket a Chroma interfész megfelelő memóriája felülírja, ha engedélyezve van. A ZXpand és a Chroma RAM-jának kombinálásával a 16K-64K-os tartományban egy folytonos RAM memóriát lehet elérni.

64K RAM bővítők
A 64K RAM bővítők kompatibilisek a Chroma interfésszel, és szintén a Chroma mögé kell kapcsolódniuk. Ha bármelyik Chroma RAM vezérlő kapcsolója be van kapcsolva, akkor a Chroma RAM felülírja a hozzáférést a 64K RAM megfelelő memóriaterületéhez.

Belső 4K grafikai ROM
A Kayde Electronic Systems és a dk'tronics is kiadott egy 4K Graphics ROM kártyát, amelyet a ZX81-be kellett beépíteni, és hét alternatív karakterkészletet biztosított. A Chroma interfész helyesen jeleníti meg az alternatív karakterkészleteket, ha van beépítve a ZX81-be, de a ROM kártya produkálhat némi inkompatibilitást.

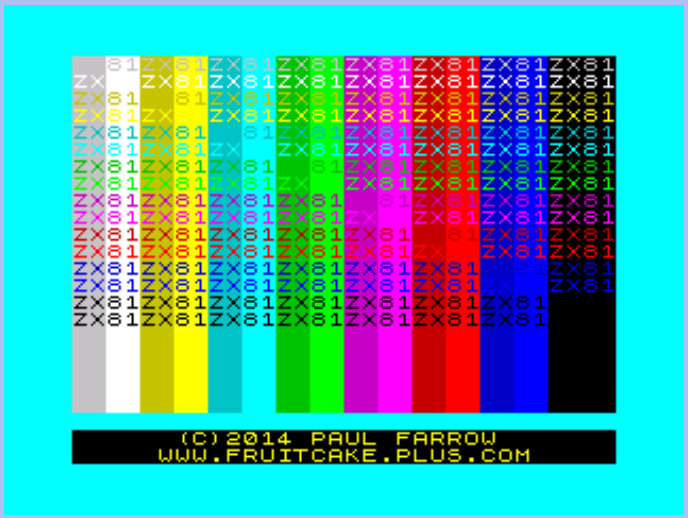
Memotech billentyűzet
A Memotech által gyártott billentyűzet interfész a Chroma interfésszel egyidejűleg csatlakoztatható, és megfelelően működik, azonban a billentyűzet-interfész áramköri felépítése miatt a Chroma interfész joystick-aljzatát nem szabad használni, amíg a billentyűzet interfész csatlakoztatva van, egyébként a Chroma interfész meghibásodását okozhatja.

TV
A Panasonic televíziók több típusa úgy tűnik nem felel meg a SCART-specifikációnak, és a kép balra vagy balra/jobbra ugrál. Úgy tűnik, hogy a TV arra vár, hogy a videó szinkronjel színes burstjelet tartalmazzon, de ez csak RF modulált jel esetén szükséges.

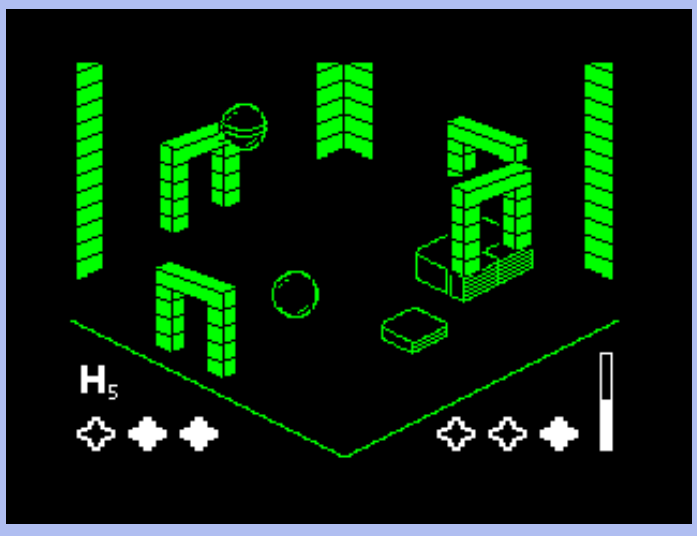
További információk a www.Fruitcake.plus.com oldalon érhetők el (További programok).

