IK Plus TV-Computer (2019)

Ez az írás többek kérésére született, arra a kérdésre válaszolva, hogyan hoztuk létre az Interkarate+ játék 2019-es változatát TV-Computerre.

A teljesség igényére törekedve próbáljuk felidézni a program kezdeteitől a kiadásáig történt főbb eseményeket, hogy tanulságul szolgáljon nem csak másoknak, hanem magunknak is. Lássuk hát, hogyan jutottunk el a játék megalapozásától a 2019-es TVC játékfejlesztői verseny első helyezéséig.

A TVC verzió címképe

Előzmények

1987-ben Major Tamás és Gábor átírta az IK+ ZX Spectrumos verzióját TVC-re. Ehhez a program képernyő kezelését, az irányító gombok működését és a hangokat kiadó részeket kellett módosítani. A feladat nehéz volt, de nem lehetetlen: néhány nap alatt megvoltak vele. A program sebessége hagyott némi kívánnivalót maga után, de ezt a kényelmetlenséget áthidalta az élmény: két játékos tudott játszani egy másik platformról áthozott játékkal, a munka színvonalát pedig emelte a képernyőn három helyen is változó grafikus mód: a pontszám kijelzés 4, a háttérkép 16, a karakterek ismét 4 színű üzemmódban voltak megjelenítve. Mindezeken felül a játék hónapokon át a magyar toplistákon szerepelt (egyszer holtversenyben a szintén Tamás által átírt Elite-tel).

IK+ TVC verzió, átirat ZX Spectrum-ról

Persze az ember mindig többet akar. Alig készült el az átirat, máris arról ábrándoztunk, mekkora dolog lenne a Commodore 64-es verziót átírni a gépre! Persze tisztában voltunk vele, hogy kis eséllyel lehetett volna egy ilyen projektet befejezni.

A projekt

Az álmok viszont makacs dolgok. Valamelyest eltávolodnak ugyan tőlünk, de előbb-utóbb kikényszerítik, hogy megvalósuljanak. És minél később, annál többe kerülnek. A miénk (mivel harmincöt évről beszélünk) jó sokba került. De ne vágjunk a dolgok elébe, mert a történet egész másképp indult... :)

A 2018 végén úgy döntöttem, szeretnék egy bizonyos játékot megírni TVC-re. Alig pár nappal később Bertók Zsolt (Bery) a TVC-s facebook csoportban játékfejlesztői versenyt hirdetett, aminek hatására Tamás úgy döntött, ő is beszáll a versenybe, “hivatalosan” jelezve, hogy együtt fogunk valamit készíteni. Ez azt jelentette, basic helyett egy full assembly játékot tehetünk le az asztalra, amivel nevezni (akár nyerni is) lehet.

Tamás elkezdte írni a DevTool nevű TVC-s fejlesztő rendszert (erről még lesz szó), eközben én a képernyőterveket, sprite-okat, hangokat készítettem. Mivel az animáció szerkesztő és lejátszó még készült, belefogtam egy teljesen *másik* játék grafikájába és pályáiba *is*. Mikor Tamás látta ennek az állapotát (kvázi kész lett), úgy döntöttünk, hogy akkor csináljuk meg inkább azt.

Míg készült Tool, közben (pihenésképpen) más platformokról hoztam át képeket TVC-re. Ehhez egy komplett “színkonverziós-pepitázós” munkafolyamatot dolgoztam ki, amit be is mutattam a TVC csoportban és az ehhez használt “fejlesztő eszközöket” szintén közreadtam.

De az “áthozott” képek között volt egy, amit rajtam kívül még senki nem látott.

Az C64-es IK+ játékbeli képe.

Persze a színek és a felbontás miatt alaposan át kellett dolgoznom. Szétbontottam rétegekre és az eredetileg 160 pixel széles képet 128 pixelbe sűrítettem, úgy átrajzolva, hogy lehetőleg senkinek ne tűnjön fel, hogy a kép jóval keskenyebb, mint az eredeti.

Mindezek után áttoltam Tamásnak is a képet, hogy mit szól hozzá.

Nem volt nehéz kitalálni az eredményt: De jó lenne átírni ezt is! Innen még futottunk pár udvariassági kört, hogy “így-meg úgy”, de mindketten tudtuk, mostantól mi a csapásirány.

Az Interkarate Plus, C64-hez hasonló grafikával!

Kezdetek

Az alapállású karatékát már áthoztam TVC-re és mivel ez az egyik legnagyobb méretű sprite, ebből tettünk ki hármat és jobbra-balra mozgatva őket, a kirakó-mozgató rutint átírogatva próbáltunk minél nagyobb sebességet elérni. Több dolgot is szem előtt tartottunk: a karakterek legyenek átszínezhetők (ruha, öv, haj), lehessen tükrözni őket, maszkolhatóak legyenek, minél kevesebb helyet foglaljanak a memóriában és minél gyorsabban lehessen őket kirakni (méghozzá villogásmentesen).

A sprite-ok memóriafoglalását rögtön tudtuk felezni azzal, hogy a karaktereket az alsó nyolc színre alakítva tároltuk és a képernyőre másolás közben festettük át a kívánt színekre, kihagyva a maszkolt (lyukas) pixeleket. Emellett minden fázis tömörítve van tárolva, de erről majd később.

Tehát három sprite mozgott a képernyőn, jobbra-balra, direktben, főciklusban. Ahogy fejlődött (rövidült és gyorsult) a rutin, úgy tették meg az útjukat egyre rövidebb idő alatt. Tamás folyamatosan adta a verziókat, én pedig mértem a működési idejüket, először stopperrel, majd egy sokkal egyszerűbb és hatékonyabb módszerel: több emulátort futtattam egymás mellett, ilyen módon pedig alig egy percen belül látszott, melyik gyorsabb a másikhoz képest (hogy mennyivel, az nem volt fontos, csak hogy gyorsabb, vagy nem).

Ez a tesztelés több napon át tartott, a finomítások után javarészt gyorsabb és gyorsabb lett a rutin.

Őket mozgattuk a sebességtesztek alatt jobbra-balra

Render

(Tamás)

A játékban a megjelenítésért egy komplett programmodul a felelős, amit általában “rendernek” hívnak “tudományos körökben”. Ez a programrész felelős a háttér letörléséért, az új kép kiszerkesztéséért illetve a képernyőre rajzolásáért. (Tehát nem csak a sprite rajzolás a feladata.)

Nagyon fontos kritérium volt, hogy a nagy sprite-ok használata ne menjen a játszhatóság rovására, azaz a program sebességét ne csökkentse túlságosan. Továbbá ragaszkodtunk hozzá, hogy a kép ne villogjon.

Az nyilvánvaló volt, hogy a játéktér újrarajzolása nem fog beleférni két megszakítás közben eltelt időbe. A megoldás: előbb rajzoljuk ki a képet a legfelső memória lapon és amikor a teljes kép ki van szerkesztve, egyetlen szubrutinnal másoljuk a képernyőre. Viszont a karatékák (és később a labdák-karatéka kombó) összességében kevesebb mint harmad annyi színes byte-ot jelent, mint a hátteret alkotó szürke byte-ok. De akkor miért másolnánk ki a video RAM-ba a teljes háttérképet?

Ennél gyorsabb megoldásra van szükség.

Minden sprite kirakása után tegyük be egy memória tömbbe a sprite köré írható téglalapot. (bal felső X, bal felső Y, szélesség, magasság). Tegyük bele utána az előző kép köré írható téglalapokat, majd ciklusban vizsgáljuk le, hogy találunk-e két olyan téglalapot, ami metszi egymást, és ha találunk ilyet, akkor vegyük e kettő téglalap köré írható téglalapot, és ezt az összevont két téglalapot vegyük ki a tömbből. És ismételjük meg ezt a folyamatot addig, amíg találunk egymást metsző téglalapokat.

Amikor az összevonás megtörtént, utána már nincs más dolgunk, mint végig menni az összevont téglalapokon és csak ezeket a téglalapokat alkotó memória területeket másoljuk a képre. Az előző fázis téglalapjait is bele kell vonni ebbe a folyamatba, hogy ha azokhoz képest az új képen elmozdulás történt, akkor íródjon felül az előző fázis területe is. Így elkerülhető, hogy a video RAM-ban “szemétként” benne maradjon az előző fázis maradéka.

A sprite kirakás sebesség optimalizációja, illetve a képernyőre részlegesen kimásolt háttérbuffer tartalom jelentős sebesség növekedést eredményez, ami igencsak kihat a játékélményre.

A fenti folyamat algoritmusának lekódolása sok időt emésztett fel, többször is újra lettek írva részei, de az eredmény - úgy gondoljuk - magáért beszél.

Animációk

(Misi)

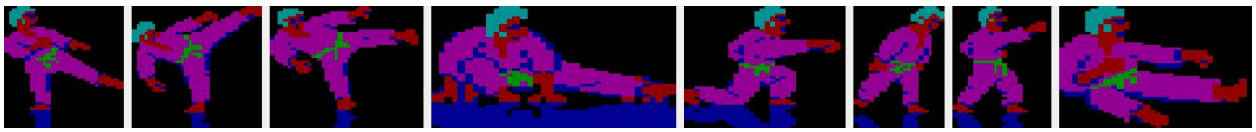
Miközben a render tesztelési folyamatot végeztük, folyamatosan emeltem ki a C64-es verzióból a karakter animációs fázisokat.

Ez persze nem úgy ment ám, hogy rá lett eresztve valamilyen sprite kereső a C64 memóriájára, ami szépen kimentette nekünk a kockákat (de szép is lett volna), már csak azért sem, mert a fázisok a C64-es verzióban tömörítve vannak(!) Most lehetne azt mondani, hogy miért nem írtunk egy programot, ami a játék dekompresszorát használva kitömörít mindent szép sorjában, de mivel csak ketten voltunk, nem volt kapacitásunk ilyesmire. Maradt a favágó módszer: emulátorból mentettem le képernyőket a mozgások fázisaival.

Ilyen képekből lettek kivágva az animációs fázisok

Persze csak olyan képeket tudtunk használni, amiken a karakterek nem takarták egymást (el lehet képzelni hány mentett képből kellet ehhez válogatni miközben arra is kellett ügyelni, hogy meglévő fázisokat lehetőleg ne mentsünk ki még egyszer).

Ezekből a mentett képekből vágtam ki a mozgásfázisokat, amiket átszíneztem az alsó 8 színre, majd (hogy át lehessen látni a munkát), 8x8-as munka mátrixba rendeztem (ez táblázat a vártnál gyorsabban elfogyott, ezért nyitnom kellett egy másikat is).

