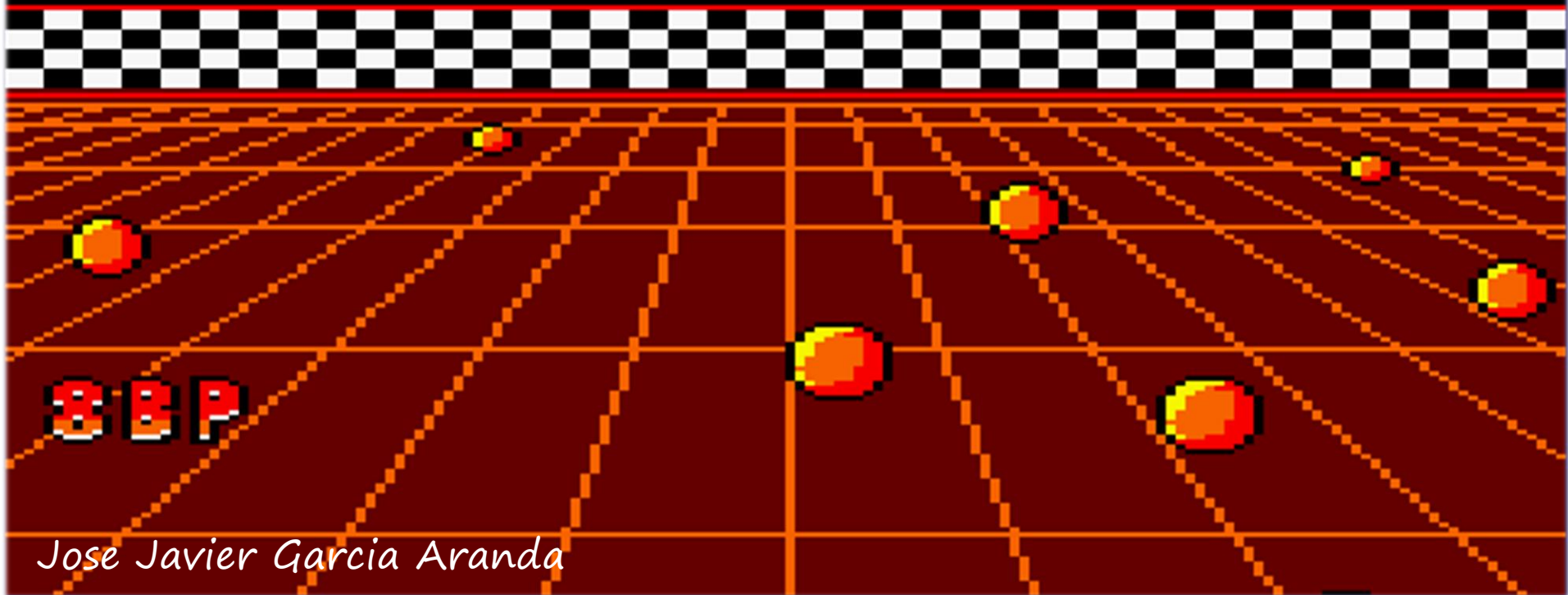
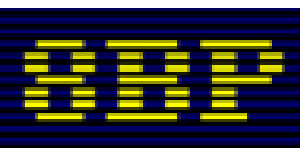


Programando videojuegos pseudo-3D con 8BP

8 BITS DE PODER



Jose Javier Garcia Aranda



8bitsdepoder.blogspot.com
<http://github.com/jjaranda13/8bp>

28 Abril 2018





AGENDA



1| *Concepto Pseudo-3D*

2| *historia y técnicas Hardware*

3| *historia y técnicas software en 8 bit*

4| *Introducción a 8BP*

5| *pseudo-3D en 8BP*

6| *programación avanzada y logicas masivas*

8BP

PSEUDO-3D

CONCEPTO:

Sólo es 3D el plano del suelo.

Las curvas son una ilusión.



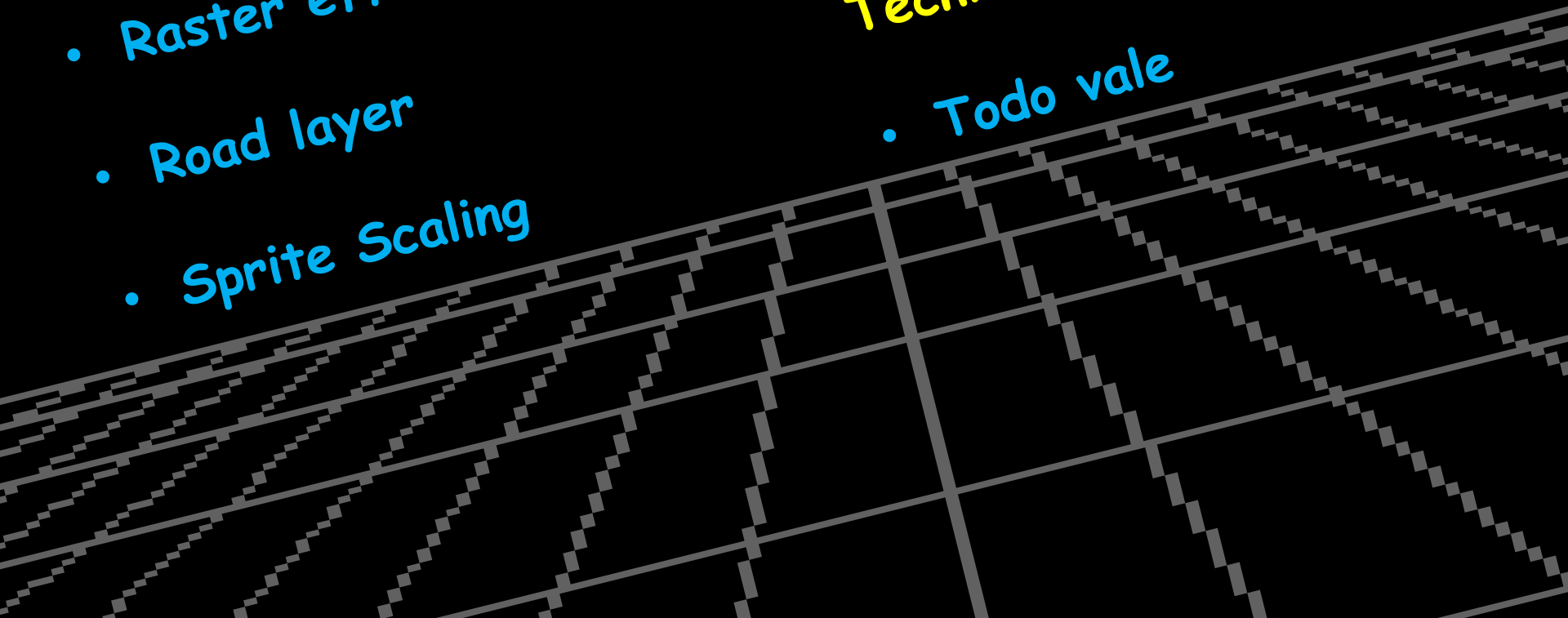
1| concepto pseudo-3D

Técnicas basadas en HW

- Raster effects
- Road layer
- Sprite Scaling

Técnicas basadas en SW

- Todo vale





AGENDA



1| Concepto Pseudo-3D

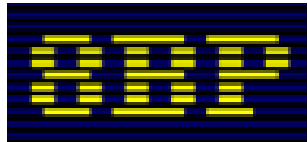
2| historia y técnicas Hardware

3| historia y técnicas software en 8 bit

4| Introducción a 8BP

5| pseudo-3D en 8BP

6| programación avanzada y logicas masivas



2| historia y técnicas HW del pseudo 3D

Pseudo-3D logrado con Raster effects sobre placa Capcom system 1



CAPCOM®

1991

RETRO
MADRID

2| historia y técnicas HW del pseudo 3D



Night driver Atari 1976



Turbo 1981 de Sega

SEGA[®]

Pole Position

1982 de Namco, distribuido por Atari

- Estableció la definición del género "3D Racing". Es **"la referencia"**
- Crearon la tecnología "Road layer" y "Sprite Scaling", en la que se basarían todos los siguientes juegos 3D
- Creado por **Toru Iwatani**



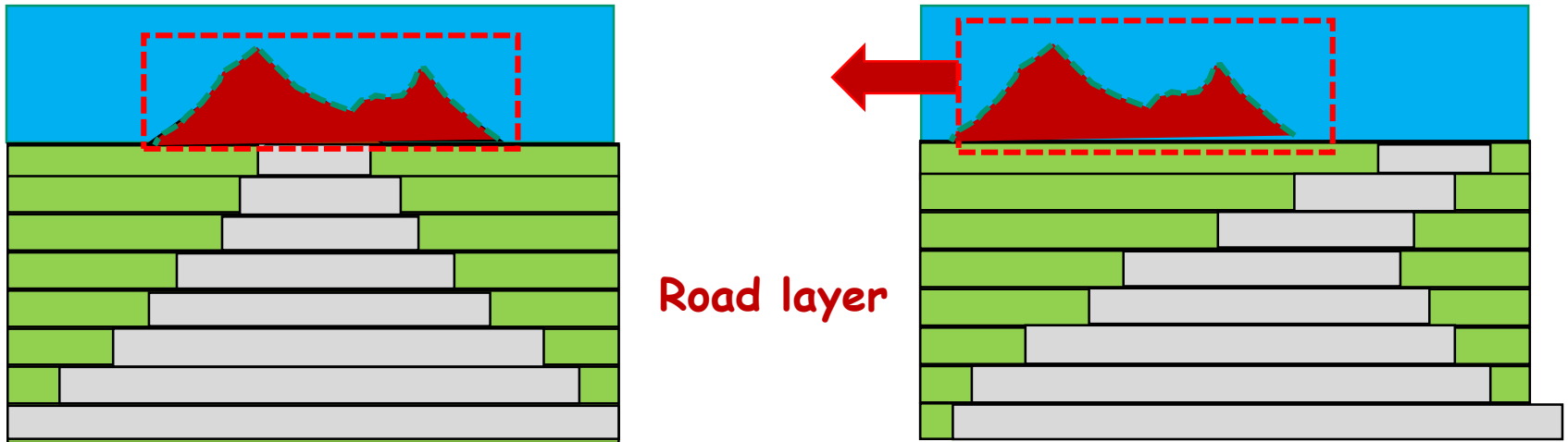
Namco pole position
arcade system board
Resolucion 256x224
Main CPU: Z80 (8bit)
Secondary CPUs: 2xZ8002
(16bit)
GPU : 7 chips