Kardos Balázs (Balee)

Néhány minta képernyő:



Attribútum teszt program (Paul Farrow)



Against the Elements (Paul Farrow)

JÁTÉKÚJDONSÁGOK

2018-AS ZX SPECTRUM MEGJELENÉSEK

2018. JANUÁR

Apulija-13 v2.0
(Alessandro Grussu, 48K+128K, MOD) a 2013-as játék továbbfejlesztett verziója

Astrosmash! ZX
(AMC Games, 48K+AY, lövöldözős)

Bobby Carrot
(diver4d/Quiet/Kyv/Zorba, 48K+128K, logikai)

Crazy Kong City – Episode 2: Saving Kong
(Gabriele Amore, 48K/128K, akció) a Bumfun Gaming kiadta az első résszel egy kazettán, Saving Kong címmel £10.00 (+posta) áron

Doctor Who - Surrender Time
(Errazking/ThEpOpE, 128K, akció)

Extruder
(Rui Martins, 48K, logikai)

Gandalf
(Cristian M. Gonzalez, 128K, platform)

Gimmick! Yumetaro's Odyssey
(greenwebsevilla, 128K, platform) kazettán kiadta a Matranet, €8.75(+posta) a normál, €25.00 az enhanced verziója

Harbinger 2 - The Void
(APSIS, 128K, akció)

In Nihilum Reverteris
(Hooy-Program, 128K, szöveges kaland)

Jet Set 40-40
(jswmm.co.uk, 48K, Jet Set Willy MOD)

Mighty Final Fight
(Sanchez crew, 128K, verekedős)

Mike The Guitar
(Sebastian Braunert & Uwe Geiken, 128K, platform)

Ninja Gaiden Shadow Warriors
(Jerri & DaRkHoRaCe & diver4d, 128K, verekedős)

Parachute
(Miguetelo, 48K/128K, akció) a Bumfun Gaming kiadta kazettán £10.00 (+posta) áron

Roust
(Allan Turvey, 128K, platform) ingyenes 48K demója van és $2.50-ért letölthető 128K teljes verziója az itch.io-ról

Sqij 2018
(Tardis Remakes, 48K, akció) az 1987-es, legendásan rossz játék C64 változatának remékje, kazettán kiadta a Psytronik £6.99(+posta)-ért

Stepping Stones
(SinDiKat, 48K/128K, logikai)

The Adventures of Jane Jelly - The Egg Diamond
(Jaime Grilo, 48K/128K, platform) a Bumfun Gaming kiadta kazettán £10.00 (+posta) áron

Vindius
(Ancient Bits, 48K, szöveges kaland)

Zxombies: Dead Flesh
(James Broad, 48K/128K, akció)

2018. FEBRUÁR

Castle of Sorrow
(ZXMan48K, 48K, platform)

Dungeon Raiders
(Payndz, 48K, akció)

Eurostriker
(Valdir, 48K, sport)

Fantasy Zone - Escape from the Pyramid
(greenwebsevilla, 48K+128K, MOD) a 2016-os 48K-s verzió továbbfejlesztése, kazettán kiadta a Matranet, €8.75(+posta) a normál, €25.00 az enhanced verziója

Impossabubble
(Dave Clarke, 48K/128K, arcade/platform) $1.49-ért letölthető az itch.io-ról

Invasores Aquáticos 2.0
(EspectroTeam, 48K, MOD) egy 1985-ös BASIC játék remékje

Scuttlebutt
(Eweguo, 48K+AY, akció)