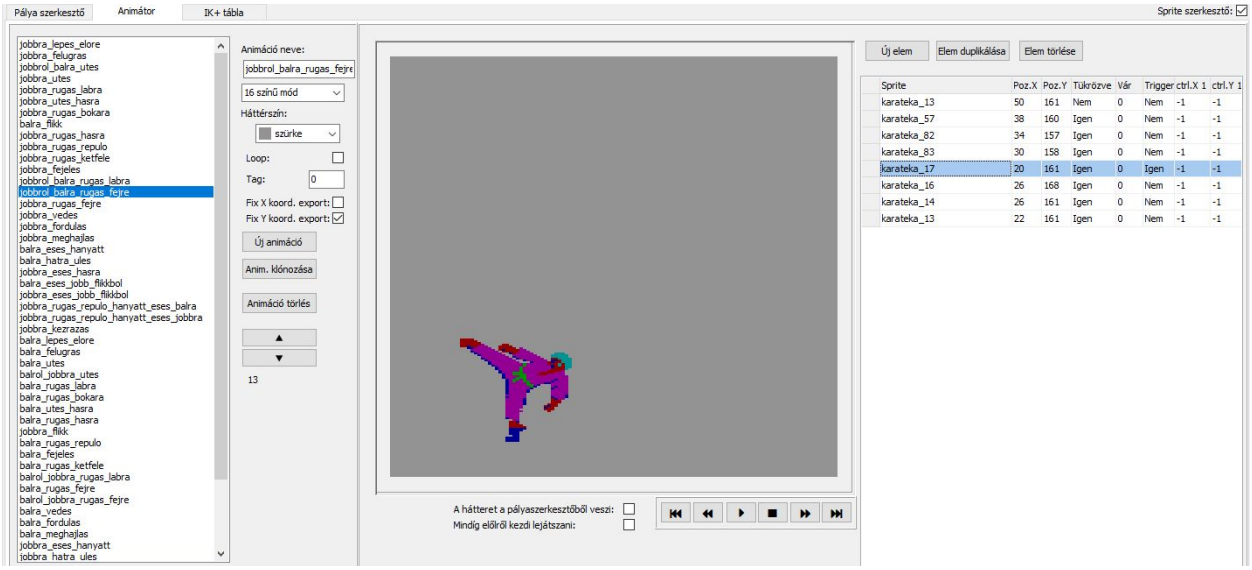


Sprite-ok az alsó 8 színre átfestve (palettásítva)

Ezután következett a fázisok egyesével való kimentése, majd beemelésük a DevTool-ba, ahol már a tényleges animációkat fűztem össze az animátor részben. A fázisok paraméterei nem csak a képeket tartalmazták, hanem az előző fázishoz képesti x és y eltolást, a fázisok kint létének idejét, valamint a tükrözést is. Természetesen minden animációt jobbra és balra néző helyzetben is össze kellett állítani a fenti delta eltolási értékekkel.

Mindet, egyesével.

Nem csak lépés, támadás animációk vannak, hanem védekezések, sérülések, elesések is, így ezeket ki tudtuk bővíteni olyanokkal, amelyek az eredeti játékban nincsenek is. Például módunkban áll a hátraflikkelő ellenfeleket hátba-hasba, vagy az éppen ugrásuk közben levegőben lévőket földre rúgni (egy kis meglepetés a veterán IK+ harcosoknak).



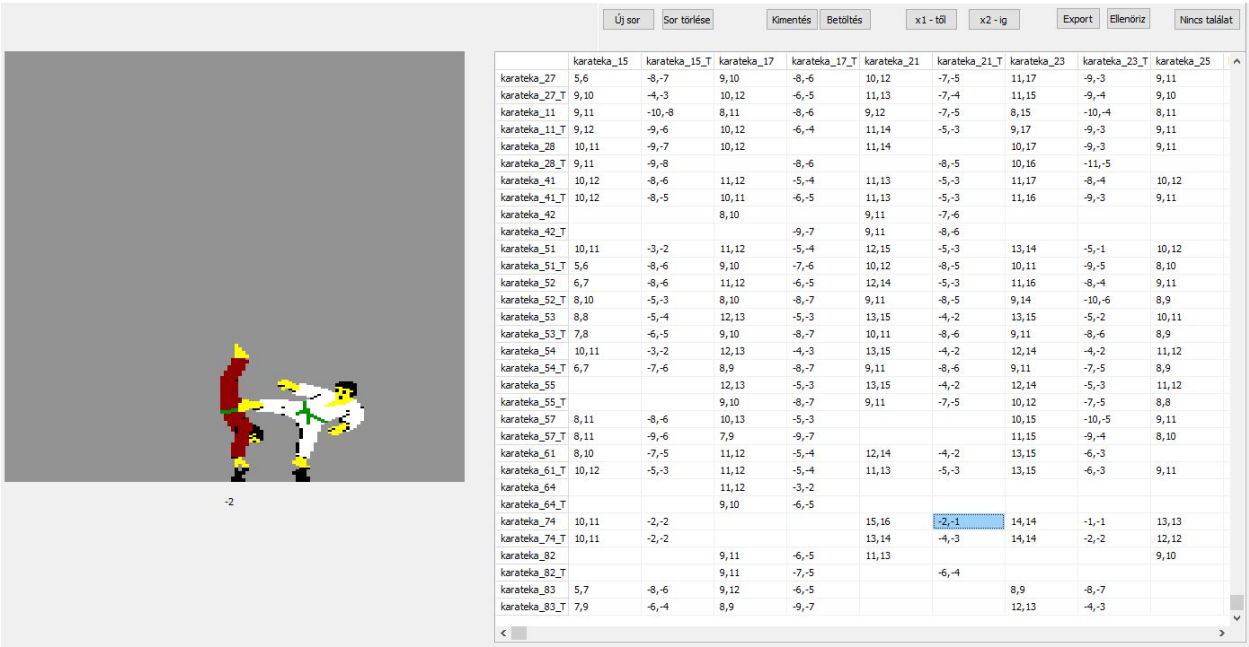
Bal oldalon az animációk, jobb oldalon a kiválasztott animáció fázisai

Az editorban a mai napig minden animáció az alapállás fázissal kezdődik, de mivel ez nagy pazarlást jelentene a memóriát tekintve, a Z80 forrásba exportált adatokból ezek a fázisok ki lettek véve (természetesen utólag, kézzel).

Minden byte-ra szükségünk volt.

Találat ellenőrzés

A találat ellenőrzést először táblázatból akartuk megoldani, de nem volt teljesen világos a módszer, ezért Tamás úgy gondolta, kicsit komolyabbra vesszük a figurát, már-már tekken szintű vizsgálattal. Készült is rá koncept, hogyan kéne definiálni az ütő és érzékelő pontokat, de ezt elég gyorsan elvetettük, így maradt a táblázat, amit sokkal egyszerűbb és gyorsabb is volt vizsgálni.



	karateka_15	karateka_15_T	karateka_17	karateka_17_T	karateka_21	karateka_21_T	karateka_23	karateka_23_T	karateka_25
karateka_27	5,6	-8,-7	9,10	-8,-6	10,12	-7,-5	11,17	-9,-3	9,11
karateka_27_T	9,10	-4,-3	10,12	-6,-5	11,13	-7,-4	11,15	-9,-4	9,10
karateka_11	9,11	-10,-8	8,11	-8,-6	9,12	-7,-5	8,15	-10,-4	8,11
karateka_11_T	9,12	-9,-6	10,12	-6,-4	11,14	-5,-3	9,17	-9,-3	9,11
karateka_28	10,11	-9,-7	10,12		11,14		10,17	-9,-3	9,11
karateka_28_T	9,11	-9,-8		-8,-6		-8,-5	10,16	-11,-5	
karateka_41	10,12	-8,-6	11,12	-5,-4	11,13	-5,-3	11,17	-8,-4	10,12
karateka_41_T	10,12	-8,-5	10,11	-6,-5	11,13	-5,-3	11,16	-9,-3	9,11
karateka_42			8,10		9,11	-7,-6			
karateka_42_T				-9,-7	9,11	-8,-6			
karateka_51	10,11	-3,-2	11,12	-5,-4	12,15	-5,-3	13,14	-5,-1	10,12
karateka_51_T	5,6	-8,-6	9,10	-7,-6	10,12	-8,-5	10,11	-9,-5	8,10
karateka_52	6,7	-8,-6	11,12	-6,-5	12,14	-5,-3	11,16	-8,-4	9,11
karateka_52_T	8,10	-5,-3	8,10	-8,-7	9,11	-8,-5	9,14	-10,-6	8,9
karateka_53	8,8	-5,-4	12,13	-5,-3	13,15	-4,-2	13,15	-5,-2	10,11
karateka_53_T	7,8	-6,-5	9,10	-8,-7	10,11	-8,-6	9,11	-8,-6	8,9
karateka_54	10,11	-3,-2	12,13	-4,-3	13,15	-4,-2	12,14	-4,-2	11,12
karateka_54_T	6,7	-7,-6	8,9	-8,-7	9,11	-8,-6	9,11	-7,-5	8,9
karateka_55			12,13	-5,-3	13,15	-4,-2	12,14	-5,-3	11,12
karateka_55_T			9,10	-8,-7	9,11	-7,-5	10,12	-7,-5	8,8
karateka_57	8,11	-8,-6	10,13	-5,-3			10,15	-10,-5	9,11
karateka_57_T	8,11	-9,-6	7,9	-9,-7			11,15	-9,-4	8,10
karateka_61	8,10	-7,-5	11,12	-5,-4	12,14	-4,-2	13,15	-6,-3	
karateka_61_T	10,12	-5,-3	11,12	-5,-4	11,13	-5,-3	13,15	-6,-3	9,11
karateka_64			11,12	-3,-2					
karateka_64_T			9,10	-6,-5					
karateka_74	10,11	-2,-2			15,16	-2,-1	14,14	-1,-1	13,13
karateka_74_T	10,11	-2,-2			13,14	-4,-3	14,14	-2,-2	12,12
karateka_82			9,11	-6,-5	11,13				9,10
karateka_82_T			9,11	-7,-5		-6,-4			
karateka_83	5,7	-8,-6	9,12	-6,-5			8,9	-8,-7	
karateka_83_T	7,9	-6,-4	8,9	-9,-7			12,13	-4,-3	

A vízszintesen a támadó technikák, a függőlegesen a támadott fázisok, a számok az eltolási értékek)

A támadó és sérülési pontok - szerencsére kivitelezni nem kellett :)

AI (mesterséges intelligencia)

(Tamás)

A játék alatt az az igazi, amikor két játékos egymás ellen küzd, de még ebben az esetben is ott a harmadik karatéka, akinek azért illik tevékenyen beszállni a küzdelembe (hacsak nem akar egy “come on” intést a Mestertől). Tehát minimum egy játékost a számítógépnek kell irányítani.

Mivel a program nem átirat, ezért az AI is teljesen saját elgondolás és ötletek alapján készült.

Amikor a számítógép egy karatékát mozgatni kezd, megvizsgálja, hogy a másik két karatéka áll-e, vagy mozgásban van-e (ha igen, akkor milyenben), Ezután a találat ellenőrzési táblázat és néhány véletlen generátoron alapuló paraméter alapján kiválasztja a számára optimálisnak ítélt mozgást, és mintha csak joystickkal irányítaná a saját emberkéjét ezt a “joystick kódot” adja át a játéknak.

Először úgy volt megcsinálva a játékosok gépi mozgása, hogy a végrehajtandó karate technikában - mintegy előre gondolkodva - vegye számításba azt is, hogy amikor a technika “berobban” akkor az ellenfél melyik mozgás fázisban lesz.

Meglepetésünkre az intelligencia hatékonysága nem volt sokkal jobb, mint amikor csak az ellenfelek pillanatnyi helyzetét veszi figyelembe, ezért ezt az “előre gondolkodós” verziót elvetettük és csak a “pillanatnyi helyzetet figyelembe vevő” változatot hagytuk meg, ami harmad akkora kód méretet eredményezett.

Lehetett volna még egy-két plusz paramétert figyelembe venni az AI döntéshozásához, de ezek a későbbiekben már nem fértek be a programkódba és nem is hozott volna jelentős változatosságot a gépi játékba.

A végső döntés eredményeképp (egy véletlen faktor figyelembevételével) vagy a megtalált támadó technikát, vagy az odaillő védekezést választja a program.

Hátránya ennek a verziónak, hogy a véletlengenerátor olyan amilyen, illetve a találati táblázatban szekvenciálisan keres az algoritmus és ha talál szerinte optimális technikát, akkor nem keres tovább, hanem azt indítja el. Ennek okán történik az, hogy ha a játékos csak egy technikát használ, akkor az AI sok esetben nem tud optimálisan dönteni.

De hát melyik karatebajnok tökéletes?

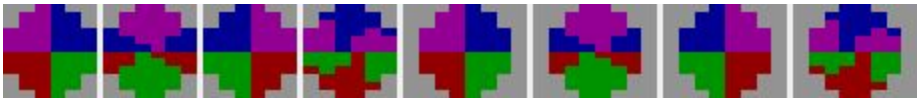
Labdák

(Misi)

A projekt elején még nem volt biztos hogy bele kerül-e a “labdázás”. Valahol tudat alatt éreztük hogy ez nem lesz egyszerű történet (de nem gondoltuk, hogy ennyire nem).

