**Az IK+ sprite tömörítése**

A játék készítésekor több problémával szembesültünk, amire mindenképp megoldást kellett találni ahhoz, hogy a játék elkészülhessen. Egy olyan megoldás született, ami véleményem szerint ebben a játékban bizonyított, ugyan akkor biztos vagyok benne, hogy más programokban is használható.

A játékba bekerült megoldás nem tökéletes, több hátránya is van, mégis szerintem több az előnye, mint a hátránya.

Amit mindenképp meg kellett oldani:

1. A sprite adatait tömörítve kell eltárolni, mert tömörítetlenül esély sincs, hogy minden grafikai fázis beférjen a memóriába.
2. Mivel a mozgatandó sprite-ok nagy méretűek, ezért a lehető leggyorsabb legyen a kirajzolás.
3. A sprite-okat maszkolt technikával kell tudni kitenni a képernyőre (háttér bufferbe)
4. A sprite-okat függőleges tengely körül tükrözve is képesnek kell lenni megjeleníteni, hogy a balra ill. jobbra néző karakterek ne foglaljanak külön-külön memóriát.
5. A sprite-ok egyes színeit változtathatóan kell tudni kitenni, hogy a különböző színű figurák se foglaljanak dupla, tripla helyet.
6. A kitömörítés ne foglaljon plusz memória buffert, hanem egyből a képernyő területre (háttér bufferbe) történjen a kitömörítés. (lásd: sebesség mindenek felett!)

A kompromisszumok, amiket el kellett fogadnunk (Szerencsére ez a játék ezeket nem is kívánta meg.):

1. A sprite-okat nem tudjuk vágni.
2. Nem használhatjuk a sprite-ban mind a 16 színt egyszerre.

Az alkalmazott tömörítési eljárás a TVCDeveloperTool-ba lett beépítve, így az a teljes sprite atlaszt képes volt minden más segéd program nélkül exportálni a forráskódba.

A sprite-ok 16 színű módban jelennek meg, de csak 7+1 színt tárolnak. A legtöbb játék nem is igényli, hogy spriteonként több szín jelenjen meg. (A TVC hardver adottságait figyelembe véve)

A tömörített sprite adat szerkezete meglehetősen egyszerű:

Az első két byte a sprite szélessége byte-okban, ill. a magassága képernyő sorokban.

Ezután következik maga a tömörített adatszerkezet. Hogy a kitömörítéshez ne kelljen plusz memóriát használni, ezért soronként van letömörítve a sprite.

Minden byte-on 2 vagy több pixel adata van tárolva. Mivel csak 7 szín használata van megengedve, ezért így egyetlen pixel 3 bitet foglal el. Értelem szerűen itt szakítottunk a TVC képernyőn megjelenő byte szerkezettel. Itt a 3 bit szín indexet – egy virtuális palettára értelmezett indexet – jelöl.

Ha mindhárom bit 0, azaz a szín index 0 azt a kirakó átlátszó részként jeleníti meg.

Ha ezt két pixelt így egymás mellé tesszük a byte alsó ill. felső 4-4 bitjére akkor – mivel igazából 3-3 bit tartalmaz szín idexet – a legfelső 7-es bit 0.

Ennek a 7-es bitnek szerepe lesz, mert ez a bit jelzi magának a tömörítésnek a tényét.

A tömörítésnél azon byte-okon lehet nyerni, amik egyrészt ugyan azt a szín indexet tartalmazzák az alsó és a felső 4 bitjükön, másrészt több ilyen byte következik egymás után. Szerencsére a karatékák esetében akár az átlátszó részt vesszük figyelembe, akár magát az átlátszatlan részt elég sok esetben fenn áll a fentebb említett feltétel. (És ez még sok más játékban is igaz lehet).

Tehát, ha a 7-es bit 0, akkor a byte felső és alsó 4 bitje egy-egy pixel szín indexét jelöli.