2018. MÁRCIUS

Go Race!
(MonkZy, 48K, autóverseny) vacak játékok versenyére készült, de annyira nem rossz

Knights
(Darryl Sloan, 48K, táblás)

Manic Miner - Highscore Challenge
(Andy Ford, 48K, Manic Miner MOD)

Pushbot
(Dave Hughes, 48K, logikai) teljes játéknak tűnik, de a szerző szerint csak demo

Steamed Hams
(PROSM Software, 48K, szöveges kaland)

Steel Jeeg
(Francesco Forte, 48K/128K, labirintus)

2018. ÁPRILIS

Aeon
(Sunteam, 48K/128K, platform)

Gem Slider
(Noniewicz.com/BCF, 48K+AY, logikai)

Ninjakul in the AUIC Temple
(greenwebsevilla, 128K, arcade/platform) kazettán kiadta a Matranet, €12.50(+posta) az ára
O.P.Z.
(Dave Hughes, 48K, platform)
Pixel Quest Zero
Einar Saukas, 48K+AY, logikai) az Espectro magazin 3. számának kazettamellékletén szerepel
ROVR
(Paul Jenkinson, 48K/128K, akció)
Tea-Leaf Ted
(Jaime Grilo, 48K, létrás)
Thoroughly Modern Willy
(Bob Fossil, 48K, Manic Miner MOD)
Varina
(Espectroteam, 48K, akció) az Espectro magazin 3. számának kazettamellékletén szerepel

2018. MÁJUS

Hibernated-1
(Stefan Vogt, 48K, szöveges kaland) kiadta a poly.play, kazettán a CPC verzióval együtt €20.00(+posta), 3" lemezen €25.00(+posta), multiplatform gyűjtői kiadásban pedig €33.00(+posta)
Nixy the Glade Sprite
(Bubblesoft, 128K, platform) kazettán kiadta a Monument Microgames
Quahappy
(Jaime Grilo, 48K/128K, labirintus)
Rubicon
(Rucksack Games, 48K/128K, akció)

2018. JÚNIUS

Lost in Worlds
(Nihirash, 48K, platform)
Rogue
(g0blinish, 48K, roguelike)
Tank-1990
(Dwa83, 128K, lövöldözős)

2018. JÚLIUS

4 Knights
(g0blinish, 48K, logikai)
Dizzy and the Mystical Letter
(Hippiman, 128K+TR-DOS, akció/kaland)
Doom Pit
(Monument Microgames, 128K, létrás) a Death Pittel egy duplakazettás kiadványt alkot, £25.00 (postával) áron megvásárolható
Inertia
(g0blinish, 48K+AY, logikai)
Prospector
(AMC Games, 48K, platform)
Spych
(g0blinish, 48K, logikai)
The Big Sleaze 2.5
(Mark Hardisty, 128K, szöveges kaland)
The World War Simulator: Part II
(Retrobytes, 48K, akció)

2018. AUGUSZTUS

Magic
(g0blinish, 48K, logikai)
Max Pickles Part I: The Haunted Castle
(World XXI Soft, 48K, platform)
Slither
(g0blinish, 48K, logikai)
Tumult: Phase 1
(Bumfun Gaming, 48K, lövöldözős) £10.00(+posta) áron megvásárolható

2018. SZEPTEMBER

All Hallows
(Rise of the Pumpkin) (Rucksack Games, 48K/128K, platform)
Behind Closed Doors 7
(Zenobi, 48K, szöveges kaland)
Grid I
(g0blinish, 48K+AY, logikai)
Max Pickles Part II: The Mine of Doom
(World XXI Soft, 48K, platform)
Max Pickles Part III: The Price of Power
(World XXI Soft, 48K, platform)
Pipes!
(g0blinish, 48K+AY, logikai)
Pooper Scooper
(The Death Squad, 48K+AY, akció)
Ramsbottom Smith and the Quest for the Yellow Spheroid
(Zenobi, 48K, szöveges kaland)
Robots Rumble
(Miguetelo, 48K/128K, akció)
Super Serif Bros
(Robin Allen, 48K, logikai)
Vradark's Sphere
(Sanchez crew, 128K, roguelike) van ingyenes demója, a teljes verzió pedig €1.00-ért tölthető le a zxonline.net-ről