Az eredeti játékban két küzdelem-menet után labdákat kell kivédeni (ha ezt “pihentető résznek” szánták, pont az ellenkező hatást érték el vele). Minden labdáért pontokat kapunk és ha végigcsináljuk, akkor még egy rakás pontot. Mivel a maximális játékélményre törekedtünk, sok dolgot kellett megoldani. Hol is kezdjem..?

Az egyik ilyen a labdák vágása volt (amihez Tamásnak módosítania kellett a sprite kirakót), jobbról és balról is tudni kellett vágni a képernyő szélén. A másik, hogy a lehető legfinomabb mozgást akartuk elérni, minél kevesebb, de persze minél gyorsabb kóddal. Pixelenként kellett tudni mozgatni a labdákat, de nem akartunk *menet közben* biteltolást használni (a sebesség miatt). Ezért minden labda fázis kétszer van meg a memóriában, egyszer a helyén, egyszer egy pixellel eltolva. Mozgás közben hol a páros, hol a páratlan pixelre eső fázist rakjuk ki.



A labdák, ahogy a memóriában vannak (az eltolt fázisokkal együtt)

A következő dolog a szinuszos, gravitációt és pattogást modellező mozgás volt. Ehhez egy negyed ciklusnyi táblázatot használtunk, amit Tamás oda-vissza játszott le (memória spórolás ismét) a függőleges tengelyen. Most nem mesélem el, mi mindenen mentünk ehhez keresztül (még az Appaloosa engine-jében készített Amiga ball-omat is előszedtem megnézni) és a menet közbeni sebesség növekedés+amplitúdó-összefüggés vitáinkat sem bontanám ki (sok órát rágódtunk ezen).

Az árnyékolt labda

A kódolás és tesztelés közben közben felmerült, hogy egyszerűsítünk C64-es kinézetű labdákra, de a magam részéről ragaszkodtam az Amigásokhoz (nagyon). Ezeket eleve nem két, hanem négy színnel rajzoltam meg, hogy a labdák alja “árnyékolva” legyen (ilyen szempontból még szebbek is vagyunk, mint az Amigás labdák). A labdák eredeti négy színe menet közben módosítva van, nyolc féle “pepita” kombinációban rajzolódnak ki.

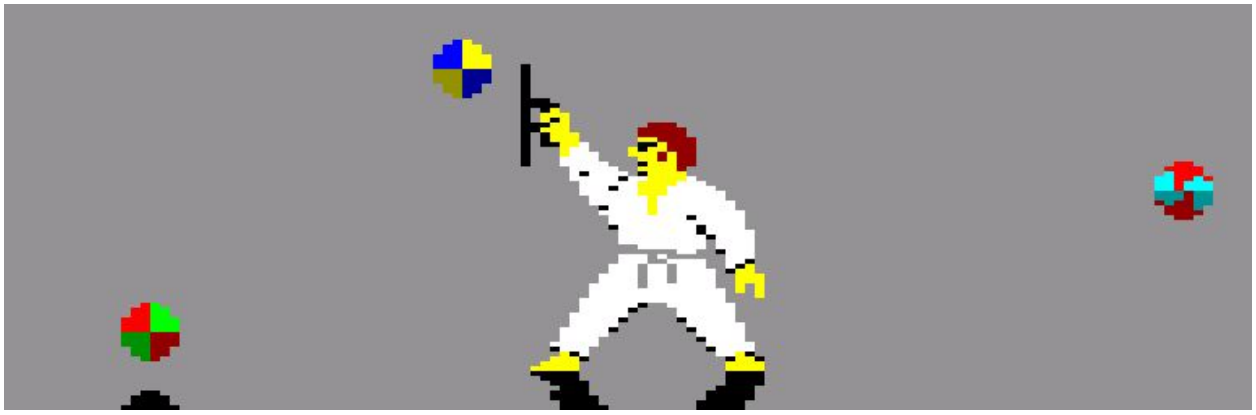
Természetesen nem maradhatott ki az árnyékuk a képernyő alján (amihez egy “vágással” is kellett kiegészíteni a sprite kirakó rutint), ami fontos látványelem, emellett a visszapattanás pontját is segíti megállapítani a játékosnak.

Mikor a hangok is elkészültek, rájöttünk, hogy még egy “apróság” hiányzik a labdázás teljességéhez (amit gyorsan meg is csináltunk és alig pár perc alatt be is lőttünk): a pajzs és a labdák pattogásának folyamatosan emelkedő hangja.

A kivédendő labdák darabszámát 32-re állítottuk be (a C64-es verzióban 64db van), mert a folyamatos sebességnövekedés és a keskenyebb képernyő miatt nem lenne elég idő reagálni rájuk.

Mikor mindezzel megvoltunk, találtunk egy furcsa hibát: elméletileg minden harmadik pályán kellett volna hogy labdázzunk, ami első körben stimmelt is. Két pálya, egy labdázás. De innentől már minden küzdelem után jött egy labdás rész és ez ismétlődött, egyesével váltogatva egymást.

Látszólag egyszerű hiba volt, mégis hosszú órákba telt, mire ki tudtuk javítani.

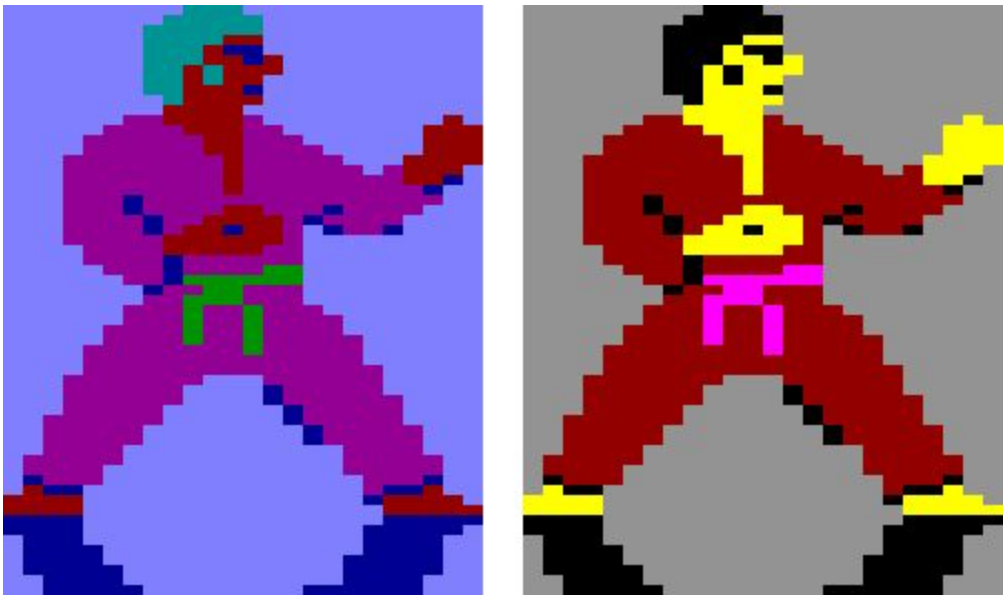


A labdák sebességét is jól be kellett állítani

Pontozás, övfokozatok színei

A memóriát teletöltő adatmennyiség és a gép sebessége miatt gyorsan eldőlt, hogy nem tesszük ki a technikákért kapott pontokat a játékosok fele fölé (ahogyan egyébként a többi gépen történik).

Ehelyett inkább megcsináltuk azt, hogy az övek színe a pontszámok függvényében változzon a karaktereken (tudomásunk szerint ilyen kizárólag Amigán van), ezért a ruhák egyik “színét” (amit a program a karatékák kirakása közben átszínez), az övre használtunk el.



A memóriában tárolt és a menet közben színezett karakter

Az övek színezésén túl, a jobb felső sarokban is mindig a legmagasabb övfokozat neve van, természetesen szintén az övnek megfelelő színnel kiírva.

Eredetileg volt még plusz két övfokozat (és szín), de végül maradtunk annyinál, amennyi a többi verzióban is van.

A pontozáson viszont már változtattunk, úgy, hogy kicsit magasabb ponthatárokat állítottunk be az övfokozatok eléréséhez. Az eredetiekhez képest talán egy kicsit arányosabban is osztottuk el őket.

Buborékok

A Mesterrel és a mondanivalóival pont annyit dolgoztunk (és szenvedtünk), amennyit előre is gondoltunk. Ehhez először is a Mestert a karatékáktól minél messzebbre kellett kitenni, emellett a feje fölött megjelenített buborék nem lóghatott ki a képernyőről (ami a c64-esnél jóval keskenyebb és emiatt zsúfoltabb is).

Tamás több algoritmust is írt, ami a mester pozícióját igyekezett kiszámítani. Ezek között volt egyszerűbb és bonyolultabb, végül a harmadik vagy negyedik vált be annyira, hogy száz mester megjelenésből kilencven biztosan helyesen van elrendezve. Ez egyébként olyan pontos, hogy néha egyetlen pixelnyi(!) különbség dönti el, hová kerül a Mester a képen.

A szövegek mondatai indexelt szavakból (félmondatokból) tevődnek össze, ezeket táblázatokban definiáltuk. Fontos volt, hogy a mondatokban bizonyos szavak lecserélhetőek legyenek (pl. *red/white/blue* wins/out), a cserék pedig természetesen eseményvezérelten történnek meg.

Nem csak a játék, hanem a demózás alatt is jelennek meg szövegek (többféle, mint a játék alatt), ezeket egy-egy küzdelem, high scores tábla után léptetjük körbe-körbe (sokat szórakoztam azzal, hogy minél szebben tördeljem ezeket a sorokat).

A játékos észre sem veszi, de *a buborékok mérete dinamikusan változik*. Figyelembe véve a sorok hosszát, Tamás kóddal rajzolja ki őket a Mester feje fölé, kiegészítve a Mester felé mutató nyilacskával). Erre a méretre igazított buborékra jönnek a tényleges szövegek.

Demó alatti szövegbuborék

Ponttáblázat

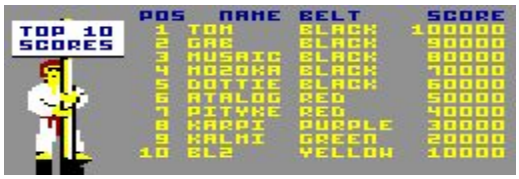
(Tamás)

Mivel a játék kifejezetten pontra megy (az övfokozatokat is eszerint osztja a program), a ponttáblázatot mindenképp meg akartunk csinálni. Több verzió elvetése után a legegyszerűbb nyert (a “kevesebb: több” elve szerint) és úgy láttuk, hogy az eredeti harminc soros listához képest elég lesz nekünk a tízes is. Négy oszlop: helyezés, név, öv, pontszám. Egyszerű, mint a faék, nem igaz? Mi is azt hittük...

Először a lista megjelenítése készült el (majdnem), de már a legelső tesztnél valami egészen furcsa dolog történt: a “helyezés” oszlop egytől-tízig való számai közé bekeveredtek más számok (még nulla is), igencsak bizarr hatást keltve. Értetlenül bámultunk ki a fejünkből (hiszen egyszerűbb kiírónk nem is lehetett volna), hogy kerülhetnek oda *random* számok?

Kiderült, hogy a megszakítás a “bűnös”, ezért a táblázat kirakása elé betettünk egy “halt” utasítást, hogy megvárjuk a következő megszakítást. Az “összefirkálások” ugyan ritkultak, de nem szűntek meg. A hosszú-hosszú hibakeresés közben észrevettük, hogy csak a számokat rontja el a táblázat kirakó. Na de mi az, ami *összefügg* a szám kirakással és a megszakítással? Hát az idő kijelző! A játék közben a program méri az kumite idejét: 30 másodpercről számol vissza.

