INTRODUCCION A 8BP

Y

Como programar:

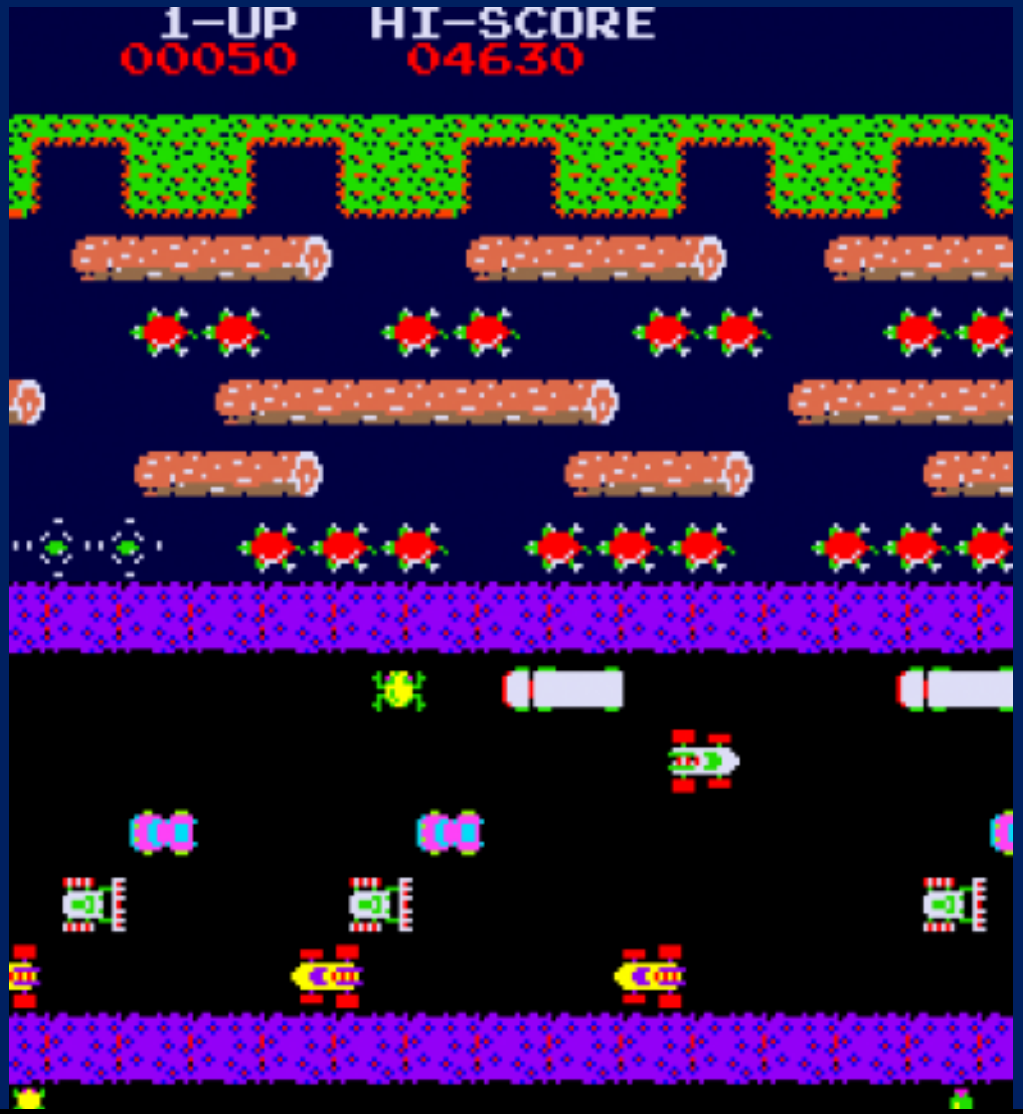
2019 JOSE JAVIER GARCIA ARANDA

8BP: THE ULTIMATE RSX LIBRARY FOR GAMES

www.github.com/jjaranda13/8BP

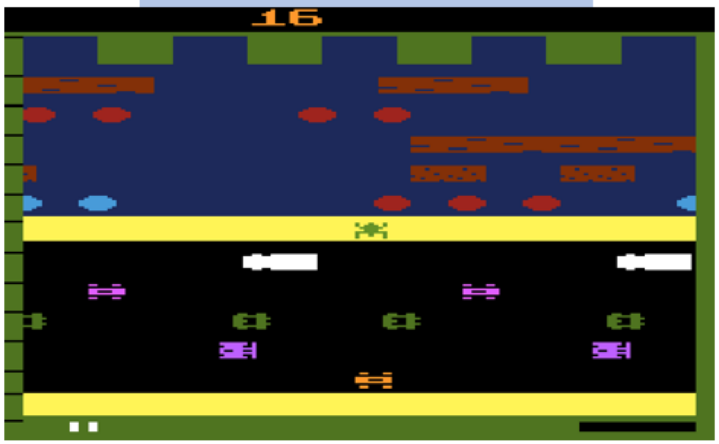
www.8bitSdepoder.blogSpot.com

Historia de frogger



Frogger es considerado como uno de los 10 mejores videojuegos de todos los tiempos según Killer List of Videogames (KLOV)