Ha a 7-es bit 1, akkor a byte szerkezete a következő:

1. bit: jelző bit.

6-5-4-3. bit: egy 4 bites érték, megmondja, hogy az adott színindexű (mindkét pixel azonos színindexű) byte-ból mennyit kell egymás után rajzolni.

2-1-0. bit: maga a szín index.

Pofon egyszerű szerkezet viszont ha már csak 2 db azonos színű duplapixel következik egymás után máris nyertünk egy byte-ot!! Extrém esetben 15 duplapixelt tudunk egyetlen byte-on tárolni.

A kitömörítés soronként halad, egy sornak akkor van vége, mikor pontosan annyi byte-ot rajzoltunk ki, mint amennyi a sprite szélessége.

A sprite kirajzolás előtt definiáljuk magát a palettát. A paletta 8 elemű mivel 8 (7+1) különböző színindex tárolható 3 biten.

A paletta már a TVC hardver specifikációjának megfelelő formátumban tárolja a színeket.

Hogy a maszkolás könnyebb legyen 3x8 byte-on tároljuk a színeket.

Első 8 byte byte-onként a TVC hardvernek megfelelő első pixel bitjeit tartalmazza, formátuma:

7-6-5-4-3-2-1-0 biteken

3-x-2-x-1-x-0-x, ahol a számmal jelölt az első pixel bitjeit jelenti (IGRB), az x-el jelölt bitek 0-ák!!

A második 8 byte byte-onként értelem szerűen a hardver második pixelét prezentálja:

7-6-5-4-3-2-1-0 biteken

x-3-x-2-x-1-x-0, az x-el jelölt biteket itt is 0-ra kell állítani, a számok itt is az IGRB-nek megfelelő bitek

A harmadik 8 byte byte-onkénti felépítése: IIGGRRBB

7-6-5-4-3-2-1-0

3-3-2-2-1-1-0-0

A paletta első ill. második 8 byte-ját a tömörítetlen azaz két eltérő színű pixel kirakásánál használjuk úgy, hogy a byte-on az első pixel értékét az első táblázatból vesszük és a 0-s bitekre kerül az esetleges átlátszó pixel érték (azaz a háttér színe), a byte-on tárol második pixel értékét pedig a második 8 byte-os táblázatból vesszük.

A harmadik 8 byte-os táblázat viszont a tömörített adatok kitömörítésénél használjuk.

Ha tömörítetlen byte-tal van dolgunk, akkor az azon tárolt 2 pixel eltérő színű, annak megfelelően jelenítjük meg. Külön választva az első ill a második pixelt azokat ha kell külön-külön lemaszkolva tesszük a képernyőre (háttér bufferbe)

Ha a byte tömörített adatot tartalmaz, akkor két lehetőség van:

Ha az alsó 3 bit 0 – azaz a szín index 0 – akkor a byte 3-4-5-6. bitjén tárolt darabszámú byte-ot átugrunk a megjelenítésnél, mert az a rész átlátszó tehát nem kell kirajzolni.

Ha az alsó 3 biten tárolt szín index nem 0, akkor az a byte egyetlen pixel sem átlátszó, tehát kikeressük a harmadik táblázatból a megfelelő indexú byte-ot és a 3-4-5-6 bitjén tárolt értéknek

megfelelő darabot rámásolunk a kép vagy háttér buffer megfelelő pozíciójába.

Mivel a sprite-ok soronként eltérő számú byte-ot tartalmaznak, ezért az IK+-ban nem valósítottuk meg az ezen formátumú sprite-ok vágását. Természetesen megoldható a vágás is, de ez jóval bonyolultabb mint egy azonos sormérettel rendelkező sprite estében, több processzor időt fogyaszt, mint mikor a teljes sprite a látható képterületen van, ezért célszerű a vágáshoz külön sprite megjelenítőt írni. Jellemzően az X vágás a nehezebb feladat.