Szép lassan összeállt a kép: amikor a számokat megjelenítendő karakterekké alakítjuk, akkor egy 6 byte-os memória bufferbe tesszük le sorba a karaktereket, majd amikor kész a szám átalakítása, akkor (mint egyszerű szöveget) kitesszük a képernyőre. Szépen el is kezdtük kirakni a táblázatot.. Aztán jött egy megszakítás ami kirakta az időt (is): ami ugye szintén számokból áll, szintén *ugyanabba* a memória bufferbe teszi le a karaktereket - ezzel szépen felül is írva a táblázatban épp kijelezni készülő pontszám karaktereit. A megoldás az lett, hogy végül két *különböző* memória buffert használtunk a számok karakterekké alakításához. Az egyik bufferbe csak a megszakítás dolgozott, a másikba pedig maga a főprogram. De mire az egész működési láncot és az abból következő hibát feltártuk (és megoldottuk), hosszú órák teltek el...



Csapatunk tagjai is szerepelnek

A név beírása már nem okozott problémát: a joy fel-le állítja az aktuális karaktert az ABC-ben előre-hátra haladva, jobbra a következő karakterre lép, balra töröl, tűzgombra pedig kész, ekkor a lista előtt álló nyertes karatéka meg is hajol.

Apropó, a *demózás közben* lista előtt álló, táblát tartó karatékáról még lesz szó!

Átszínezés

(Misi)

Már a tervezés korai szakaszában (tehát több, mint harminc évvel ezelőtt) úgy akartuk elkészíteni a programot, hogy a játékosok tetszőlegesen állíthassák be a karaktereik színeit (persze egy adott szín csak egy karatékán lehet), így nem csak fehér-piros-kék, hanem bármilyen (TVC palettájából származó) színű ruhában küzdhetnek. A színek léptetését három billentyű nyomogatásával lehet elérni, kizárva közben a “színazonosságot” is. Persze ezt sem ment simán, itt a billentyűkezeléssel gyűlt meg a bajunk: hol érzéketlen, hol meg túlérzékeny lett, többször újra lett írva, de végül sikerült elérni, amit akartunk.

Madarak

Tamás szívügye volt a madarak megléte és mozgása. Nem volt egyszerű összehozni, mert több dolognak is stimmelnie kellett. A magam részéről szerettem volna elkerülni a maszkolást (a kapu mögött takarni kell őket, ami gépidő), ezért olyan mozgáspályákat javasoltam, ami azt a helyet elkerüli. Aztán felmerült, hogy csak fekete madarak legyenek, így (mivel a kapu nagy része fekete) többfelé is tudnak mozogni anélkül, hogy maszkolni kelljen.

De aztán nem volt kerülőút: teljesen korrektre kell megírni, mert hol egy, hol két madárnak kell tudnia repülni, fehér és fekete színben is, lehetőleg random pontoktól-random pontig, így a takarásuk is elengedhetetlen. Tamásnak azt is meg kellett oldania, hogy pixelesen mozogjanak, ami újabb kódrészt igényelt, meg aztán volt pár bug, pl. néha a memóriába teleportáltak azok a madarak, stb. Összességében a madár-ügy ilyen módon belekerült jó két napba. Micsoda befektetés volt ez ez kétszer három pixel mozgatásába!

Demo

Teljesen nyilvánvaló, hogy a demózást nem lehetett kihagyni, ami Tamás előrelátó kódolásának hála nem volt olyan bonyolult, mint amilyennek látszik. A kék karatéka egyébként is a gép által van irányítva, ami voltaképp “joystick-szimulációval” lett megoldva, így a demó alatt a másik két karakter is ugyanezen a módon lett vezérelve (ugyanazt az intelligenciát használva a támadásokhoz).

Fejlesztés közben felmerült, hogy esetleg mindhárom karatékát ember irányítsa, de az nagyon belekavart volna a szabályrendszerbe, így ejtettük a megvalósítását.

Szerencsére, mert utólag visszatekintve nem fértünk volna be a memóriába.

Pause

Ez is olyan pont volt, amiben nem voltunk biztosak, hogy lesz. A memória végén jártunk (nagyon), a legkisebb lépést is meg kellett gondolnunk. Az gyorsan kikristályosodott, hogy a képernyő mérete és a TVC sebessége miatt három karatéka lesz öt helyett. Amíg Tamás írta a pause kódját, addig a C64-es verzió pause mozgási láncát néztem le és az ehhez illeszkedő animációink index számait kikeresve egyszerű szöveg táblázatba írtam őket. Ez került végül a kód-datába is (annyi előnyünk volt, hogy mindhárom karakter ugyanazt a mozgást végzi).

Apró részlet, de a “paused” feliratot is meg kellett csinálni, majd visszacserélni a “level”-re

Az eredmény olyannyira sikerült, hogy mikor az összehasonlító, promóciós videót készítettem, tökéletesen egymásra illeszkedett a mozgás (és a karakterek pozíciója) az eredeti, ZX-Spectrumos verzióval.

Idő kijelző

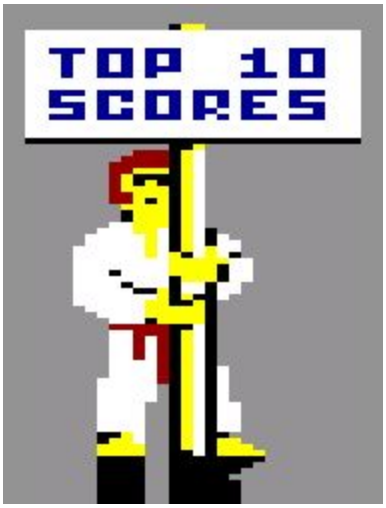
Az memória végének pár byte-ja az idő kijelző átszínezésére lett felhasználva. Az utolsó tíz másodperc kiírása pirossal történik (a villogtatásra már nem volt elég erőforrásunk, illetve akkor más, lényegesebb részek minőségét kellett volna csökkenteni, ami nem érte volna meg).

Kivételek

Bizonyos sprite-ok valós 16 színben lettek elkészítve és használva, ilyen volt például a Mester, akit TVC-n a (többi gépen való megjelenéshez képest) a lehető legszebbre akartuk csinálni. Természetesen nem hiányozhatott a lábfejének animációja sem (Tamás megtalálta a módját hogy beférjen a memóriába és maradhasson), a demózás alatt ezzel a topogással jelzi, hogy várja a küzdeni akaró játékosokat.

A C64 és TVC verzió

Van egy karakter, aki a küzdelemben semmilyen módon nem vesz részt, csak.. áll. Őt egyszerűen csak Mumusnak hívtam, mert a háttérkép után a legtöbb helyet foglalja a memóriában (leírni is szörnyű: tömörítve 420 byte!). Ő az, aki a highscores top ten tábláját tartja...



Mumus

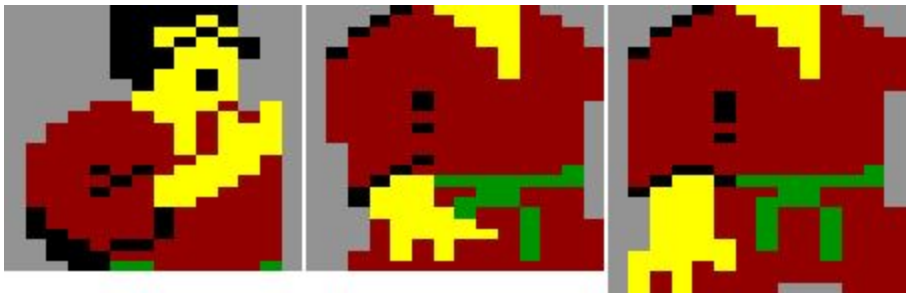
Optimalizálás

Ahhoz képest, hogy a legelején több háttérképet is terveztünk betenni a játékba, nagyon gyorsan a memória végére értünk. Kénytelenek voltunk elővenni az ecetes ollót és vágni ahol csak tudtunk, hogy beleférjünk a .cas formátum és a memória keretei közé.

Első körben elvetettünk bizonyos, még le sem kódolt részeket (az esetleg már bekerült grafikával együtt). Rögtön mellőztük például a “több háttér” ötletét, maradt az eredeti. Kivettük a nadrág lecsúszás geg-et is, mert a megjelenési gyakoriságához képest túl sok helyet foglalt volna. A pók, kukac és társai amúgy sem kerültek volna bele, mivel azok hi-res animációk, amelyek a 16 színű, dupla széles pixeles módban nem mutattak volna jól.

A kézfújás volt az első animáció, ahol elkezdtünk kivételeket tenni a sprite kirakásban. Sok memóriát pancsoltunk volna el, ha ezt az animot egész alakosan tároljuk le, ehelyett csak részleteiben írtuk felül a karaktert, kisebb méretű sprite-okkal (amihez ismét a sprite kirakó lett módosítva) és mindig csak akkora méretűvel, amekkora változás történt az előző fázishoz képest (ennél spórolósabbra nehéz lett volna csinálni).

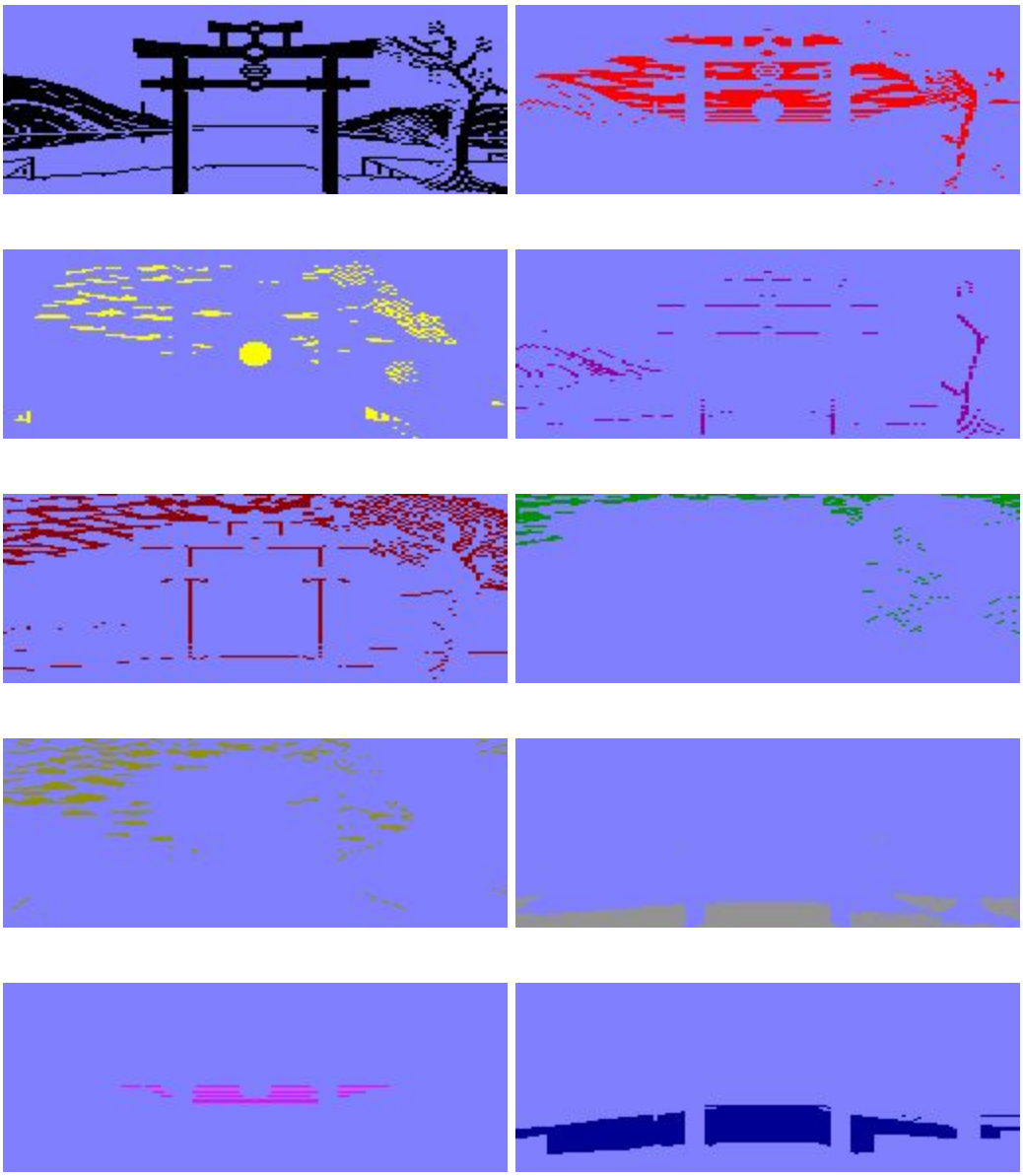
Mindezek mellé társult a simább lejátszás előnye is, mert csak akkor fújja a kezét a karakter, ha mindkét ellenfél fekszik a földön és a processzornak nem kell foglalkoznia sem az AI-val, sem az irányítással (egyik karakternél sem).