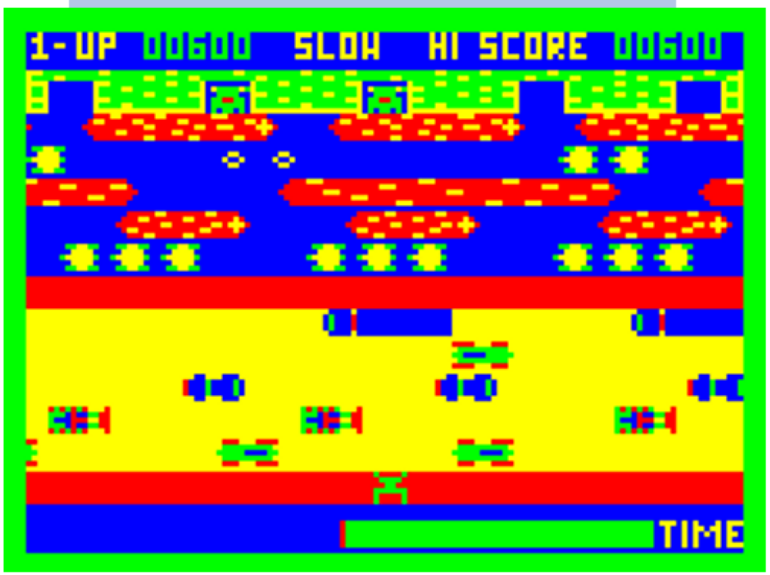
Atari 2600



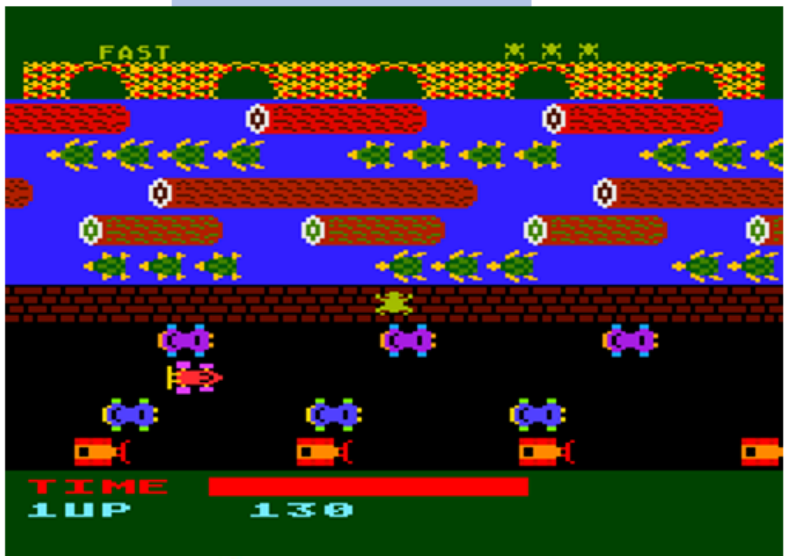
COLECOVISION

Vic 20

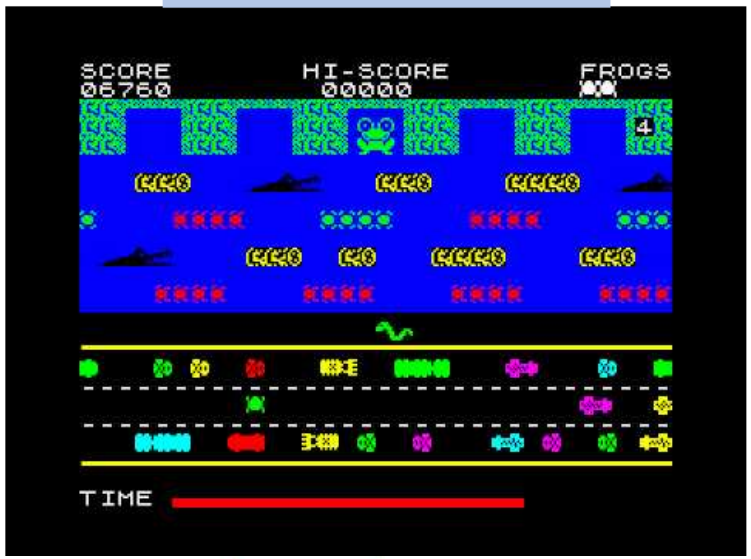
Dragon 32



Atari 400/800



ZX spectrum



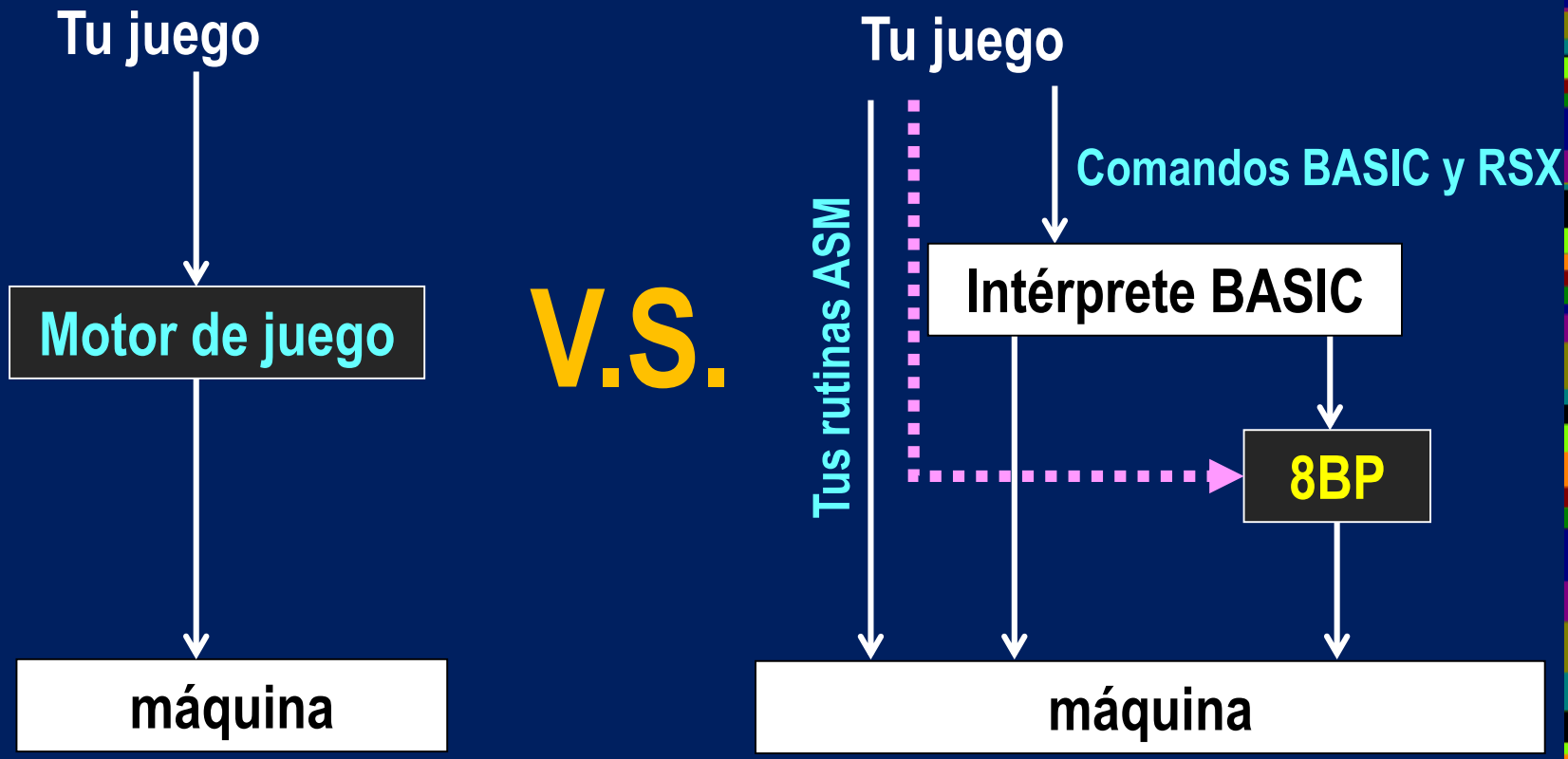
Amstrad CPC

MSX

Versión de frogger con 8BP

Introducción a 8BP y lógicas masivas

8BP es una librería de rutinas útiles para videojuegos y accesibles desde BASIC mediante comandos RSX



RSX (Resident System eXtensions)