- **Road layer**
- **Sprite scaling**

2| historia y técnicas HW del pseudo 3D

Principios del pseudo 3D usados en HW en los 80s

Hardware dedicado para dibujar la carretera. Ejemplo :
Sega Road chip, usado en "Out run", Hang on, y Space Harrier



Road layer

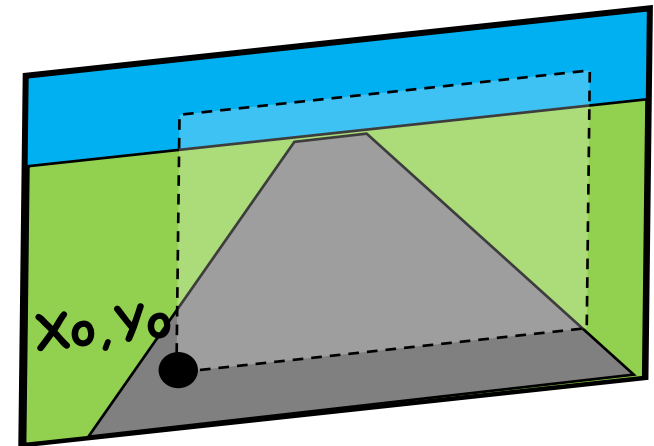
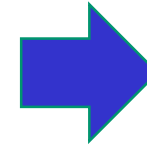
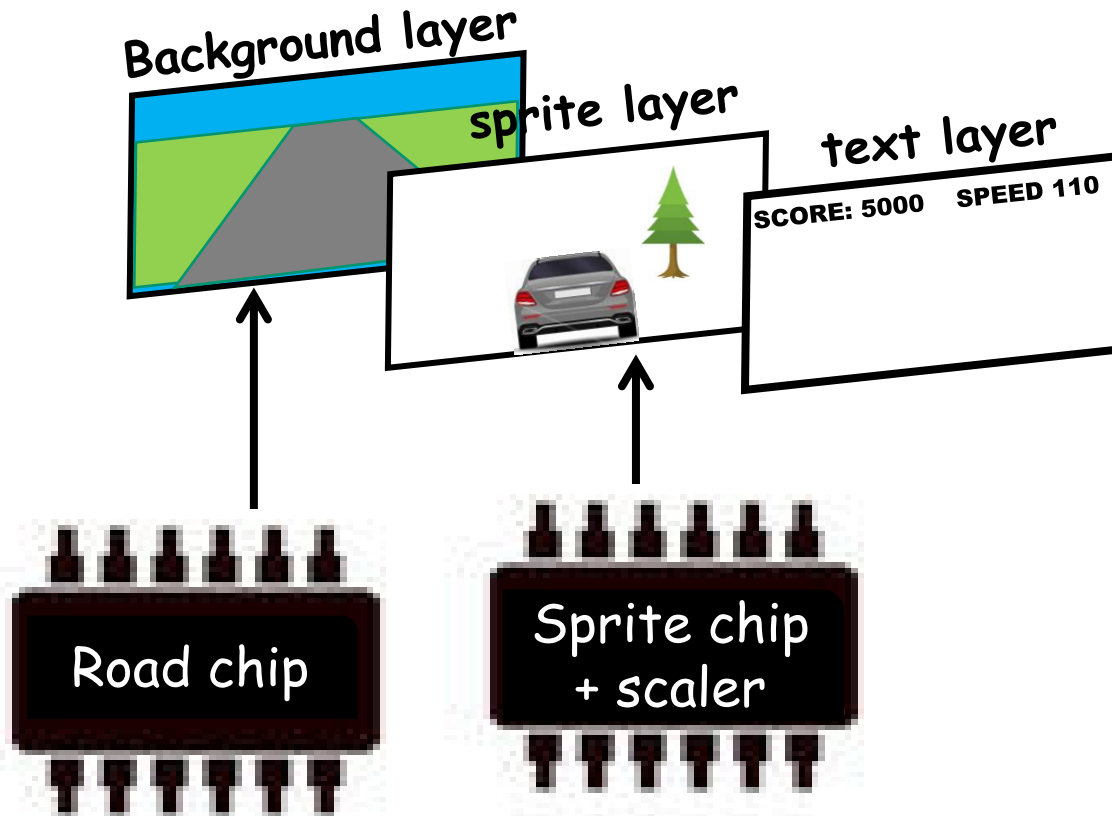
Las curvas...son "falsas"!!!, no hay nada que gire cuando tomas una curva.

Sprite scaling



2 | historia y técnicas HW del pseudo 3D

Principios del pseudo 3D usados en HW en los 80s



No se traslada a pantalla todo el background

INPUTS

- Parámetros para "torcer" la carretera
- Velocidad del suelo

INPUTS

- Coordenadas de cada Sprite (x, y, z)
- Escalado $= f(z)$

2| historia y técnicas HW del pseudo 3D



La Sega "super Scaler board" podía escalar miles de sprites por segundo. El Space Harrier (1985 de Sega) podía escalar 32K sprites por segundo



No busques curvas, No hay!!!!

2| historia y técnicas HW del pseudo 3D

En 1987 SEGA introdujo rotación de sprites por hardware en Afterburner

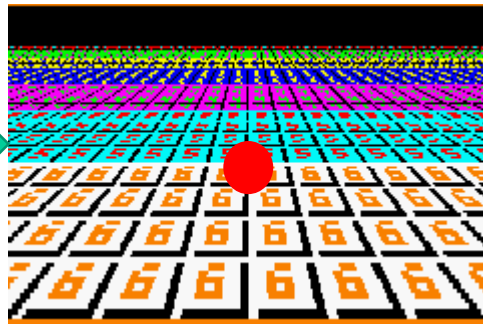
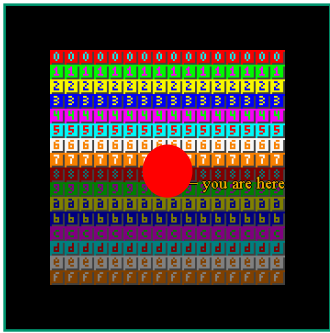


Los desniveles de la carretera en outrun se logran jugando con el road plane. Outrun tenía un solo road layer pero podía dibujar 2 carreteras, a diferencia de su antecesor, el hang on

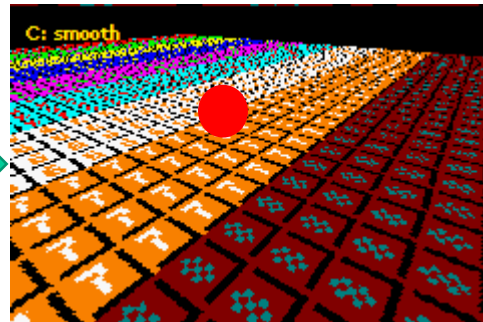
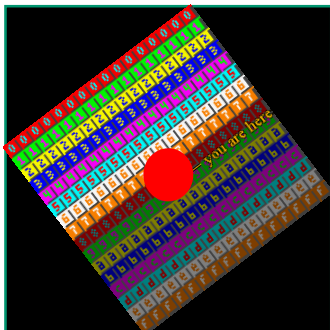
2| historia y técnicas HW del pseudo 3D

Rotación de sprites HW + Sprite scaling+ raster effects

El modo 7 de SNES (1990) es el resultado de combinar una rotación de sprites HW + un Sprite Scaling con efecto "raster". Para dar efecto de profundidad se cambia en cada línea el factor de escalado, haciendo aumentar las líneas cercanas al borde inferior de la pantalla. (La Sega super Scaler board ya lo hacía en 1987)



Al fin hay curvas reales!





AGENDA



1| Concepto Pseudo-3D

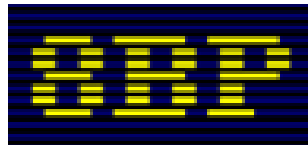
2| historia y técnicas Hardware

3| historia y técnicas software en 8 bit

4| Introducción a 8BP

5| pseudo-3D en 8BP

6| programación avanzada y logicas masivas



3 | historia y técnicas software en 8 bit

Pseudo-3D en 8-bit home computers:

- Al no haber aceleración HW, **el lema es "todo vale"**
- **Las curvas son falsas**, no hay nada que gire
- Solo se proyecta en 3D el plano de la carretera (y a veces ni eso)
- Los sprites se escalan cuando se alejan, 2 o 3 versiones (no hay Sprite scaling por HW aunque a veces se hace por SW)
- Muchos dan falsa impresión de velocidad con **rotación de tintas**
- Normalmente tienen **pocos fps**
- Programación muy limitada, inválida para otros juegos 3d



Veamos ejemplos!!!