A kézfújás sprite részlet-fázisai

Háttér

Amire viszont nagyon ki kellett találni valamit, a háttérkép. Mivel ez foglalta a legnagyobb helyet a memóriában, kipróbáltunk valami merész, új dolgot: a színenkénti tömörítést. Az egész képet színeire bontottam és Tamás különálló képekként tömörítette le őket. Pár óra múlva beláttuk, hogy ennek hatékonysága elmaradt a byte tömörítéstől, körülbelül 300 byte-ot nyertünk vele összesen, aminél azért többre volt szükség. Csak utólag jöttünk rá, hogy voltaképp egy IFF-ILBM formátumhoz hasonlót akartunk csinálni, TVC-re!



Színre bontott rétegek (részlet)

A vége persze az lett, amit a legkevésbé akartam: alaposan átrajzolni a képet (hogy a byte tömörítésnek minél inkább tetsszen). Kezdtem a felhőkkel, “byte határ” felbontásra tettem át őket (ez játék alatt annyira nem zavaró, de tömörítve sokat számít), majd következett a fa lombjának teljes újrarajzolása, a part és hegyek vonalának finomítása, a kapu oszlopainak levágása. Ezeket Tamás inkább programból húzta meg.

A egész kép úgy lett összeállítva, hogy az ismétlődések kevésbé legyenek láthatók és minél jobban kihasználjuk a kompresszor lehetőségeit. Kibontva mindkét kép 6144 byte, de tömörítve sokat nyertünk a második verzión az elsőhöz képest: (4283 vs 3506 byte).

Az oszlop rajzoló kóddal és a hozzá szükséges táblázattal (ez együtt 129 byte) összesen a 648 byte volt a teljes nyereség a két tömörített verzió között!

Optimalizálatlan háttér (4283 byte)

Optimalizált háttér (3506 byte)

További csökkentést értünk el azzal, hogy bizonyos animációs fázisokat töröltünk és hasonlókkal helyettesítettük őket. Így a kétfelé rúgásból, ugró rúgásból, hasba rúgásból ki tudtunk venni egy-egy fázist. Gyakorlatlan szem észre sem veszi. Aztán egyszer csak oda jutottunk, hogy már nem volt mit kivenni, *sehonnan*.

Legalábbis azt hittük.

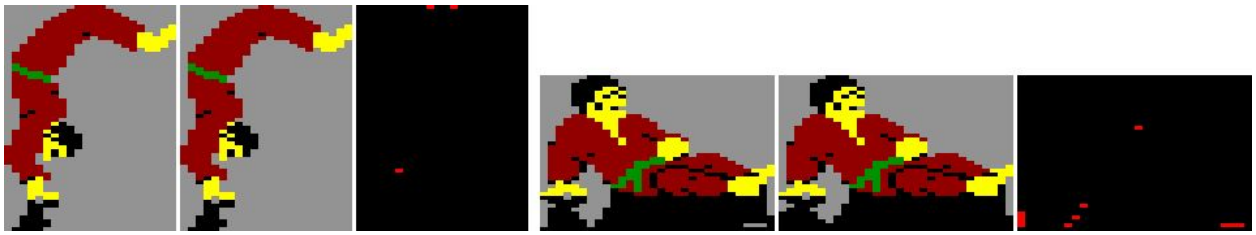
Aztán egy napon… Kéne még 20-30 byte! - szólt Tamás. Hát persze hogy kéne, de *honnan*? Ami memóriában volt, mindenre szükség volt, már nem volt mit *kitörölni*. Márpedig elő kell teremteni azt a pár byte-ot, ha törik, ha szakad. És mivel kódból nem lehetett törölni semmit, nekem kell tenni valamit, kontent oldalról.

Ezért bizonyos animációs fázisokat elkezdtem a kompresszorhoz igazítani.

Hogy hogyan? Ahogy a háttérképet is: innen elvettem egy pixelt, amoda pedig betettem. A nadrág szárába plusz egy pixel, a karjából le egy pixel. Alig néhány sprite-hoz nyúltam hozzá, máris 26 byte-nál tartottunk, így meg lett Tamásnak a kellő hely.

Számítva rá, hogy további byte-mentésre szükség lesz még, egyelőre nem használtam ki az ebből fakadó összes lehetőséget. Az ember legyen óvatos, gondolkodjon előre…

Nem kellett sokáig várni. Hamarosan ismét szükség volt “néhány byte-ra” (talán a highscores táblázat következett), de már tudtam hova nyúlni, volt egy kis mozgásterünk (hogy pontosan mekkora, azt persze nem tudtuk). Ezután már nem vártam, az összes fázist átalakítottam a fenti módon, első körben 80, összesen pedig közel 200(!) byte-ot felszabadítva.



256/253 és 293/283 byte (a különbséget piros pixelek mutatják)

Mondanom sem kell, hogy minél több ilyen pixel-variációt próbáltam ki egy-egy képen, annál inkább kellett emlékezni azokra, amelyekkel addig a legtöbbet tudtam nyerni (és hogy ne csináljak olyat, amit már egyszer próbáltam).

Közjáték

Az utóbbi években többször is nyüstöltem Tamást, ássa elő a régi kazettáit, mentsük meg ami menthető és ki tudja, talán elfeledett kincsekre is lelhetünk közben… Aztán végre hozott nekem egy csomó anyagot (közben öcsém megajándékozott egy működőképes rádiómagnóval) és bedigitalizáltuk az összes megtalált kazettát. Többször is.

A kazettákon voltak HomeLab és TVC programok, és ezekhez mindenféle egyéb fájlok (például screen mentések, adatok, stb). Meglett többek között Tamásék egyik (még BASIC) játéka, a “Katakombák Mélyén”. Örömömben nekiálltam egy kicsit pimpelgetni (címképernyő, pályaszínesítések, HUD, stb), aztán Tamás is belenyúlt itt-ott és az egésznek a vége az lett, hogy versenyen kívül (és nagyon alaposan kitesztelve) még a nyáron ki is adtuk.

Pihenésképpen ezt is kiadtuk (versenyen kívül)

Aranyhíd

A háttérkép aranyhídját nyilvánvalóan nem hagyhattuk ki (az egyik fő, állandóan szem előtt lévő látványelem). Tamás felvetette hogy egyben lévő animációként játsszuk le, de számomra egyértelmű volt, hogy ez teljesen esélytelen (annyi fázis kéne hozzá, hogy ha semmi más nem lenne a memóriában csak ez, akkor sem férne be).

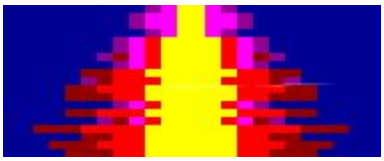
Ehelyett egy algoritmust javasoltam, amit aztán gyorsan meg is írt (őszintén szólva majdnem gyorsabban, mint amennyi idő alatt el tudtam neki magyarázni, mit is akarok). Aztán persze kellett még finomhangolni, de az már apróság.

Az aranyíd alap mintája

Az algoritmus egyszerű, látványos eredményű és igen takarékos. A “híd” képe egyben van a memóriában, de hogy a háttér adott sorára a híd *melyik* sorát másoljuk be, azt már egy táblázatból vesszük ki.

Az aranyhíd vízen lévő helyének minden sorára meg volt adva, hogy a híd képének melyik sorától melyik soráig vegyen mintát (ciklusonként haladva a táblázaton). A kirajzolás eredménye azért *tűnik* random hullámzásnak, mert a híd képéből való, soronkénti mintavétel *darabszámban eltérő* (hol három, hol négy, hol öt) és a program ezeket ping-pong módon olvasva teszi ki a táblázat rá vonatkozó sorából.

Emiatt alakul ki a rendezetlennek tűnő látvány (ami ennek ellenére mindig ugyanazt a lejátszási mintát követi).



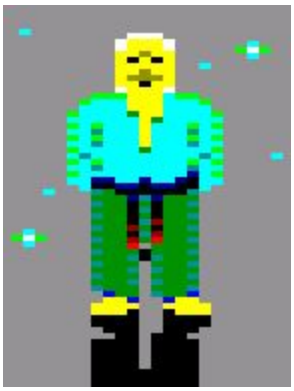
Az algoritmussal kevert kép

Végül Tamás megfelezte az aranyhíd minta-képet és inkább kódból tükrözte meg a képernyőre kirakáskor. Ezzel is spóroltunk pár byte-ot.

Particle

Két hely is van a játékban, ahová particle készült: az egyik a víz felszíne, a másik az éppen megjelenő Mester körüli terület. Ráadásul minden particle sprite-ja egyben animáció is.

A Mester körüli csillogás három, oda-vissza lejátszott fázisból áll (szintén indítási eltolással).

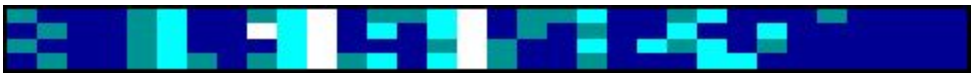


Mester és particle

A víz csillogása egy táblázatban meghatározott területen belül, random helyre kerül kirakásra, ugyanúgy, ahogy a Mester körül is.

A víz esetében először minden animáció azonos fázisban volt indítva, ami előre várható módon nem volt túl szép (akkor sem, ha az animációk különböztek egymástól), ezért indítási offset értéket is kellett, hogy tudjon a rutin.

A fázisok grafikailag maszkolatlanok (nem lyukasak) és az összes szín használható bennük. A csillogás grafikai tömbjében négy darab, nyolc fázisból álló animáció található, amelyek mindkét irányban lejátszhatók.



A csillogási fázisok négy elnyújtott pixelnyi szélesek.

Zene

Kezdettől fogva kérdéses volt, lesz-e a játéknak zenéje. Felmerült, hogy csak prütty-prütty effektek lesznek és kész, hiszen a gépnek *egyetlen* hangcsatornája van, amin csak a hangmagasság és hangerő állítható - semmi több.

Aztán ott volt a szükséges memória kérdése… ennyi grafika és kód mellé hová férhetne be a song data (nem beszélve magáról a lejátszóról)? Na és melyik zenét írjuk meg: az Amigás, a C64-es, vagy valami saját verziót)? Milyen zeneszerkesztőt csináljunk ehhez (hiszen ki vágyik arra, hogy a forrásban, sima dw/db datában adja meg a hangokat)? Az ilyen kényelmetlen kérdések miatt a zene a legvégére maradt, “ki tudja lesz-e belőle valami, ezer fontosabb dolog van előtte amúgy is” - felkiáltással.

Így mentünk bele a novemberbe. Hat hét volt a határidő végéig.