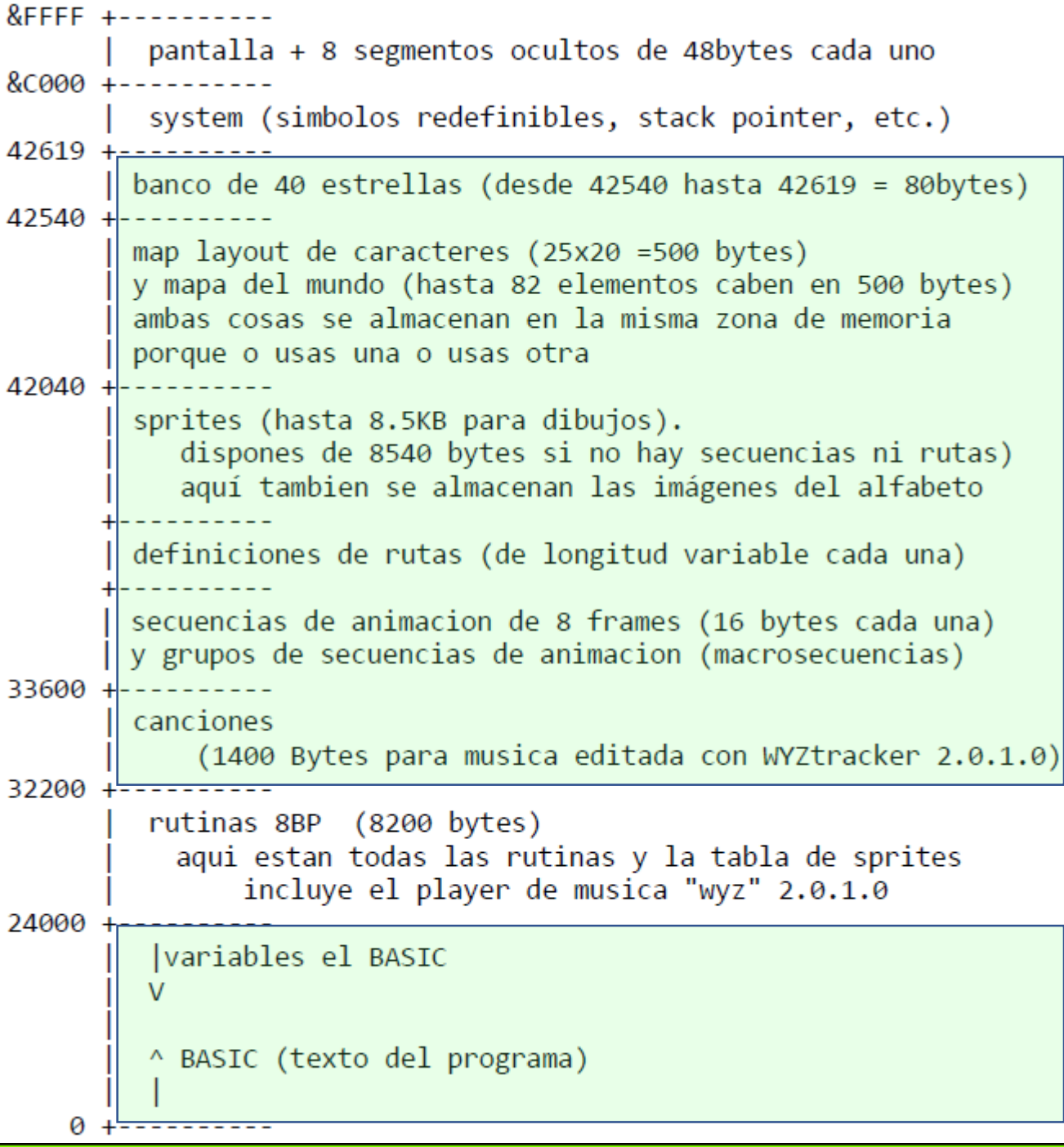
8BP Memory map

8BP Sólo ocupa 8 KB y te proporciona 27 comandos

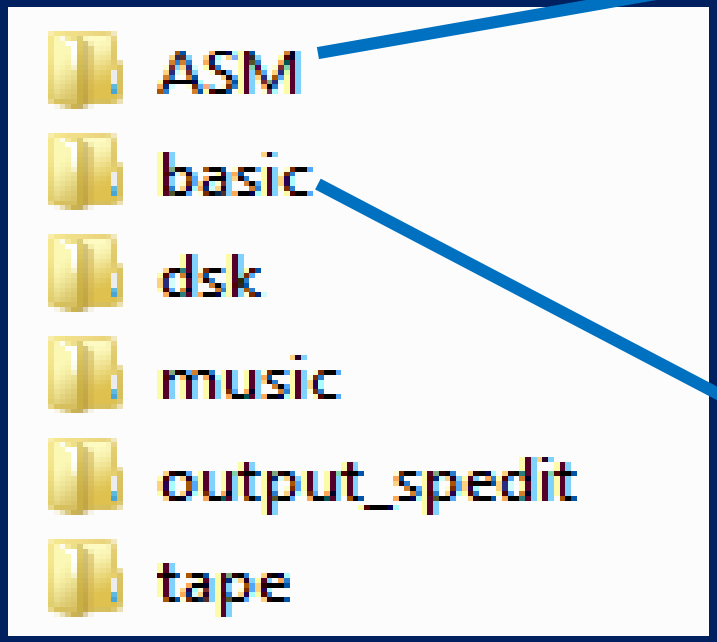
24 KB libre para BASIC

1.4 KB musica

8.5 KB sprites



Carpetas de un juego 8BP



Make_all.asm
Images_mygame.asm
Routes_my_game.asm
Secuences_my_game.asm
…

Loader.bas
Frogger.bas

capacidades

- 32 sprites con clipping (SETLIMITS), sobreescritura, ordenación y detección de colision (COLSPALL).
- 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 (comandos 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, tiera, Lluvia...
- Capacidad PSEUDO-3D
- Sólo ocupa 8 KB y reserva 8.5KB para sprites y 1.4KB para musica, dejando 24 KB para lógica BASIC.

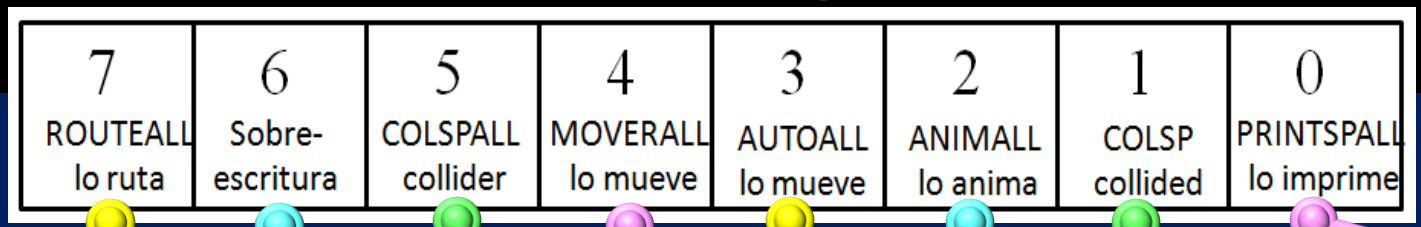
Sprites

- 8BP soporta 32 sprites de cualquier tamaño
- 16 bytes por sprite (9 parámetros)
- Comienzan en 27000
- Podemos leer con PEEK o con |PEEK
- Podemos escribir sus parámetros con POKE o con |POKE o con |SETUPSP
- El primer byte es el de estado
- Los puedes colocar con |LOCATESP

	1byte	2 bytes	2 bytes	1byte	1byte	1byte	1byte	2 bytes	1byte
sprite	status	coordy	coordx	vy	vx	seq	frame	imagen	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
7	27112	27113	27115	27117	27118	27119	27120	27121	27127
8	27128	27129	27131	27133	27134	27135	27136	27137	27143
9	27144	27145	27147	27149	27150	27151	27152	27153	27159
10	27160	27161	27163	27165	27166	27167	27168	27169	27175
11	27176	27177	27179	27181	27182	27183	27184	27185	27191
12	27192	27193	27195	27197	27198	27199	27200	27201	27207
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

Cada sprite tiene un byte de status:

SETUPSP,sp,0,status



visible

colisionable

Puede tener una secuencia de animación
ANIMALL o PRINTSPALL,1

Puede tener velocidad
AUTOALL o AUTOALL,1

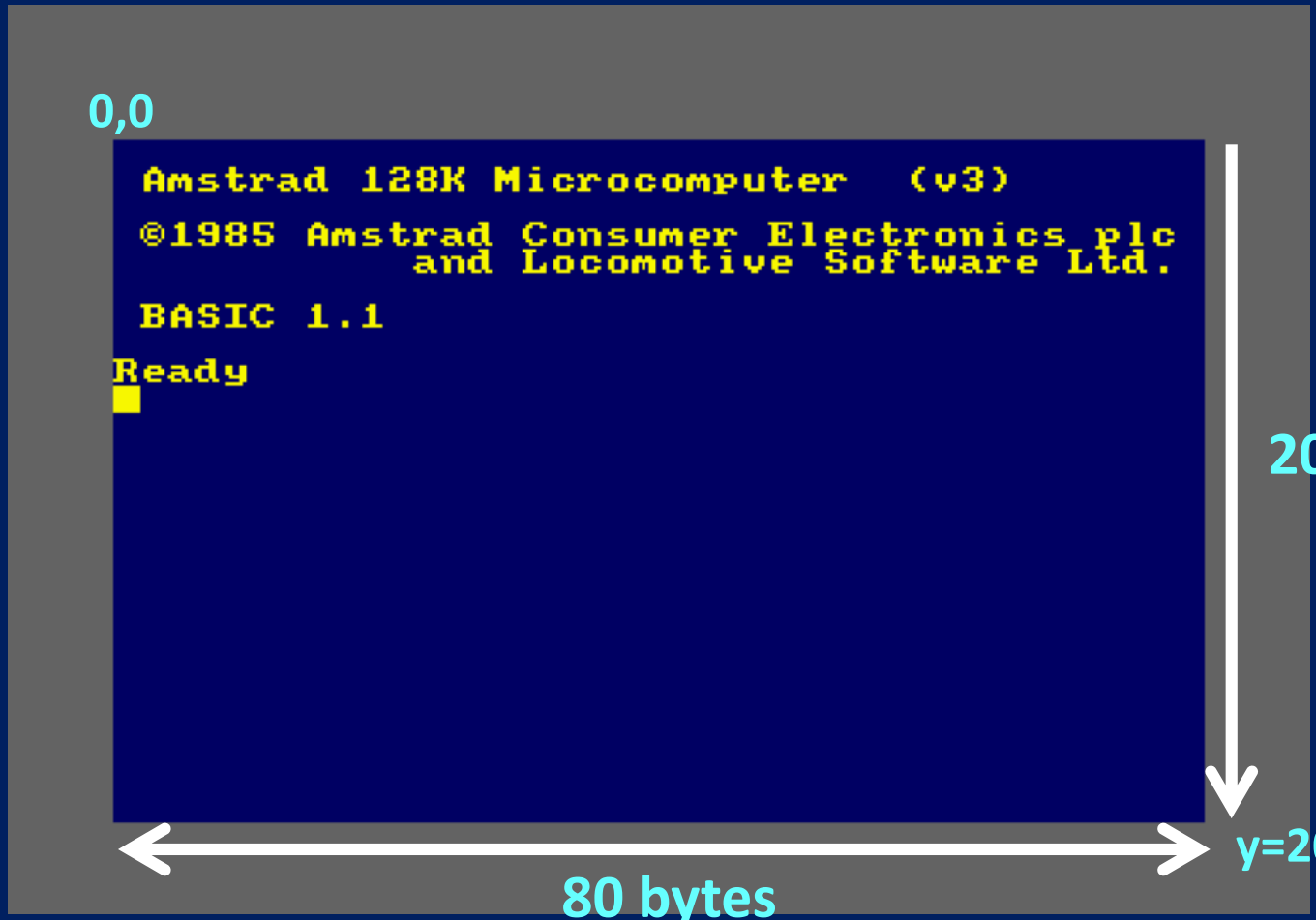
Puede moverse de forma relativa solidariamente con otros sprites
MOVERALL,dy,dx

Puede ser colisionador
COLSPALL,@collider,@collided

Puede tener sobreescritura

Puede tener una ruta asignada
SETUPSP, sp, 15, ruta

Coordenadas en 8BP



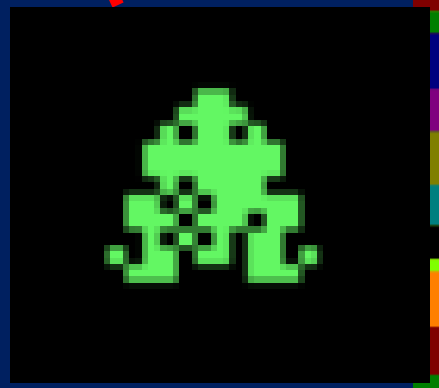
1 BYTE = 2 pixels de MODE 0

1 BYTE = 4 pixels de MODE 1

Primer ejemplo:

```
Ready
list
10 MEMORY 23999
20 CALL &6B78
30 DEFINT A-Z
40 |SETUPSP,31,0,1
50 |SETUPSP,31,9,16
55 x=40:y=100
60 |LOCATESP,31,y,x
70 |PRINTSPALL,0,0,0,0
Ready
run
Ready
```

```
10 MEMORY 23999
20 CALL &6B78
30 DEFINT A-Z
35 frogimg = 16
40 |SETUPSP,31,0,1
50 |SETUPSP,31,9,frogimg
55 x=40,y=100
60 |LOCATESP,31,y,x
70 |PRINTSPALL,0,0,0,0
```