"Turbo Esprit"



Poquísimos fps
No hay efecto de curvas

"Angel nieta pole 500"



Buenos gráficos, pero muy pocos FPS

"Gran Prix Rally II"



- Poquísimos FPS, trata de disimularlo sin éxito con animación de tintas
- Curvas poco convincentes

"3D grand prix"



- Buen Sprite scaling por software
- La animación por tintas esta bien integrada, dando sensación de mas velocidad

"Desert Fox"



Excelente fase de tanque con calculo de proyección real

“buggy boy”



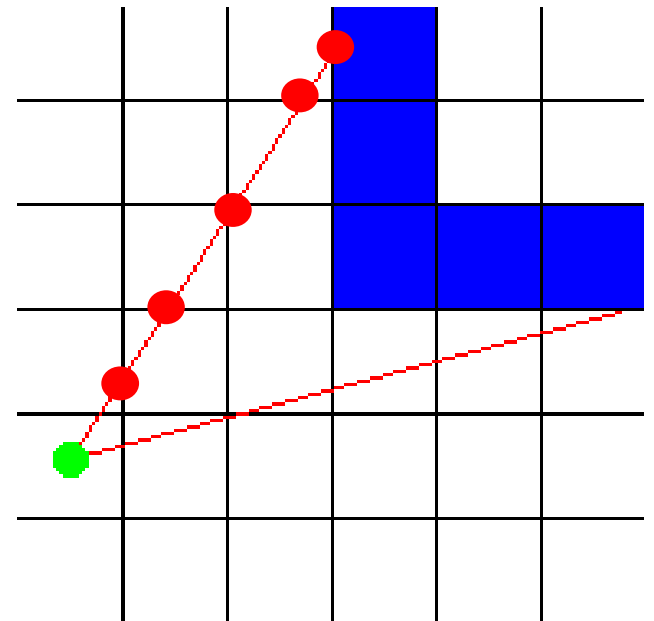
- Curvas con técnica RASTER por software
- Reducción de pantalla para poder hacer RASTER por software (54 Bytes x 60 líneas)
- no usa animación por tintas
- Buen Sprite scaling por software



Ray Casting

es más que PSEUDO 3D, es 3D con limitaciones

Usado como técnica software en ordenadores personales en los 90, aunque también ha habido ports a 8 bits. Nace con el juego "Hovertank" en 1991 pero se hace popular en 1992 con "wolfenstein 3D"



El mundo es una cuadrícula donde hay o no hay muro. No hay suelo ni techo. Se chequean las intersecciones. Si hay muro se pinta una línea centrada en el horizonte cuya longitud depende de la distancia a la que se encuentra dicha intersección. Demos más avanzadas incluso texturizan



AGENDA



1| Concepto Pseudo-3D

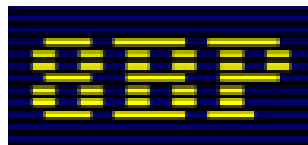
2| historia y técnicas Hardware

3| historia y técnicas software en 8 bit

4| **Introducción a 8BP**

5| pseudo-3D en 8BP

6| programación avanzada y logicas masivas



4 | introducción a 8BP

¿Por qué programar hoy en día una máquina de 1984?

Porque:

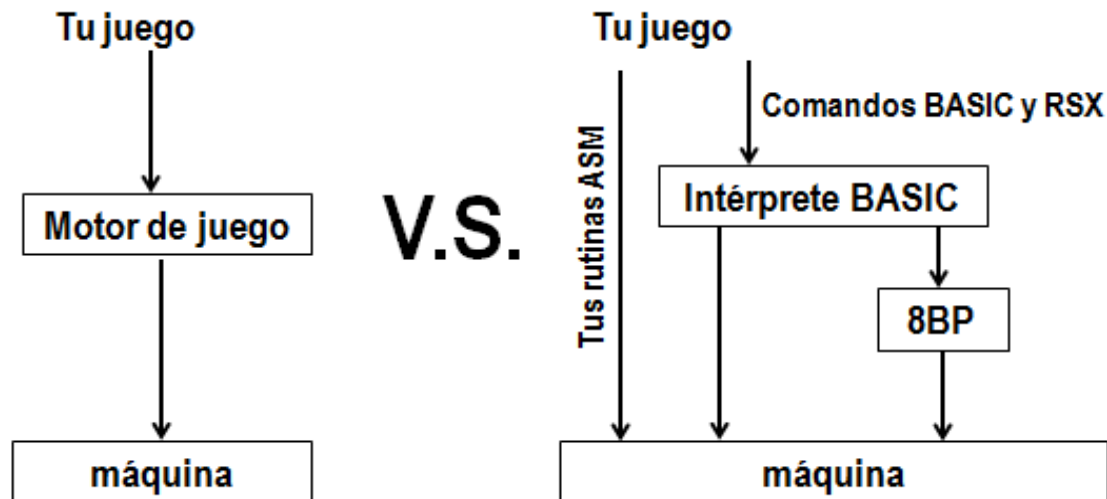
“Las limitaciones no son un problema, sino una fuente de inspiración”

JJGA.

CPU	Número de transistores	MIPS (millones de instrucciones por segundo)	Ordenadores y consolas que lo incorporan
6502	3.500	0.43 @1Mhz	COMMODORE 64, NES, ATARI 800...
Z80	8.500	0.58 @4Mhz	AMSTRAD, COLECOVISION, SPECTRUM, MSX...
Motorola 68000	68.000	2.188 @ 12.5 Mhz	AMIGA, SINCLAIR QL, ATARI ST...
Intel 386DX	275.000	2.1 @16Mhz	PC
Intel 486DX	1.180.000	11 @ 33 Mhz	PC
Pentium	3.100.000	188 @ 100Mhz	PC
ARM1176		4744 @ 1Ghz (1186 por core)	Raspberry pi 2, Nintendo 3DS, Samsung galaxy, ...
Intel i7	2.600.000.000	238310 @ 3Ghz (¡casi 500.000 veces más rápido que un Z80 !)	PC



4 | introducción a 8BP

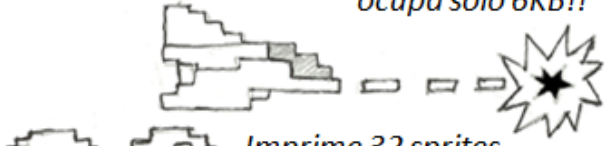


RSX es el acrónimo de *Resident System eXtensions*. Las librerías como 8BP que proporcionan comandos para extender el BASIC se les llama librerías RSX

- ¿Puedo usar 8BP desde Locomotive BASIC? si
- ¿Puedo usar 8BP desde lenguaje C? Si
- ¿Puedo usar 8BP desde ensamblador? Si
- ¿Puedo usar 8BP desde CPC Basic Compiler? si

¿qué es 8BP?

Es una librería RSX. Añade nuevos comandos a LOCOMOTIVE BASIC para hacer juegos de Calidad profesional. Y ocupa solo 6KB!!



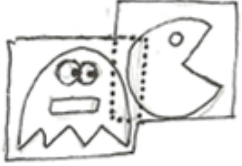
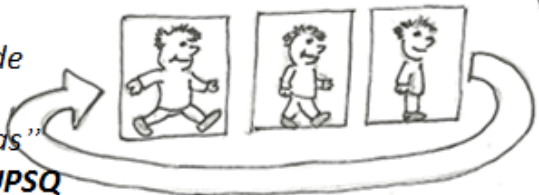
Imprime 32 sprites simultáneamente y controla su posición, velocidad, secuencia de animación, trayectoria...
|PRINTSP, |PRINTSPALL

S sprite	y	x	Vy	Vx
0				
1				
2				
...				
31				



Mueve grupos de enemigos en bloque
|MOVEALL, |AUTO, |AUTOALL

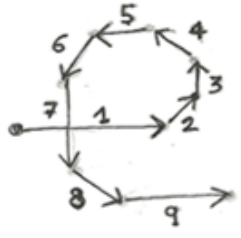
Crea secuencias de animación y "macrosecuencias"
|SETUPSP, |SETUPSQ



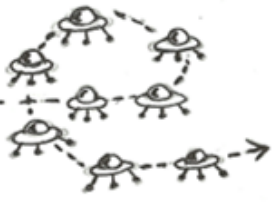
Detecta colisiones entre tus sprites
|COLSP, |COLSPALL

Soporta sobrescritura

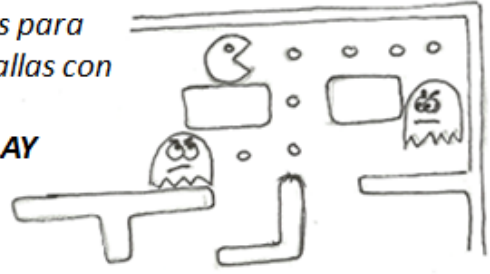
Soporta clipping **|SETLIMITS**



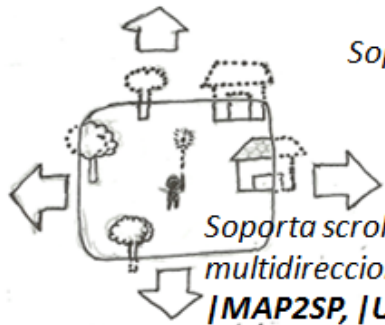
Crea tus trayectorias y asignaselas a tus sprites para que las recorran
**|ROUTEALL, |AUTOALL
|ROUTESP**



Crea tus layouts para juegos de pantallas con laberintos
|LAYOUT, |COLAY



Soporta pseudo3D
|3D



Soporta scroll multidireccional
|MAP2SP, |UMAP

Mueve grupos de estrellas para crear efectos de espacio, lluvia, tierra,...
|STARS



Edita y haz sonar tu música mientras juegas!
|MUSIC, |MUSICOFF



Y mucho mas, con 8BP!!!!