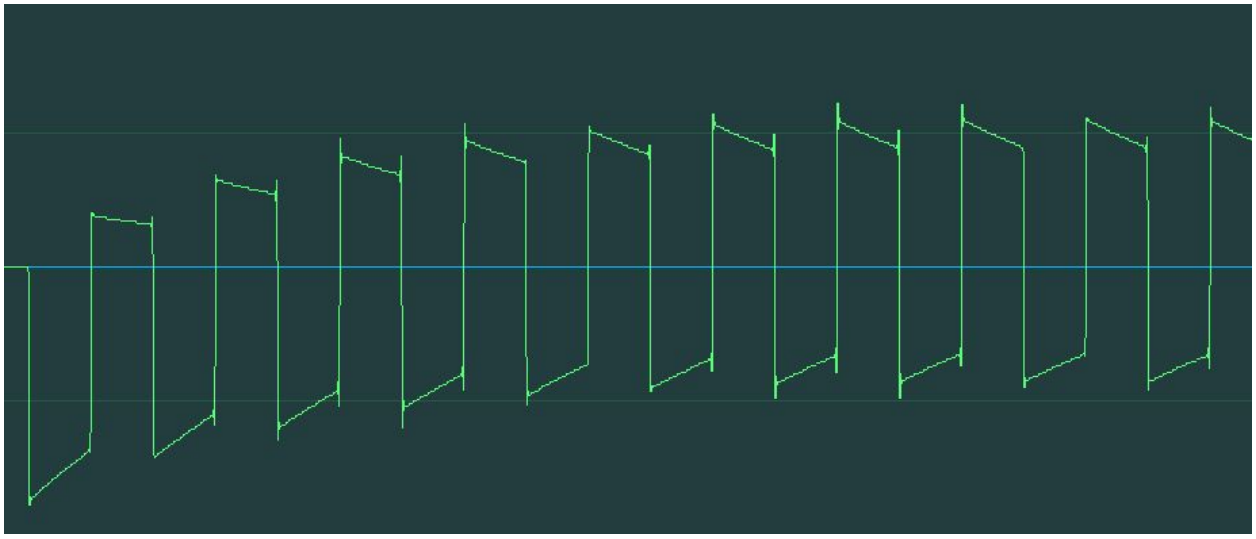
Ekkortájt kezdtünk kísérletezni egy egyszerű lejátszóval, ami (mivel még nem volt része a játéknak) nem megszakításból, hanem, saját főciklusban futott. Világos volt, hogy két (ál)csatornánál nem lehet több (basszus+szóló), így első körben a *mixelést* teszteltük. Milyen gyorsan váltogassuk a csatornák között? Milyen tempóban játsszuk le a zenét? Hogy fog az egész szólni? Lesz egyáltalán értelme az egésznek?

Megelőzendő a felesleges kódolást, elővettem a SID lejátszót és kiexportáltam az IK+ zenéjének csatornáit külön wav-okba. De úgy, hogy az adott csatorna 100% hangereje mellett szólt a másik kettő is, de csak 10-10% hangerővel.

Ezekből hallás útján beírtam (tesztnek) a finálé basszus+szólóját, úgy, hogy a patternek dupla sebességgel futottak, a leütések pedig felváltva szóltak a két csatornán (mintha már TVC-n hallgatnánk, “mixelve”). Hogy minél jobban hasonlítson a leendő hangzásra, generáltam négyszög wav-okat is, ezekkel szólaltattam meg a note-okat.

A felváltva szóló note-ok

Már az elejétől nem hagyott nyugodni, hogy az emulátorban kiadott hang valamiért másképp szól mint a fent említett, generált négyszög jel, ezért emulátorból is mentettem ki wav-okat. Kiderült, hogy ezek hullámformája csak *hasonlít* a négyszögre. Hát, ha így szól a TVC hardvere, akkor a teszt zene is úgy szóljon, ahogy a TVC, ezért ezt hangot használtam a továbbiakban.



Nem éppen sima négyszög jel :)

A minél valósághűbb szimuláció kedvéért kiexportáltam wav-ba a már TVC-ül hangzó részt és grabbeltem egy videót a futó játékunkból. Összerendereltem a kettőt és csak néztem, néztem… észbe kaptam és áttoltam Tamásnak is, várva a hatást... Úh, ez marhajó, megcsináljuk!

És hogy fog mindez beférni a gépbe? Kit érdekel? Megoldjuk és kész!

Zeneszerkesztő tekintetében (több évtizedes tapasztalatunk alapján - számunkra nyilvánvalóan) az Amigás és PC-s trackerek világából indultunk ki, annyi különbséggel, hogy a mi patternjeink nem egyben, hanem *sávonként* lettek tárolva - ebből következően töredékére lehetett csökkenteni a patternek számát és a zene data méretét.

További (nem éppen elhanyagolható) különbség, hogy a patternjeink hosszát tetszőlegesen lehetett beállítani, amivel a két sáv teljesen függetlenedni tudott egymástól, persze nagyon ügyelni kellett arra, hogy ne csússzon szét a szinkron közöttük sem zeneileg, sem technikailag.

De amíg Tamás írta a zeneszerkesztőt, nekem foglalkoznom kellett valami mással is.

Mégpedig azzal, hogy a Protrackerben kiadott hangmagasság függ a hangszer eredeti mintavételi frekvenciájától - tehát hiába írom én meg a zenét csilli-villire, ha az el lesz transzponálódva az eredetitől. Meg kellett tehát tudni a zene eredeti hangfekvését.

A SID2MIDI-vel a nevéből következően MIDI-be lehet exportálni a SID zenéket (meglepő, nemde?). Így már az első leütésből tudni fogom, mi a bázishang - gondoltam én. Az export után vettem észre, hogy hoppá, tud ez .txt-be is exportálni… kettő századmásodpercenként...(!) ami jó, mert látszanak a arpeggio-k is.. és nem jó, mert az ebből keletkező .txt 23.118 sor ritkítására nincs opció exportkor.

Szóltam Tamásnak, hogy a fő leütések közül ki kellene szedni nyolc-nyolc sort egy txt fájlból. Pár perc alatt írt egy ilyen rutint. Ráfuttatta a .txt-re, majd visszaküldte. Nyugtalanító eredmény született: alig pár sor után egy csomó note is eltűnt össze-vissza. Jó, akkor hét sort vegyünk ki a leütések közül. Az sem lett jó. Legyen hat… elszórakoztunk ezzel egy darabig, akkor esett le a tantusz: Rob Hubbard (egyébként fantasztikus) SID lejátszója *pontatlan!*

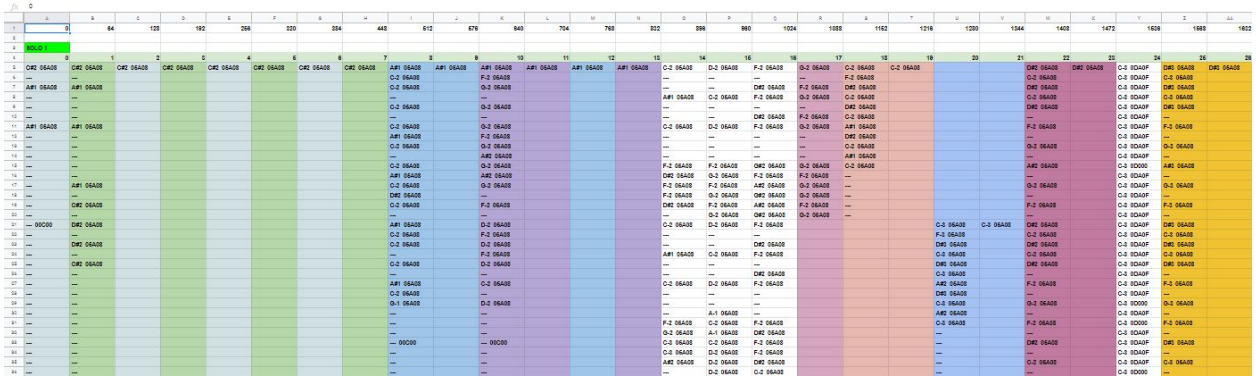
Már az eredeti vason is hallható, hogy néha elcsúsznak a hangok időben egymáshoz képest (ez például a Sanxion boot zenéjében “kiválóan” hallatszik) ebből következően az exportált note-ok hangközei szintén “pontatlanok”. Így a 2.495.545 byte-os .txt teljes hosszából kénytelen voltam magam kiszedni a felesleget, manuálisan. Erre a Notepad++ makróját használtam, amivel egy mozdulattal ki tudtam törölni hét sort, és ha volt még egy sornyi csúszás, akkor azt kézzel vettem ki. Ezután átléptem a következő (valós) leütést, megint futtattam a makrót... és így tovább, minden egyes valós note után, mind a 23.118 sort átvizsgálva. Olyasmi érzés volt ez, mint mikor a Jurassic Park könyvben elkezdik végignézni a park CRAY-re írt programjának forráskódját, hogy mit kavart el Nedry benne.

Így megkaptuk a valós, nettó note-okat, ami alapján (még mindig Protrackerben) megírtam a zenét teljes, 60 patternes hosszában (nyilvánvalóan nem használhattam ki a patternezés fő lényegét, az ismétlődést). Az első csatornára a szóló, a harmadikra a basszus, a másodikra az éppen valamelyik szabad “csatornára” (majdan) beférő köztes hangok, a negyedikre egy egyszerű dob szekvencia került (csak úgy).

Közben valamennyire elkészült a DevTool zeneszerkesztője is (persze nyilván fapados módra, házi használatra).

De még mielőtt elkezdődhetett volna a tényleges, TVC-re készülő patternek beírása a saját szerkesztőnkbe, hátra volt még egy előkészítő munkafázis: Kiexportáltam a Protracker song datáját is .txt-be, amit aztán sávonként és patternenként bemásoltam egy google sheet táblázatba úgy, hogy (megfelelően megszínezve őket) már ott lehessen látni a zenei ismétlődéseket. Így kaptunk egy 60 oszlopos, 3 zenei sávos, 204 soros táblázatot, beleírva a DevTool-os note pozíciók számával (3.744), a tényleges note-okkal egyetemben (összesen 12.240 cellában).

Talán túldolgozásnak tűnhet mindez, de itt nem lehetett viccelni, a csaknem nyolc perces zenének minden egyes hangját kézben kellett tartani.



Részlet a pozíció-pattern-note-táblázatból

Csak *mindezek után* következett a TVC-n megszólaló zene hangjainak beírása A DevTool zene szerkesztőjébe.

A legfontosabb az volt, hogy a zene teljes hossza minél előbb legyen meg. Ezért készült egy 64 és egy 32 soros “üres” pattern is (note-ok nélkül), ezeket használtam a szinkronizálásra és placeholderként.. Először a basszus sávot írtam meg, mondhatni alfa állapotra (folyamatosan hallgattam össze a C64-es wav exporttal). Utána következett a szóló sáv. Ennek írása közben ügyelni kellett arra is, hogy a patternek hossza és a bennük lévő leütések száma minél optimálisabb legyen, adattárolás szempontjából. Tehát *menet közben* kellett a TVC-s patternek számát megállapítani és a pozíció szerkesztőbe is beleírni.

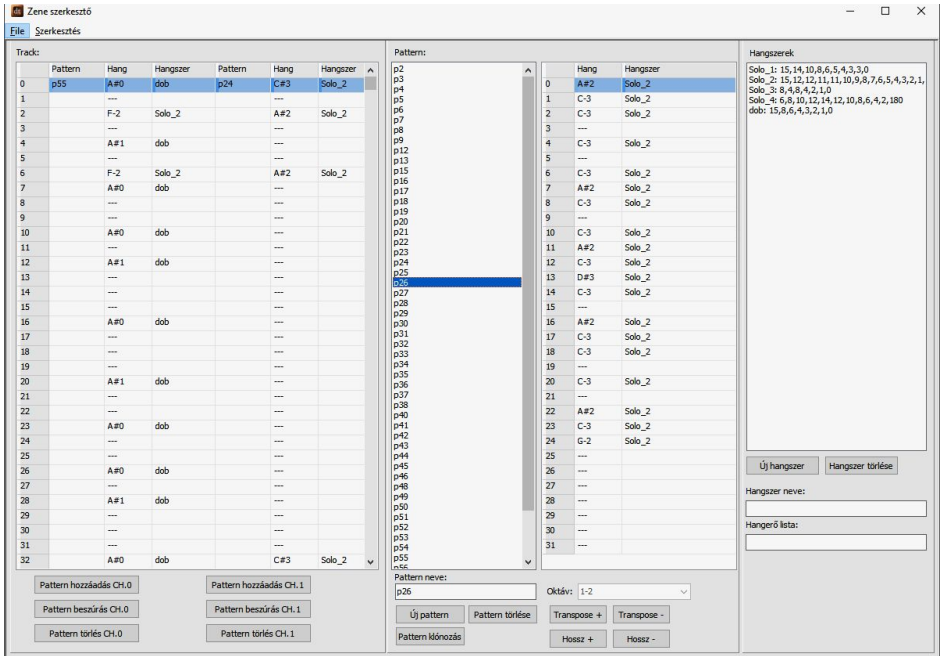
A patternek egyébként szintén (mondhatni) “tömörítve” vannak, például a leütések közötti szünetek úgy vannak tárolva hogy “szünet, x darab”.