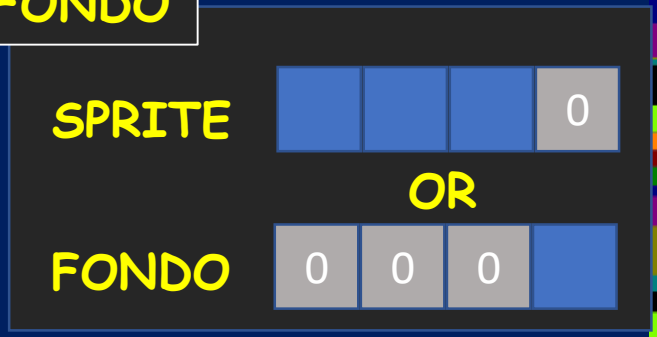


Sprites: sobreescritura

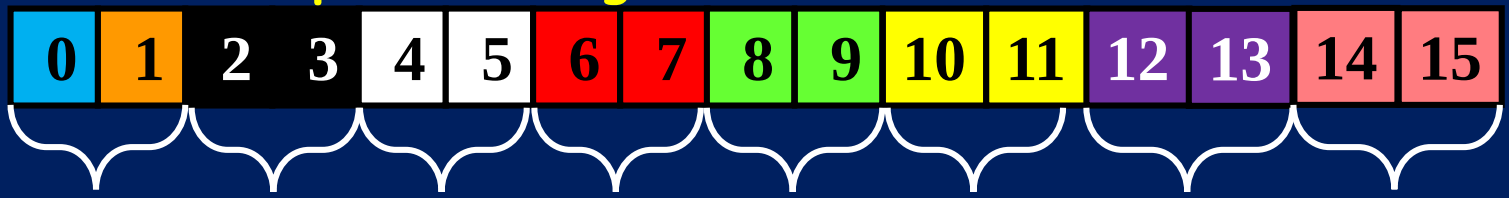
No usa doble-buffer asi que no gasta memoria y es muy rápido
Jamás destruye el fondo.
Técnica usada en juegos como "mision genocide" y "wonderboy"

FONDO= PIXEL AND 0001
New PIXEL= SPRITE OR FONDO

Se reduce el número de colores
9 en mode 0
3 en mode 1

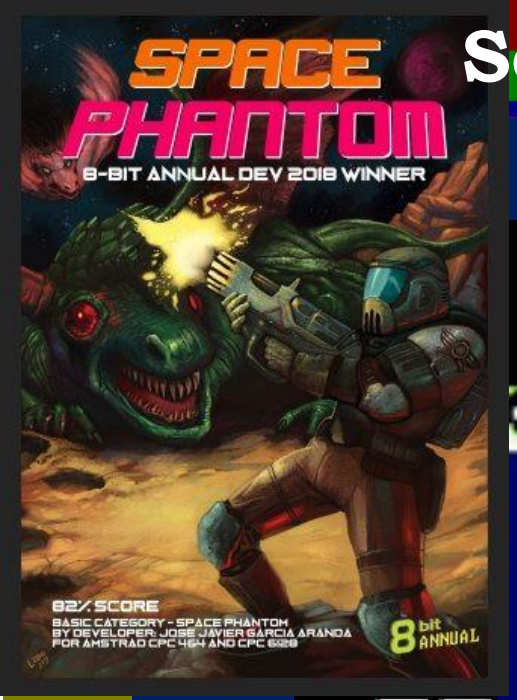


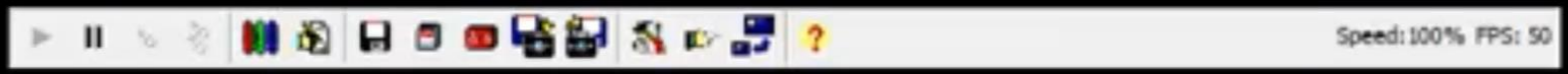
- Hay truco para usar mas de 9 colores en MODE 0 con sobreescritura
- 8BP también permite elegir 1 o 2 bit de fondo



Sprites: sobreescritura

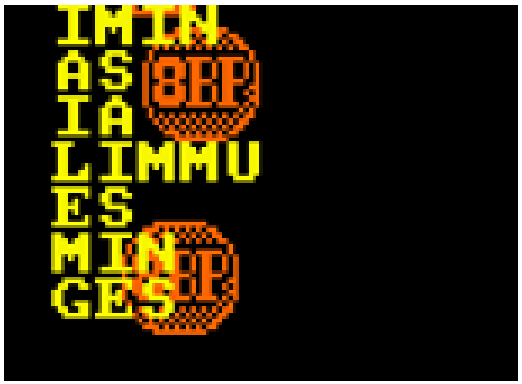
Sobreescritura y animación por rotación de tintas