4 | introducción a 8BP

- 32 sprites con clipping, sobrescritura, ordenación y detección de colision.
- Comandos para mover N sprites a la vez (MOVERALL, AUTOALL, ROUTEALL...)
- Secuencias de animacion y macrosecuencias (cualquier sprite puede cambiar su secuencia de animacion dependiendo de su Vx,Vy)
- Enrutado de sprites automatico con rutas definibles (loops, saltos,...)
- **Scroll multidireccional (comando MAP2SP y UMAP)**
- Permite musica in-game basada en WYZtracker 2.0.1.0(comandos MUSIC and MUSICOFF)
- Capacidad de juegos con layout ("tile map"), con detección de colisión.
- Capacidad de animacion por tintas (RINK)
- Set de Minicaracteres definibles para usar en tus juegos (PRINTAT)
- Comando STAR para efectos de estrellas, tierra, Lluvia...
- **Capacidad PSEUDO-3D**
- Sólo ocupa 7.5KB y reserva 8.5KB para sprites y 1.3KB para musica, dejando 24.5KB para logica BASIC.

4 | introducción a 8BP

AMSTRAD CPC464 MAPA DE MEMORIA de 8BP

```
;
; &FFFF +-----
; | pantalla + 8 segmentos ocultos de 48bytes cada uno
; &C000 +-----
; | system (simbolos redefinibles, etc.)
; 42619 +-----
; | banco de 40 estrellas (desde 42540 hasta 42619 = 80bytes)
; 42540 +-----
; | map layout de caracteres (25x20 =500 bytes)
; | y mapa del mundo (hasta 82 elementos caben en 500 bytes)
; | ambas cosas se almacenan en la misma zona de memoria
; | porque o usas una o usas otra
; 42040 +-----
; | sprites (hasta 8.5KB para dibujos).
; |   dispones de 8540 bytes si no hay secuencias ni rutas)
; |   aquí tambien se almacenan las imágenes del alfabeto
; +-----
; | definiciones de rutas (de longitud variable cada una)
; +-----
; | secuencias de animacion de 8 frames (16 bytes cada una)
; | y grupos de secuencias de animacion (macrosecuencias)
; 33500 +-----
; | canciones
; |   (1300 Bytes para musica editada con WYZtracker 2.0.1.0)
; 32200 +-----
; | rutinas 8BP (7250 bytes)
; |   aquí estan todas las rutinas y la tabla de sprites
; |   incluye el player de musica "wyz" 2.0.1.0
; 24500 +-----
; | variables el BASIC
; | V
; |
; | ^ BASIC (texto del programa)
; |
; 0 +-----
```

The logo for 8BP, consisting of the letters '8BP' in a stylized, pixelated font. The '8' is a double-digit '8', and the 'P' has a long vertical tail. The letters are yellow on a dark blue background.

4 | introducción a 8BP

sprite	1byte status	2 bytes coordy	2 bytes coordx	1byte vy	1byte vx	1byte seq	1byte frame	2 bytes imagen	1byte ruta
0	27000	27001	27003	27005	27006	27007	27008	27009	27015
1	27016	27017	27019	27021	27022	27023	27024	27025	27031
2	27032	27033	27035	27037	27038	27039	27040	27041	27047
3	27048	27049	27051	27053	27054	27055	27056	27057	27063
4	27064	27065	27067	27069	27070	27071	27072	27073	27079
5	27080	27081	27083	27085	27086	27087	27088	27089	27095
6	27096	27097	27099	27101	27102	27103	27104	27105	27111

Byte de estado indica para cada sprite sus flags activos

7	6	5	4	3	2	1	0
ROUTEALL lo ruta	Sobre- escritura	COLSPALL collider	MOVERALL lo mueve	AUTOALL lo mueve	ANIMALL lo anima	COLSP collided	PRINTSPALL lo imprime

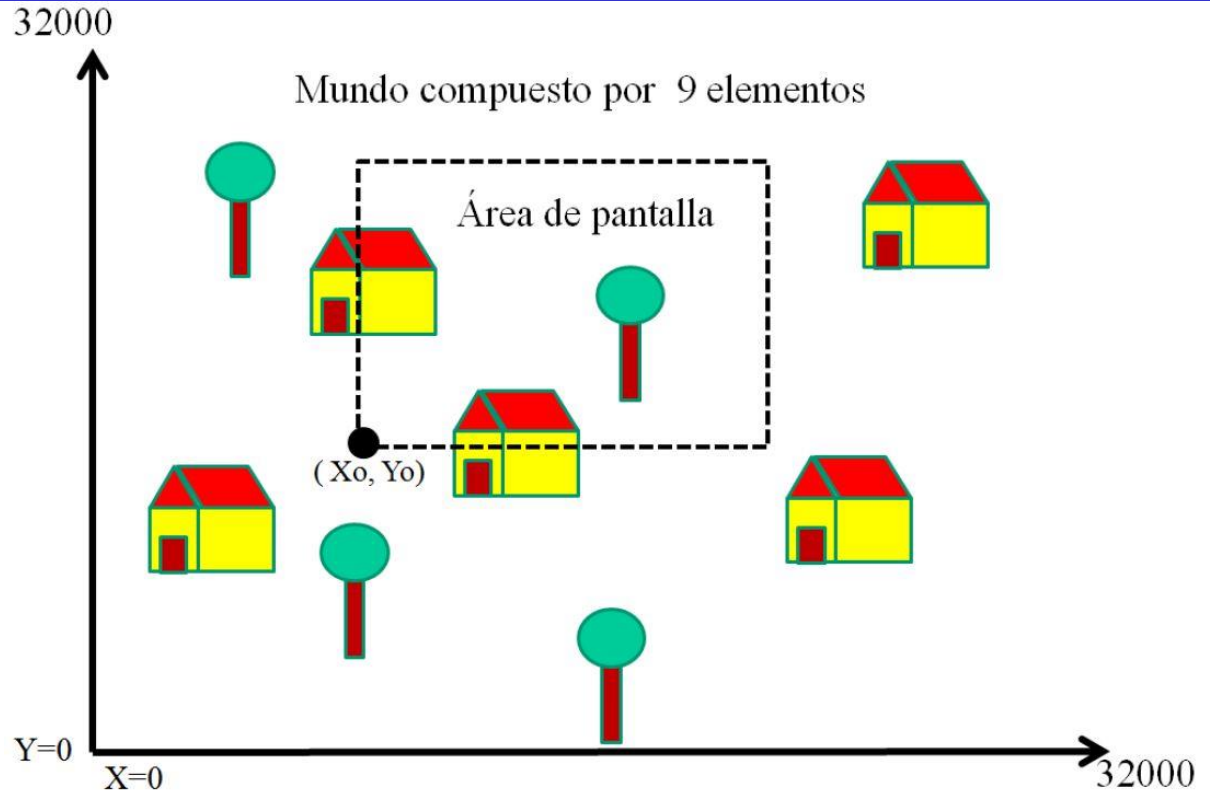
13	27208	27209	27211	27213	27214	27215	27216	27217	27223
14	27224	27225	27227	27229	27230	27231	27232	27233	27239
15	27240	27241	27243	27245	27246	27247	27248	27249	27255
16	27256	27257	27259	27261	27262	27263	27264	27265	27271
17	27272	27273	27275	27277	27278	27279	27280	27281	27287
18	27288	27289	27291	27293	27294	27295	27296	27297	27303
19	27304	27305	27307	27309	27310	27311	27312	27313	27319
20	27320	27321	27323	27325	27326	27327	27328	27329	27335
21	27336	27337	27339	27341	27342	27343	27344	27345	27351
22	27352	27353	27355	27357	27358	27359	27360	27361	27367
23	27368	27369	27371	27373	27374	27375	27376	27377	27383
24	27384	27385	27387	27389	27390	27391	27392	27393	27399
25	27400	27401	27403	27405	27406	27407	27408	27409	27415
26	27416	27417	27419	27421	27422	27423	27424	27425	27431
27	27432	27433	27435	27437	27438	27439	27440	27441	27447
28	27448	27449	27451	27453	27454	27455	27456	27457	27463
29	27464	27465	27467	27469	27470	27471	27472	27473	27479
30	27480	27481	27483	27485	27486	27487	27488	27489	27495
31	27496	27497	27499	27501	27502	27503	27504	27505	27511



4 | introducción a 8BP

Scroll basado en una "ventana deslizante"

Comando
|MAP2SP, Yo, Xo



```
;MAP TABLE
```

```
; primero 3 parametros antes de la lista de "map items"  
dw 50; maximo alto de un sprite por si se cuela por arriba y ya hay que pintar parte de el  
dw -40; máximo ancho de un sprite por si se cuelo por la izquierda (numero negativo)  
db 64; numero de elementos del mapa.como mucho debe ser 64
```

```
; a partir de aqui comienzan los items
```

```
dw 100,10,CASA; 1  
dw 50, -10,CACTUS; 2  
dw 210,0,CASA; 3  
dw 200,20,CACTUS; 4  
dw 100,40,CASA; 5  
dw 160,60,CASA; 6  
dw 70,70,CASA; 7  
dw 175,40,CACTUS; 8  
dw 10,50,CASA; 9  
dw 250,50,CASA; 10  
dw 260,70,CASA; 11  
dw 290,60,CACTUS; 12  
dw 180,90,CASA; 13  
dw 60,100,CASA; 14
```

Mapa del mundo

Puedes tener muchas fases y pokear el mapa al entrar en cada fase, o invocar a |UMAP

6 | scroll:

31

30

29

28

muñecos

...