Minden kisebb-nagyobb zenei szakaszt össze kellett hasonlítani az eredetivel, hogy a patternek egymás után stimmeljenek és az elütéseket is észrevegyem. Ez persze nem úgy ment, ahogy általában gondoljuk, hogy “play és meghallgatom”, á, dehogy! Minden egyes meghallgatáshoz ki kellett menteni a trackerünk songját, utána exportálni a clipboardra, ezt bemásolni a Z80 forráskód megfelelő helyére, lefordítani futtatható .cas-ra, betölteni az emulátorba és visszahallgatni (a fapad varázsa ugye).

Nem tudom megmondani, hány százszor csináltam meg ezeket a lépéseket, csak azt, hogy a végére teljesen kikészültem a 12Hz-es kemény arpeggio-csiripolástól, a végén minden hang kalapácsként ütötte a halló idegrendszeremet, tényleges fájdalmat okozva. De hát a tökéletességnek ára van…

Közben a lejátszó kapott egy új feature-t is: képes lett a hangerőt is állítani, így már lehetett ADSR-hez közelítő “hangszereket” összerakni (persze kézzel beírva az értékeket), amiket a hangerő-sor végétől visszafelé tetszőleges szakaszon loopolni is lehetett(!)

Nosza, lehetett az összes note-hoz (egyesével) hangszert is beállítani. Aztán rendbe tenni az oktávokat. Utána kitölteni a rendelkezésre álló helyeket töltelékhangokkal. Ezután optimalizálni úgy, hogy mégiscsak vegyünk ki néhány hangot, hogy jobb legyen a pattern méret-teljesítmény viszonya (minden exportáláskor kiíródott a zene hossza, ez alapján folyamatosan próbáltam csökkenteni a méretet). Végre elértük (és még kisebbre is vettük) azt a zene hosszt, mint ahová be akartuk cipőkanalazni: a tömörített háttérkép helyére.



A kezdő taktusok

A következő nagy lépés az volt, mikor a lejátszó bekerült a játék alá és megszakításból kezdett működni. Kicsit tartottunk attól, mennyi időt fog megenni a processzortól, de nem tapasztaltunk érezhető sebesség csökkenést a játékmenetben. Így következhettek a...

Hangeffektek

Az effektek szintén patternek, csak rövidebbek. Olyan hangokat próbáltam csinálni, amelyek önmagukban is több csatornásnak hatnak, mély és magas hangok vegyesen, ütések, elesések, stb., a labdák például kifejezetten gumilabdás hangzást kaptak.

Az egész lejátszót úgy írta meg Tamás, hogy mikor egy effektet ki kell adni, a zene szóló sávján játssza le őket, majd folytatja a zenét tovább, mindemellett bármikor kikapcsolható akár a zene, akár az effektek (amely növeli a játék komfort érzetét).

Hang és zene ügyben az első kapavágást 2019.11.05-én tettük meg és 19-én zártuk le.

Zene utántöltés

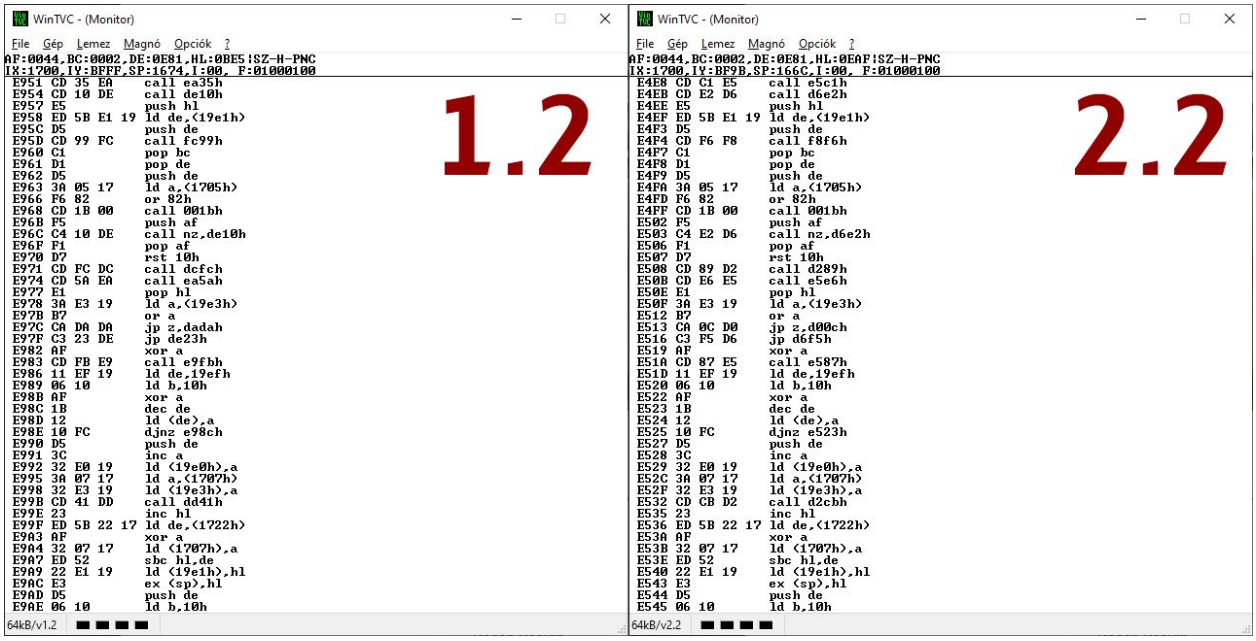
Tamásnak a következő tennivalója az volt, hogy a zenét és hangeffekeket kitegye egy külön .cas-ba, amit a főprogram tölt be, közvetlenül azután, hogy a háttérképet kirakta a képernyőre. És oda töltse be, ahol ez a kép kompresszorozva tárolódott. Ezt meg is csinálta, remekül működött a töltés, így minden adat, a zene is a helyére került és futott minden szépen.

Legalábbis neki, floppy emulációban, .dsk-ról.

A kazettás dolgokkal én foglalkoztam. A főprogramból és a zene .cas-ból generált wav-okat egymás után másolva és betöltve nekem is ment minden, egészen addig, amíg ki nem próbáltam 2.2-es ROM-mal. Mivel ott 42.155 byte-ot ír a BASIC-re, ki kellett próbálni, ott is működik-e minden, ahogy kell.

Egy darabig ment is, épp csak a zenetöltő nem működött.. De miért is? Mert a kazettáról töltéshez a 2.2-es ROM-ban *máshol* kellett a töltőrutint meghívni (egyből beugrott egy régi emlék, hogy ebbe már a 80-as években is beleütköztünk).

Mivel Tamásnak egyéb elfoglaltsága akadt, az (esélytelenek nyugalmával) magam kezdtem megkeresni a töltési hívást a másik ROM-ban. Olyan *nagy* különbség csak nincs a két ROM között… Pár perc alatt meg is találtam, így Tamás másnap olyan univerzálisra lett írta a töltőt, hogy bármilyen TVC ROM-mal működjön, kazettán is.



A két ROM cím közötti eltérés a betöltés meghívásához

Címképernyő

És most következhetett valami egészen más (egyik vesszőparipám): a title képernyős loader. Ilyen méretű és kaliberű játéknál semmiképp sem akartam hogy kimaradjon (az eredetileg Jon Egg által készített képet már hónapokkal előbb átalakítottam TVC-re). Nyilvánvaló igény volt, hogy semmilyen töltési üzenet (searching, reading, stb.) ne látsszon rajta és hogy *ugyanaz* a loader .cas képes legyen floppyról és kazettáról is betölteni magát a játékot.

Tamás összedobta a képkirakót a loaderrel és áttolta nekem, hogy szórakozzak vele én, ha már annyira akarom… Így komoly programozói teljesítményt nyújtva (beleírtam kettő sort az assembly forrásba, hihi) kiegészítettem azzal, hogy letiltottam minden BASIC visszajelzést (hogy ne legyenek töltési üzenetek és a teljes képernyőt ki tudjuk használni).

Remekül működött is minden. Lemezről...

Mert kazettán valami borzalmas pokoljárás kezdődött. A képet kirakta ugyan, épp csak semmiképp nem akarta betölteni a főprogramot. Súlyos, órákon át tartó wtf-ezések és file név próbálkozások után rájöttünk, hogy nagyon *nem mindegy*, hogy kis-, vagy nagybetűkkel van a wav fejlécében a cas neve (ez persze csak akkor derült ki, mikor kínomban visszakapcsoltam az üzeneteket és a “found:” szöveget írta ki a “reading:” helyett). Közben folyamatosan csörgött a telefonunk, utóbb kiderült, a torkos borz hívott minket, hogy adjuk már át neki ennek a szívásnak a receptjét…

Lezárás

Tulajdonképpen végeztünk, már csak apróságok voltak hátra. Például beilleszteni a készítők nevét is a játékba...

A string-szeletekben tárolt szövegek közé először a civil neveink voltak beírva, de Tamás azt mondta, miért ne használjuk a fun nevünket, a Doberdo Brothers-t? Ez nagyon helyénvaló ötletnek bizonyult és az sem volt mindegy, hogy mindez kevesebb karaktert jelentett, mint a kettőnk nevei összesen. Aztán be kellett még írni a music/sound on/off és a ruhák színeinek állítását vezérlő gombokat is a Mester buborékjába, mikor demózik a játék.

A szövegek bekerültek a forrásba, csak volt egy kis probléma. Csak egy egészen “apró”. A lefordított cas hossza 42.256 byte lett. *Eggyel* több, mint a memóriába tölthető adatok hossza… Ha írunk egy programot, ami hosszabb mint memória hossza, a program értéke egyenlő a nullával, hiszen ez *bármilyen* hibát okozhat.

Próbálkoztam mindenféle variációval, de a .cas makacsul tartotta ezt a hosszt (vagy épp egy byte-tal még hosszabb lett). Mi a megoldás? Mert valaminek lennie kell…

Észrevettem, hogy a szövegtáblánkban az 1 és a 2 stringek így vannak definiálva:”1 “, “2 “. Vagyis mindkettő után van egy-egy space, hogy ne folyjanak össze az utánuk kiírt szavakkal. Meg aztán a ruha színváltást kiíró mondat egyik stringje is így nézett ki: “4 5 6 ”. Ezek a szóközök összesen három byte-ot jelentettek. Viszont mind a “2 ”, mint a “4 5 6 ” string kiírása után *mindig* a “keys” szó következett. Ebből jött az ötlet: a 2 és a 6 utáni szóközt kivettem, és inkább a “keys” elé tettem szóközt. Így nyertünk egy byte-ot!

Ezzel lett a cas végleges hossza a maximális 42.255 byte. A program értéke “nulláról-egyre” egyetlen space-n múlt!

Legalábbis addig, amíg ki nem vettük a többek által (jogosan) kifogásolt stop gombot, így végre nem lehetett véletlenül rányomva kilépni a játékból (ezzel is nyertünk pár byte-ot, így lett a fő .cas hossza 42.241 byte).