2018. OKTÓBER

Chibi Akuma's - Episode 1: Invasion!
(Akuyou, 128K, lövöldözős) kiadta a poly.play, 3" lemezen €25.00(+posta), 3,5" és 5.25" lemezen pedig €25.00(+posta) az ára
Gandalf Deluxe
(Cristian M. Gonzalez, 128K, platform) a januári

Gandalf feltuningolt változata, kazettán kiadta a Matranet, €8.75(+posta) a normál, €25.00 az enhanced verziója[/i]

Manic Mixup
(Andy Ford & Ian Rushforth, 48K, Manic Miner MOD)

Unhallowed
(Blerkotronic Software, 128K, szöveges kaland)

2018. NOVEMBER

Astronaut Labyrinth
(Jaime Grilo, 48K/128K, labirintus)

Clicky Clisk's Dungeon of Hand Drawn Computer Graphics
(Chip-Fork, 48K, kaland)

Depth Charge
(oblo, 48K, akció)

Escape
(oblo, 48K, akció)

Maze Death Rally-X
(Tom Dalby, 48K+AY, labirintus/autóverseny)

Old Tower
(RetroSouls, 48K+128K, akció)

Quadron
(Cosmium, 48K, lövöldözős) $4.99-ért tölthető le az itch.io-ról

Quest
(g0blinish, 48K+TR-DOS, kaland)

The Eggsterminator
(The Death Squad, 48K+AY, akció)

TrolleyMania
(Gareth Pitchford, 48K, szöveges kaland)

2018. DECEMBER

Agatha Christie's Parrot+
(Jason J. Railton, 48K, akció) a Woot! 2018 játékgyűjteményén

Bean Brothers
(Stonechat Productions, 48K/128K, platform) a Woot! 2018 játékgyűjteményén

Buzzsaw+
(Dim Sun Edition) (Jason J. Railton, 48K+AY, logikai) a Woot! 2018 játékgyűjteményén

Elon M. with a Jetpack
(Rafa Vico, 48K+AY, lövöldözős)

Fillomania
(g0blinish, 48K+AY, logikai)

Log Cabin Dizzy
(Verm-V, 128K+TR-DOS, akció/kaland)

Manic Scroller
(Mark Woodmass & Daniel Gromann, 48K, Manic Miner MOD)

Mini Explorer XXXI
(Rafa Vico, 48K+AY, lövöldözős)

Mister Kung-Fu
(Uprising Games, 48K+AY, verekedős)

Nohzdyve
(Tuckersoft, 48K, akció)

O-Eyes
(oblo, 48K, logikai)

O-Puzz Attack!!
(oblo, 48K, logikai)

OctuKitty
(Ultranarwhal, 48K/128K, platform)

Pitman
(g0blinish, 48K+TR-DOS, logikai)

Rodmän
(Misfit, 48K+AY, labirintus) a thefuturewas8bit.com adta ki, £24.99(+posta) az ára a három kazettát tartalmazó kiadványnak, ingyenesen csak a demó tölthető le

Santa's Strange Dream
(textvoyage, 48K, lövöldözős) a Woot! 2018 játékgyűjteményén

Super Moritz
(Sebastian Braunert, 48K/128K, lövöldözős) a Woot! 2018 játékgyűjteményén

Survival ZX
(Mr Rancio, 128K, akció)

Your crackers, m'lord!
(Bob Fossil, 48K, logikai) a Woot! 2018 Játékgyűjteményén

Megjegyzés:
Világoskékkel (spectrumosan cyannal) a csak pénzért beszerezhető játékokat jeleztem.

Mezei Róbert (m/zx)