4

3

2

1

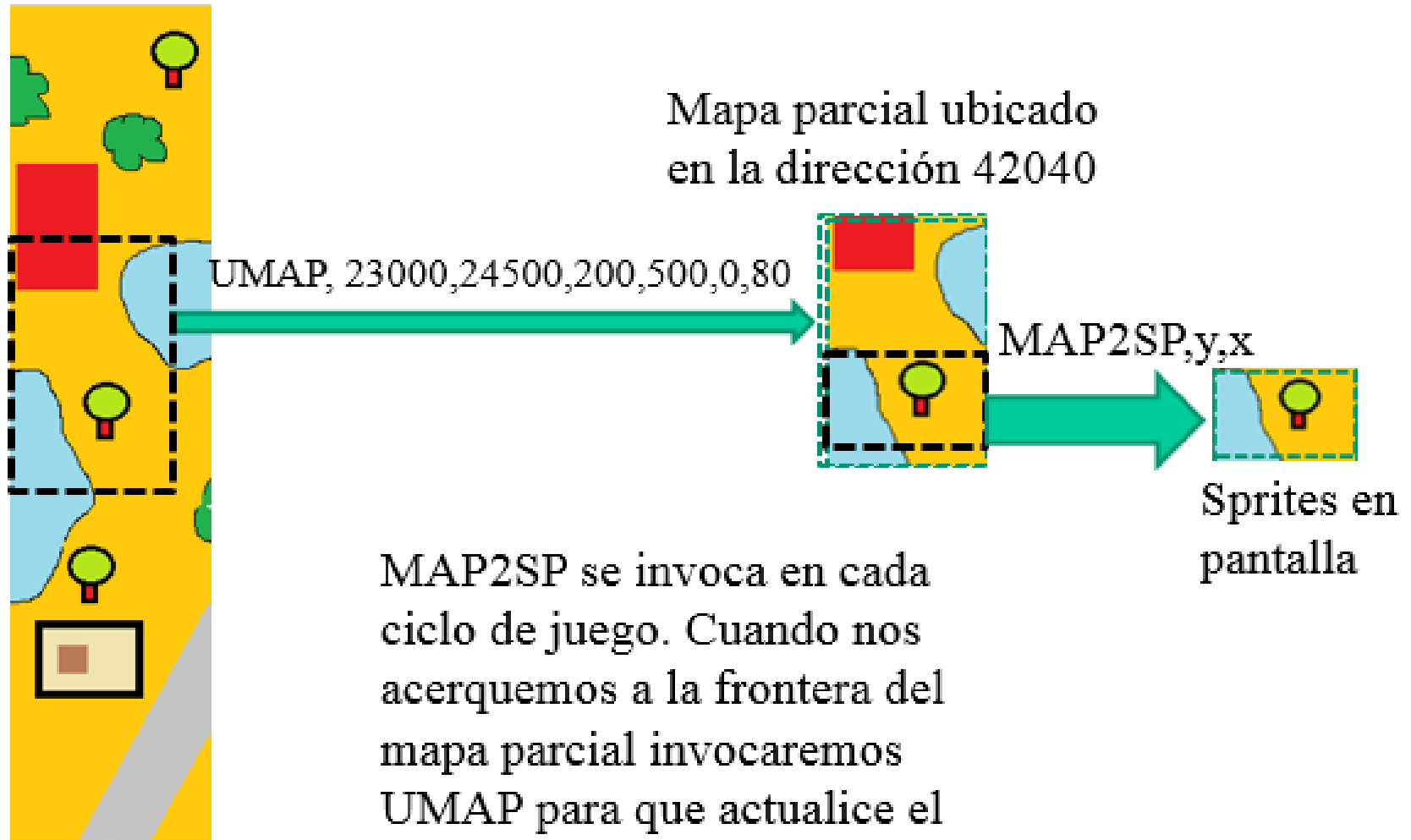
Generados
por
MAP2SP
Montañas,
casas...



4 | introducción a 8BP

Mapa completo

Dirección 23000 (por ejemplo)



MAP2SP se invoca en cada ciclo de juego. Cuando nos acerquemos a la frontera del mapa parcial invocaremos UMAP para que actualice el mapa parcial