DevTool

(Tamás)

A fejlesztés kezdetén lepróbáltam a TVCStudio-t. Akkor még nem tudta a Z80 nem publikált utasításait, különös tekintettel az index regiszterek alsó és felső regisztereinek a használatára (ixh,ixl,iyh,iyl), holott tapasztalatból tudtam, hogy ezekre nagyon nagy szükségem lesz. Eszembe jutott, hogy körülbelül tíz éve írtam már egy Z80 fordítót, ami már akkor is tudta ezeket (viszont nem volt tesztelve és igazából soha nem használtam fel).

Adta magát a lehetőség, hogy akkor ezt tegyük bele a DevTool-ba (ami a kezdetektől úgy készült, hogy minél inkább kompatibilis legyen a TVCStudio-val), bár később még módosítani kellett pár helyen, mert itt-ott akadtak fordítási hibái, illetve a játék megkövetelte, hogy az index regiszterekkel címezve struktúrákat is kezeljünk (erre találták ki ezt a címzési módot).

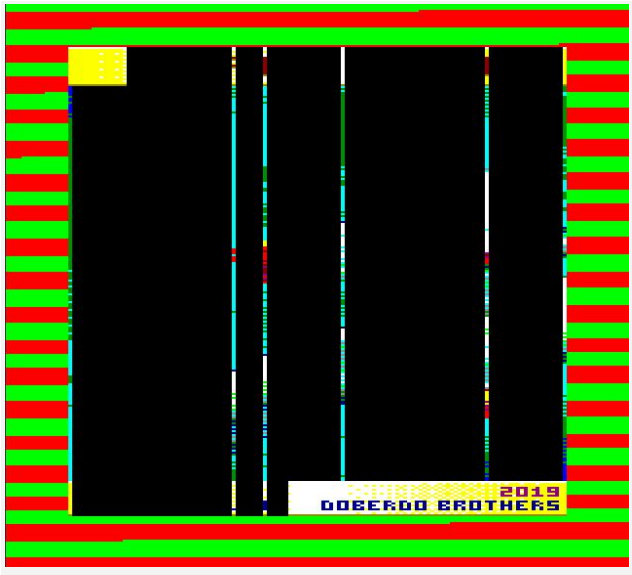
Ehhez tudnom kellett a struktúra minden változójának offset címét. Ezt eleinte “equ” val adtam meg, de kényelmetlen volt, ezért bevezettem az “enum” ill. a “struct” direktívát. Később ennek hatalmas előnyét láttuk.

Történt, hogy egyszer csak - ki tudja hanyadjára - elfogyott a memória. Mivel a legfelső (harmadik) lapot is RAM-ként használtuk, adta magát az ötlet: a program változóit tegyük fel a legfelső lap elejére, ahol van szabad hely. A “struct” direktívákkal felépített struktúra láncolatban elég volt a kezdő címet módosítani és máris felkerült az egész változó tömb a legfelső lapra. Több, mint 700(!) byte-ot nyertünk ezzel a .cas memória területen.

Promóció

(Misi)

A projekt vége felé elkezdtük a reklámozást, figyelemfelkeltést. Óvatos voltam, nem akartam, hogy túl hamar kiderüljön mit is csinálunk (bombaként akartuk robbantani), ezért formabontó módon tettem közzé részleteket. Például a címképből, a játék hátteréből csak néhány csíkot mutattam meg és olyan dolgokat, amiből nem derülhetett ki, miről van szó. Például a dokumentációból, szövegbuborékból, pontlistából is kerültek fel részletek.



Címképernyő részlet :)

Elsöprő sikerre persze nem számítottunk, de elértük hogy fejekben ott voltunk. Bár az igazsághoz az is hozzátartozik, hogy a számításunkba így is hiba csúszott: mikor a játék háttérképének egyetlen pixeloszlopát széthúztam teljes szélességűre, hát Buza Laci nem kitalálta? :)

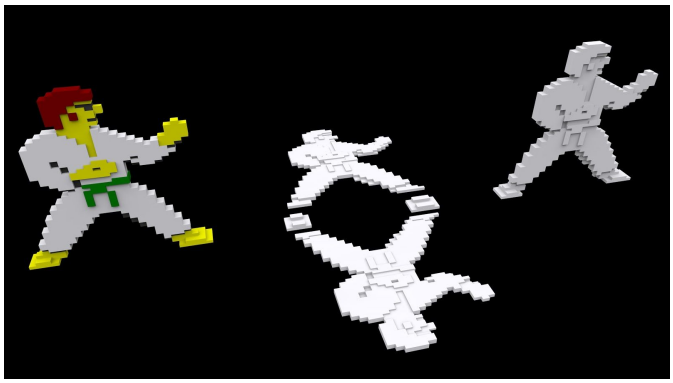
Ki is feküdtem rögtön :)

Kiadás

A fentiek után a kiadást sem akartuk elaprózni. Sőt, ebből kifejezetten stratégiai előnyt akartunk kovácsolni. Ne feledjük el, hogy versenyeztünk (kőkeményen) és ez olyan pontja a játéknak, ahol szintén próbáltunk többet nyújtani a többieknél (akik talán nem is gondoltak erre a lehetőségre). Ezért olyan komplettre akartuk csinálni, amilyenre csak lehet, minél több “szolgáltatást” nyújtva a “vevőknek”. Amíg Tamás a kód egyéb részeivel foglalkozott, a “kiadást” magamra vállaltam.

A nyomtatható borító is tesztelve lett(!)

Ennek természetesen tartalmaznia kellett a .dsk és .wav fájlokat (utóbbi kazettára vehető (vagy magnó emulátorba, telefonra tölthető), ahonnan vasba betölthető). Emellé kapott egy .pdf fájlt, teljes, minden részletre kiterjedő “gyári” leírással. Készült hozzá színes, méretarányos (külső-belső) kazettaborító is, hátha valakinek kedve lesz (akár fotópapírra) kinyomtatni. Bónuszként pedig készült .stl 3D nyomtatható modell is (pici talpakkal). Ennél a modellnél egyszerre sík és 3D is a karakter, anyagától függően kézzel ki is festhető.

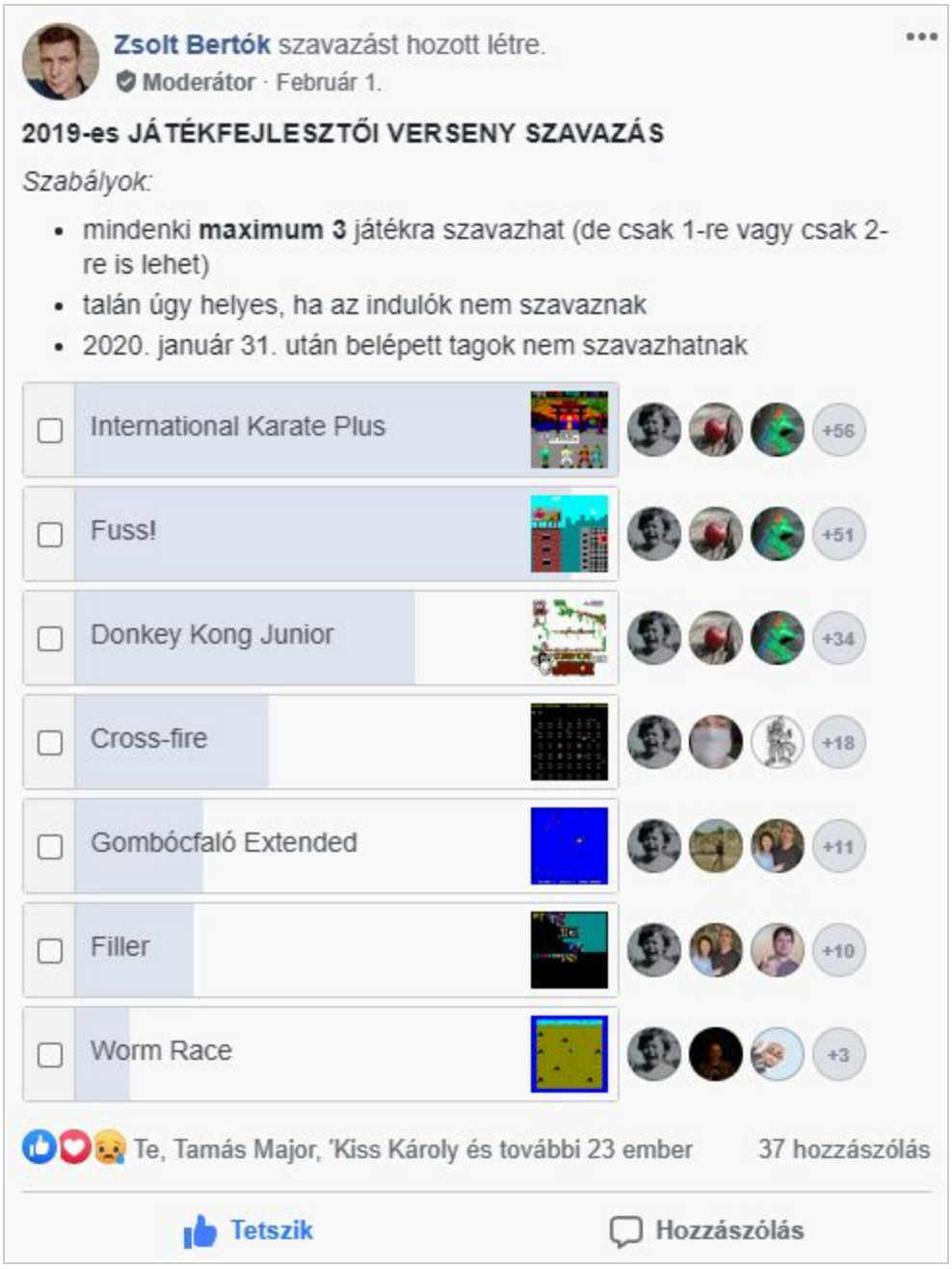


Nyomtatható 3d karakterek

Ahogy a gép memóriáját dugig töltöttük intelligens információval, ugyanezt műveltük a .dsk-val is: felkerült rá a .pdf dokumentáció is, így ha TVC-n elolvasni talán nem is, de fizikai lemezen (is) terjeszteni lehet vele a kézikönyvet.

Első helyezés

A facebookos TVC csoportban történt szavazásban elsők lettünk. Kemény munka, sok lemondás, feszültség, stressz, bizakodás, öröm. Hogy mindez megérte-e? Szerintünk igen.



A szavazás végeredménye

Olyan dolgot tettünk le az asztalra, ami világviszonylatban is komoly teljesítmény. A gépet a végletekig kihasználtuk, minden erőforrásával éltünk. Mi - a Doberdo Brothers - úgy véljük ilyen hozzáállással érdemes bármit csinálni, amit szeretünk.

Utóélet

Nagy örömünkre szolgált látni, hogy *egy teljes egészében emulátoron fejlesztett játék igazi hardveren is tökéletesen fut*. Köszönetünket fejezzük ki azon fejlesztőknek, akik ezeken dolgoztak (akár éveken át), szívüket-lelküket beletéve, hogy minél inkább lemodellezzék az eredeti gépet. Ez a tökéletességre törekvő, hosszútávú gondolkodás az, ami sikerünket is lehetővé tette, meg tudtuk valósítani gyerekkori álmunkat, emellett (mellékesen) versenyt is nyertünk vele.

Az egyik legérdekesebb élményünk az volt, mikor egy görög srác gépén láttuk futni a játékunkat (amit természetesen görögül kommentált és ahogy kiejtette a nevünket, már önmagában show-elemnek éreztük).

Videó, egy messzi, messzi gépről felvéve :)

Szerénytelenségnek tűnhet, de mikor 2019 novemben (Csoki klubnapon) arra a kérdésre, hogy mit csinálunk most, azt találtam mondani ADT-nek (amit komolyan is gondoltam), hogy “a világ legjobb TVC-s programjának zenéjét csinálom”. Szerintünk az idő minket igazolt. És hogy nevezünk-e a 2020-as év játékfejlesztői versenyére? Meglátjuk. Ja! Már neveztünk! :D

DOBERDO BROTHERS