AGENDA



1| *Concepto Pseudo-3D*

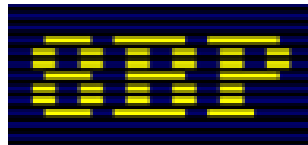
2| *historia y técnicas Hardware*

3| *historia y técnicas software en 8 bit*

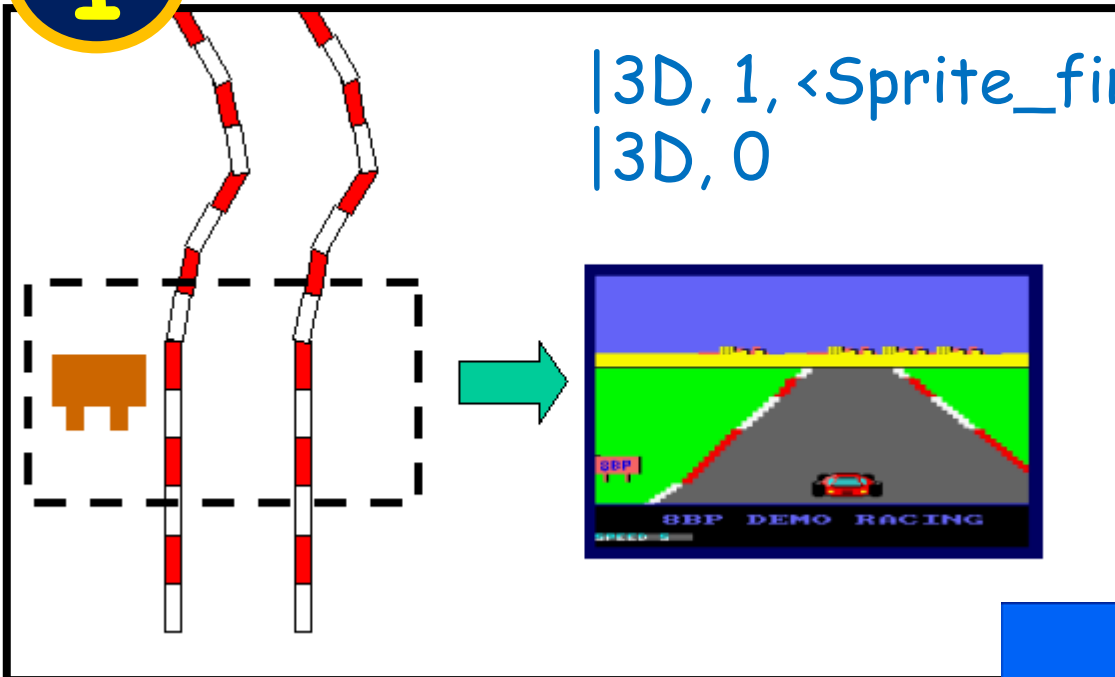
4| *Introducción a 8BP*

5| *pseudo-3D en 8BP*

6| *programación avanzada y logicas masivas*



1 Proyección 3D de la ventana deslizante



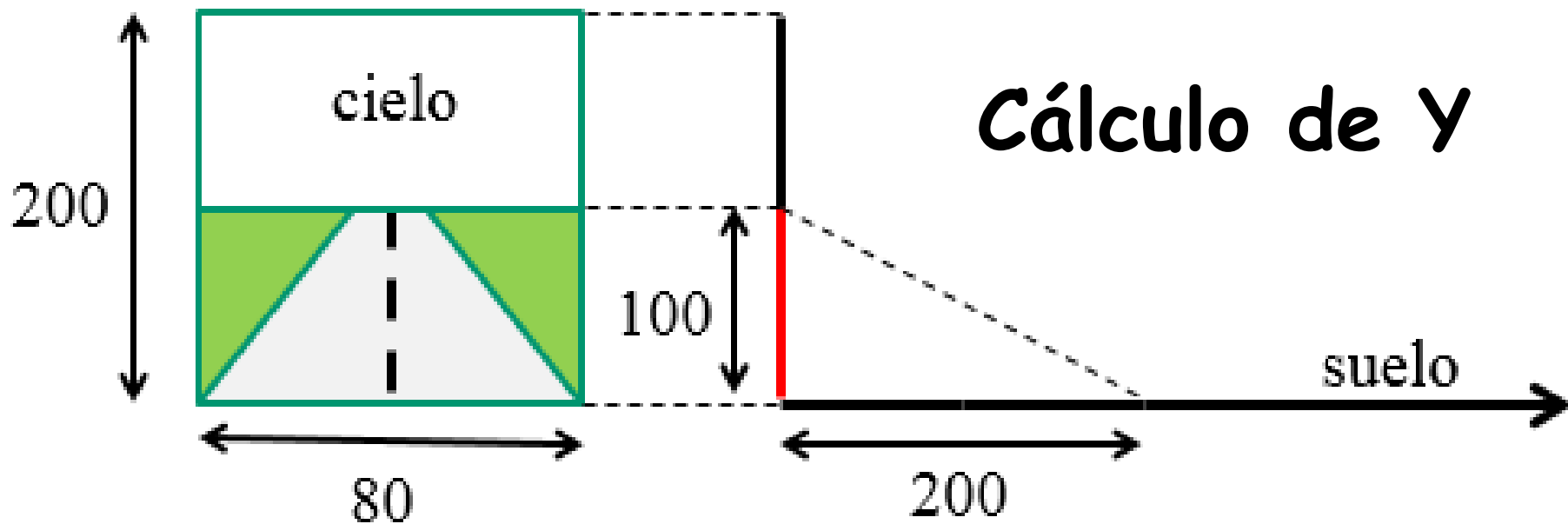
|3D, 1, <Sprite_fin>, <offsety>
|3D, 0

2 Zoom images

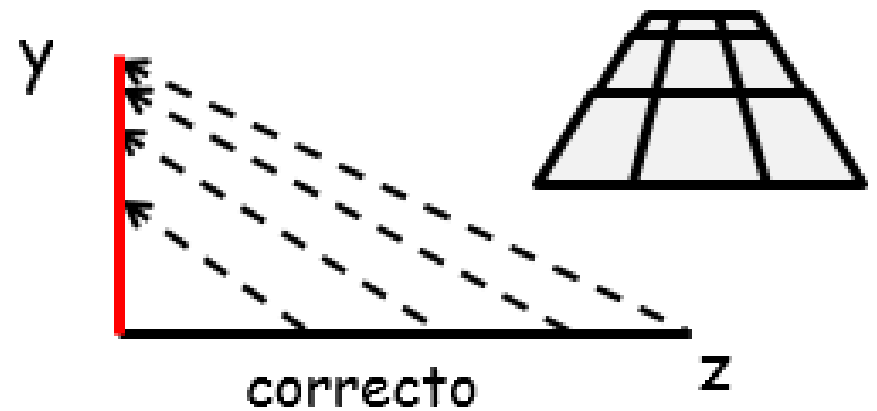
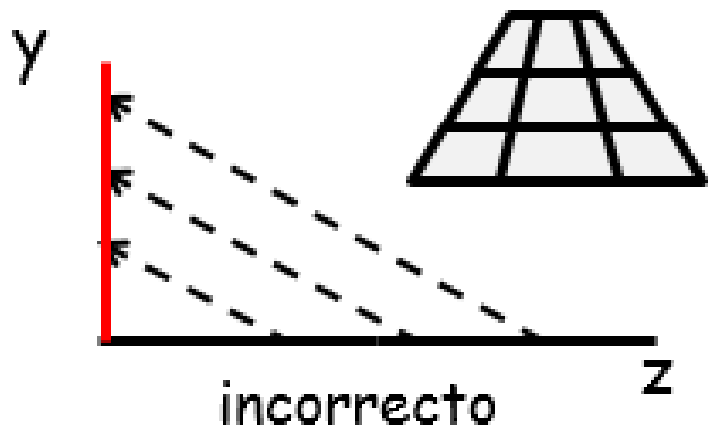


3 Segmentos

1 Proyección 3D de la ventana deslizante



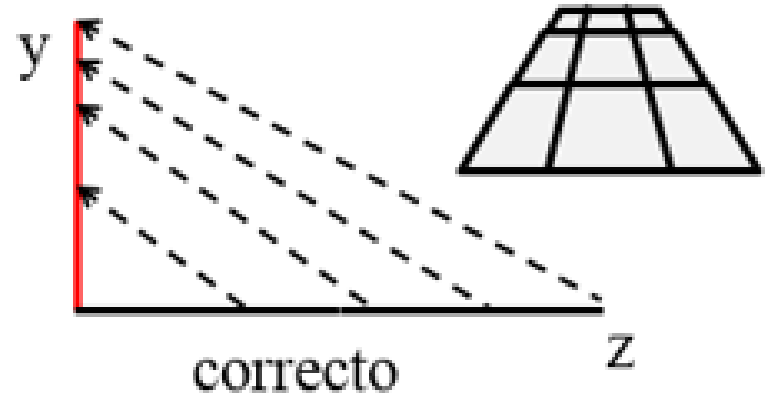
1 Proyección 3D de la ventana deslizante



1 Proyección 3D de la ventana deslizante

Cálculo de y

$$dy \text{ (inicial)} = dz$$



Ahora bien, si de nuevo nos alejamos dz , el incremento dy debe de disminuir, y si volvemos a alejarnos dz , el dy que debemos sumar cada vez será mas pequeño. Es decir:

Cada vez que nos alejamos dz , sumamos un incremento a "y" que cada vez es menor

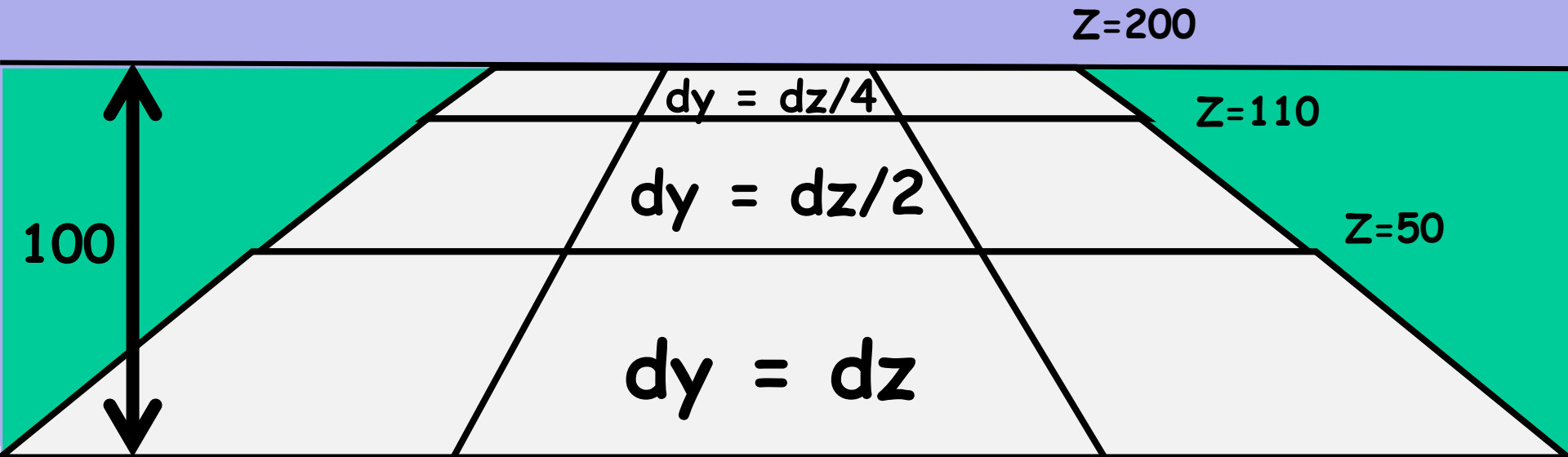
$$z = z + dz$$

$$dy = dy - ddy$$

$$y = y + dy$$

1 Proyección 3D de la ventana deslizante

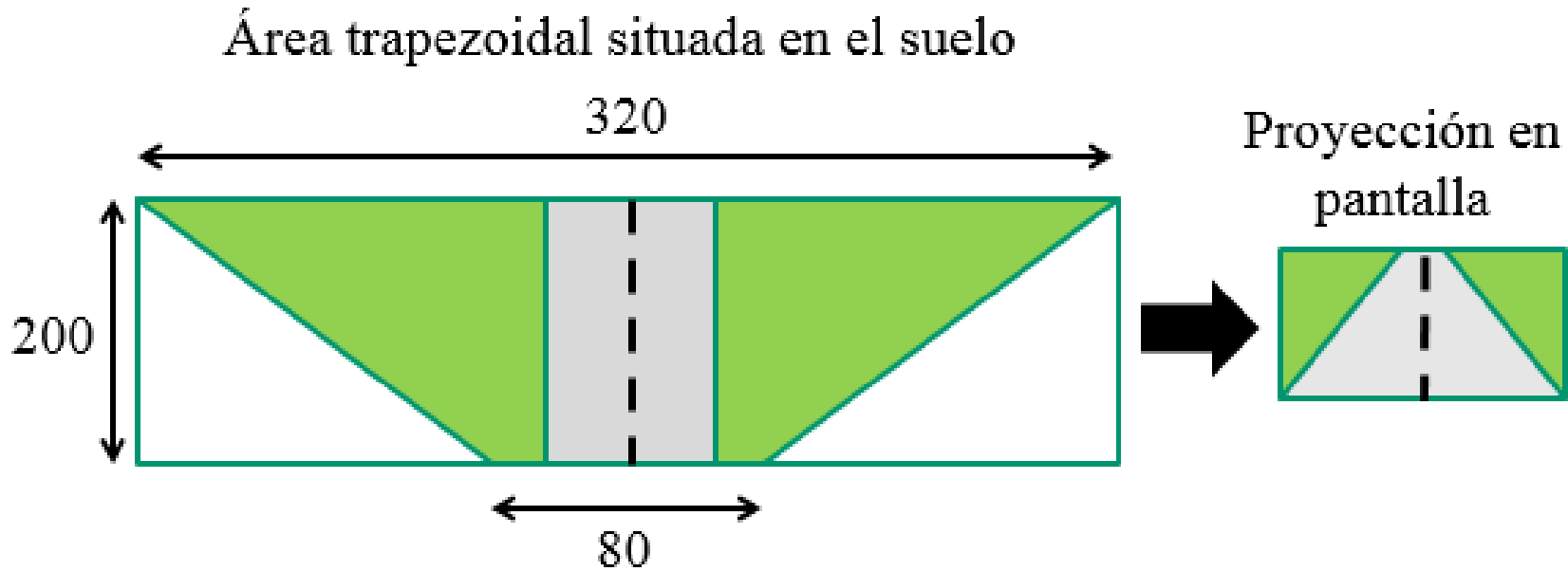
Cálculo de Y



Aproximación 8BP con resultados convincentes

1 Proyección 3D de la ventana deslizante

Cálculo de X



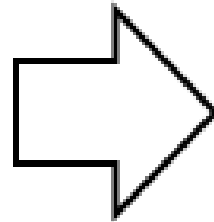
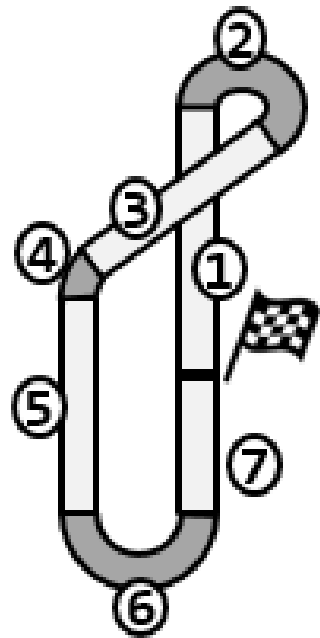
$$\text{Factor} = ((100 - y) + 32) / 2$$

$$x = (x - \text{centro}) * \text{Factor} + \text{centro}$$

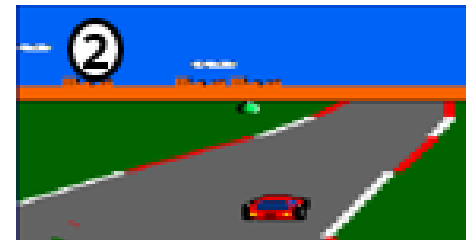
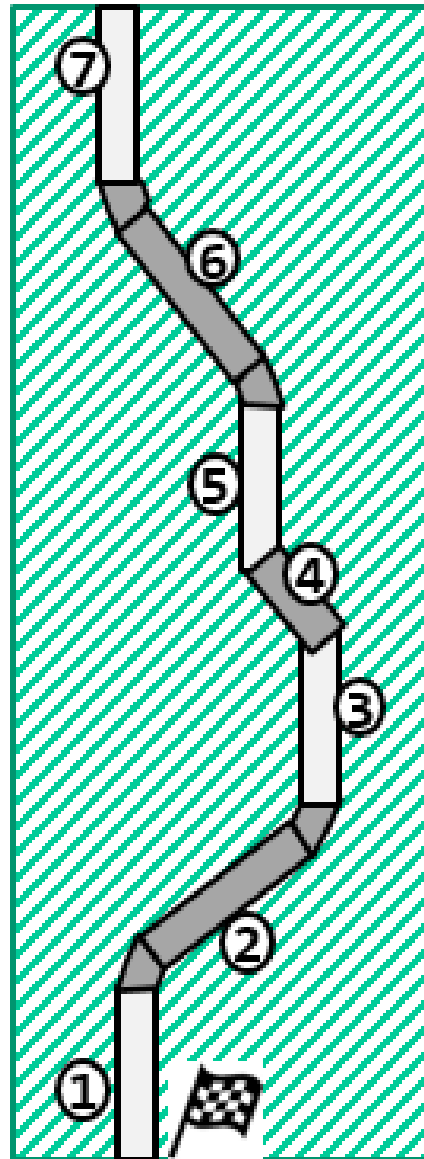
si $z=200$ entonces $y=100$, y entonces factor = 16

si $z=0$ entonces $y=0$, y entonces factor = 64 (4 veces mas)

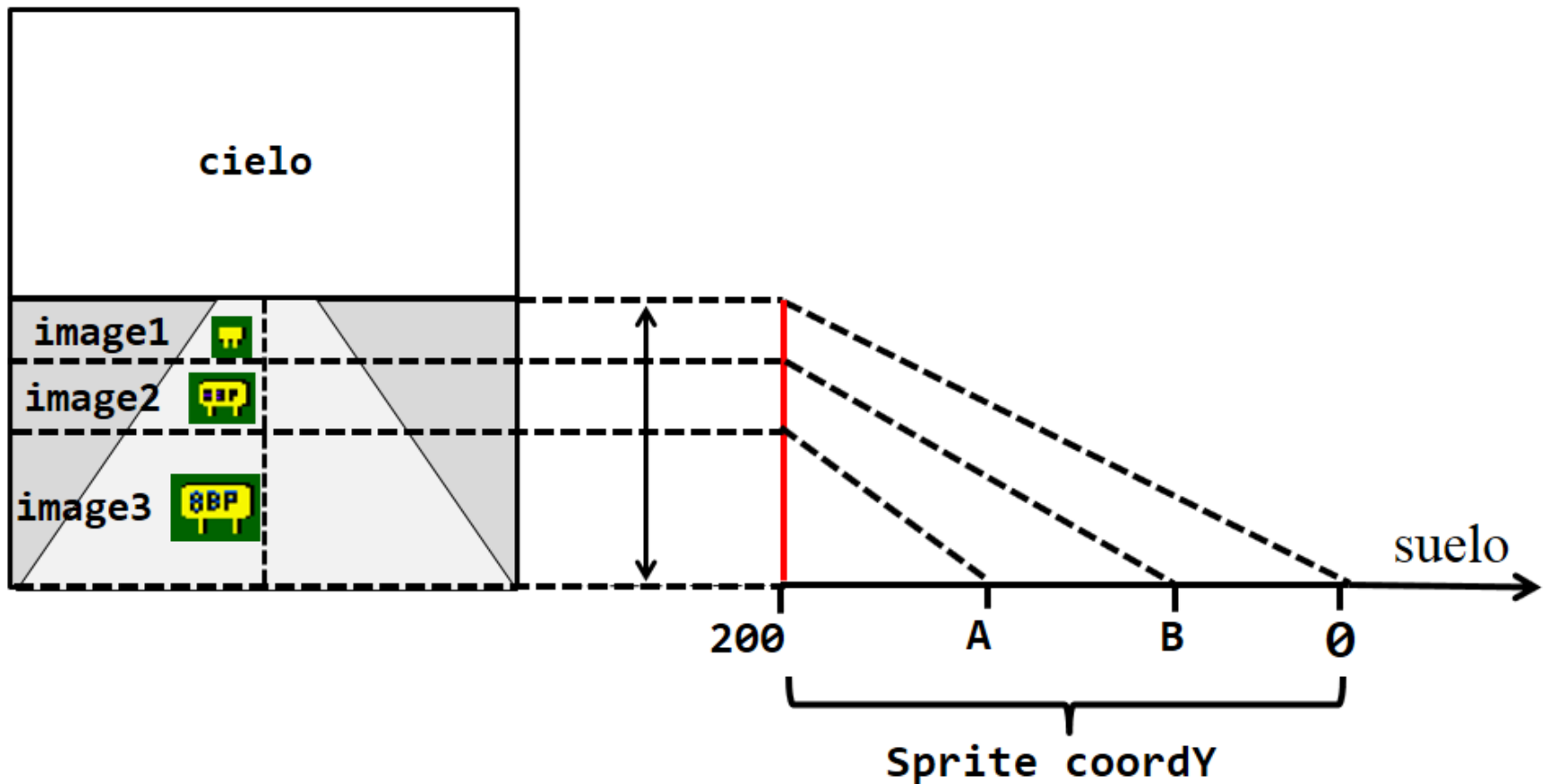
curvas



2D MAP



2 Zoom images



2 Zoom images

_3D_ZOOM_IMAGES

=====

;

; limites aplicables a todas las imagenes con zoom

; para estos limites se considera el horizonte como el 0 y hacia abajo va creciendo hasta 200

_ZOOM_LIMIT_A

db 120; entre 200 (suelo) y limitA se pone imagen 3

_ZOOM_LIMIT_B

db 50

; entre este limite y el limite A, se pone imagen 2

; mas cerca del horizonte que limit B se pone imagen 1

=====


CARTEL_ZOOM

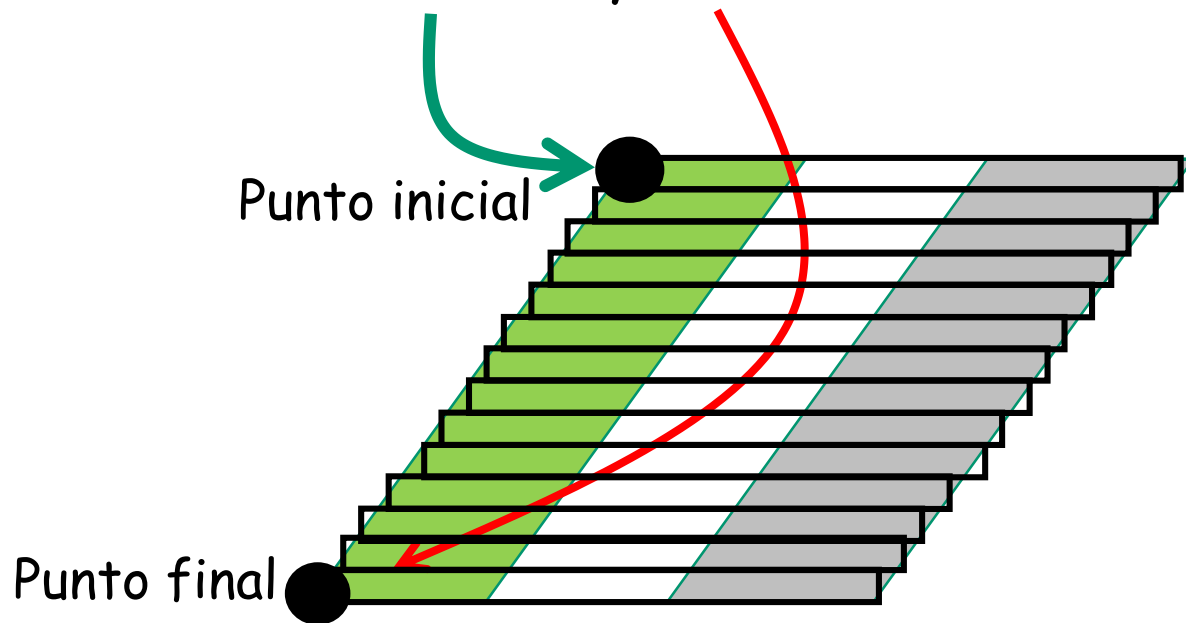
db 1; ancho simbolico

db 1; alto simbolico

dw CARTEL1, CARTEL2, CARTEL3

3 Segmentos

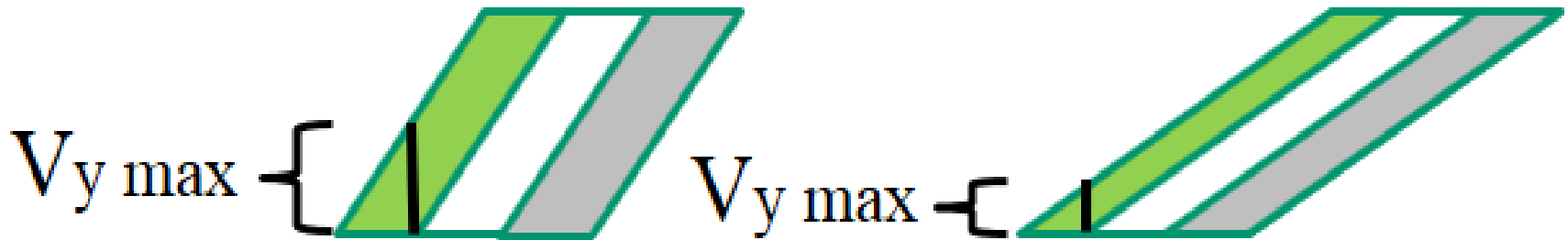
- Se definen con **una sola** scanline horizontal 
- Se **proyectan** sólo las coordenadas inicial y final



- **Anchura constante**, no se estrecha con la distancia

3 Segmentos

- Mas velocidad máxima cuanto menos torcido esté
- Más velocidad máxima cuanto mas grueso (mas margen de borrado)



5 | pseudo-3D en 8BP

3D RACING ONE



massive
logics

2018 JOSE JAVIER GARCIA ARANDA
8BP: THE ULTIMATE RSX LIBRARY FOR GAMES
www.github.com/jjaranda13/8bp
www.8bitsdepoder.blogspot.com



3D RACING ONE

SPEED HOTSP TIME: 11

PSEUDO 3D

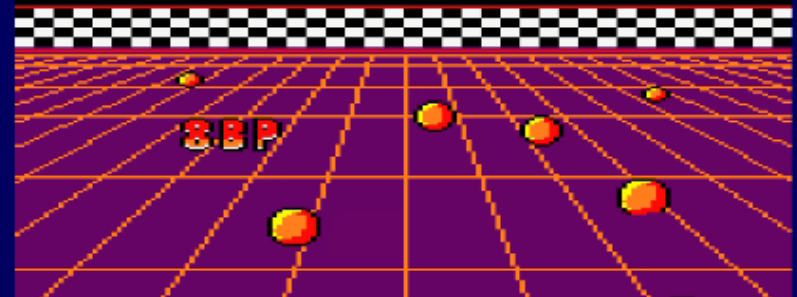


▶ || > ⌂ 🗂 📄 📁 📂 📅 📆 📇 📈 📉 📊 📋 📌 📍 📎 📏 📐 📑 📒 📓 📔 📕 📖 📗 📘 📙 📚 📛 📜 📝 📞 📟 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿

Speed: 100% FPS: 50

3D RACING ONE

BY JOSE JAVIER GARCIA ARANDA 2018
A FULLY BASIC PROGRAM CREATED WITH 8BP





AGENDA



1| *Concepto Pseudo-3D*

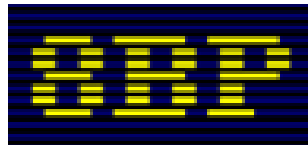
2| *historia y técnicas Hardware*

3| *historia y técnicas software en 8 bit*

4| *Introducción a 8BP*

5| *pseudo-3D en 8BP*

6| *programación avanzada y logicas masivas*



2 | lógicas masivas

Es reducir la complejidad computacional de orden N a orden 1 , con astucia, imponiendo restricciones que no sean perceptibles, aparte del uso de comandos que afectan a varios sprites.

Escuadrones:

|MOVERALL,dy,dx en lugar de |MOVER
|AUTOALL en lugar de |AUTO
|COLSPALL en lugar de |COLSP
|ROUTEALL



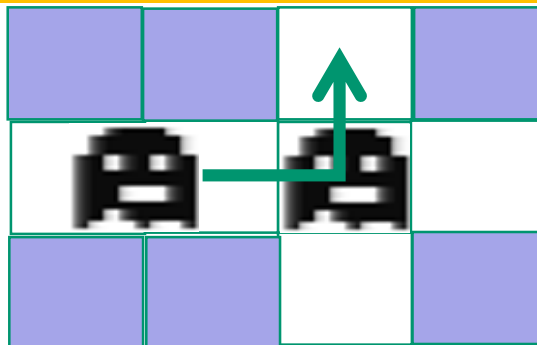
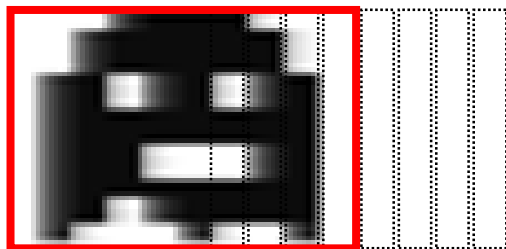
Ejecución de menos comprobaciones

- En cada ciclo de juego ejecutar un subconjunto de comprobaciones
- Todas las comprobaciones tardarán varios frames pero no importa

Ejecución de una sola lógica

- En cada ciclo de juego ejecutar la "Inteligencia" de un solo enemigo
- En juegos de laberintos podemos adaptar el mapa a la inteligencia

$t = 0$ 01234567



$t = 5$

Mide el tiempo que consume cada instrucción de tu programa y ante todo trata de ahorrar instrucciones en cada ciclo de juego

```
10 MEMORY 25999
20 DEFINT a-z
30 a!= TIME
40 FOR i=1 TO 1000
50 <aqui pones un comando, por ejemplo PRINT "A">
60 NEXT
70 b!=TIME
80 PRINT (b!-a!)
900 c!=1000/((b!-a!)*1/300)
100 PRINT c, "fps"
110 d!=c!/60
120 PRINT "puedes ejecutar ",d!, "comandos por barrido (1/50 seg)"
125 rem si dejas la linea 50 vacia , tardara 0.47 milisegundos
130 PRINT "el comando tarda ";((b!-a!)/300-0.47);"milisegundos"
```


8 | recomendaciones

- Usa DEFINT A-Z (y asigna en hexa si valor > 32000)
- No uses PRINT en cada ciclo (pero si puedes usar PRINTAT)
- Usa mucho GOTO, es mas rápido incluso que REM
- Evita el paso de parámetros (8BP lo permite)
- IF a > b and c > d then → IF a > b then if c > d then
- If a > 0 then → if a then (ahorras 0.5ms)
- Usa POKE cuando puedas hacerlo en lugar de SETUPSP
- Si necesitas comprobar algo no lo hagas en todos los ciclos de juego. Usa lógica modular (MOD) y operaciones lógicas "AND"
- Aprovecha bien la memoria, reutiliza líneas

Técnica	Tiempo consumido
A = A+1: if A = 5 then A=0: GOSUB <rutina>	2.6 ms
IF ciclo MOD 5 = 0 THEN gosub rutina	1.84ms, suponiendo que ya tienes una variable llamada ciclo que se actualiza

Técnica	Tiempo consumido
If ciclo AND 15=0 then gosub rutina	1.6 ms (se ejecuta una de cada 16 veces)
If ciclo AND 1=0 then gosub rutina	1.6ms (Se ejecuta una de cada 2 veces)

Y usa lógicas masivas!!!!

Empieza la aventura de programar!

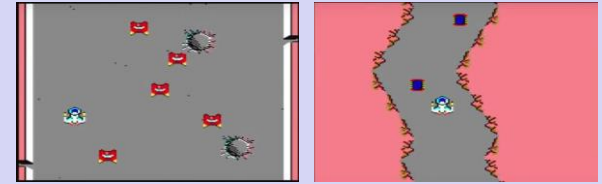
Nibiru



Mini invaders



Anunnaki



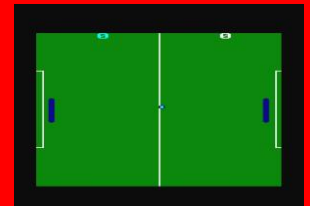
Mutante Montoya



Fresh Fruits & vegetables



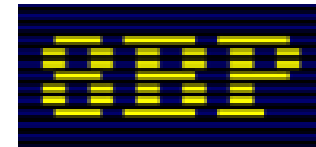
Mini pong



3D Racing one



Todos los juegos mostrados están programados en Locomotive BASIC Y ejecutados sin necesidad de compilar



Jose Javier Garcia Aranda

8bitsdepoder.blogspot.com

<http://github.com/jjaranda13/8bp>