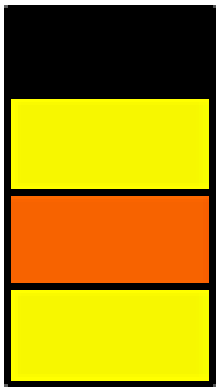




Sprites: sobreescritura

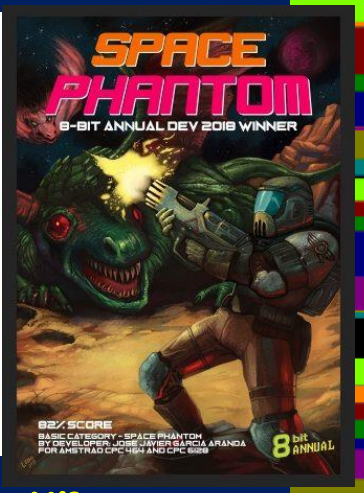
Con la misma técnica podemos pintar por detrás del fondo





00
01
Colores de fondo

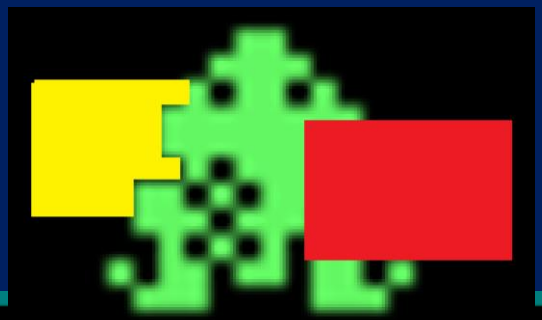
10
11
Color de sprite



En mode 0 incluso podemos hacer objetos por los que un personaje pasa por delante y otros por los que pasa por detrás

FONDO (2 bit)	SPRITE (2 bit)
0000	0100
0001	0101
0010	0110
0011	0111

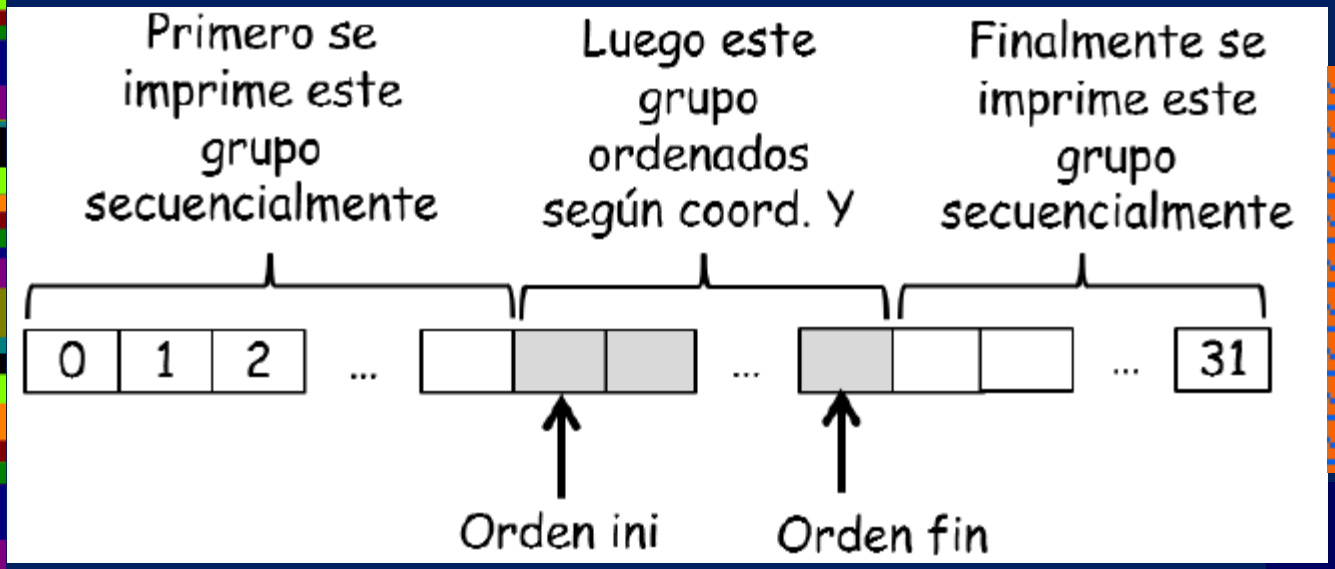
En este ejemplo de paleta con 4 colores de fondo y 3 para sprites, el color verde puede pasar por delante del color amarillo pero a la vez pasa por detrás del color rojo



Sprites: ordenamiento

La "burbuja restringida a un solo cambio" es lo mas rápido si casi están ordenados

|PRINTSPALL, ordenini, ordenfin, anima, sync



inicializacion del comando (solo una vez)

PRINTSPALL,0 : ordenacion parcial de sprites basado en Ymin

PRINTSPALL,1 : ordenacion total de sprites basado en Ymin

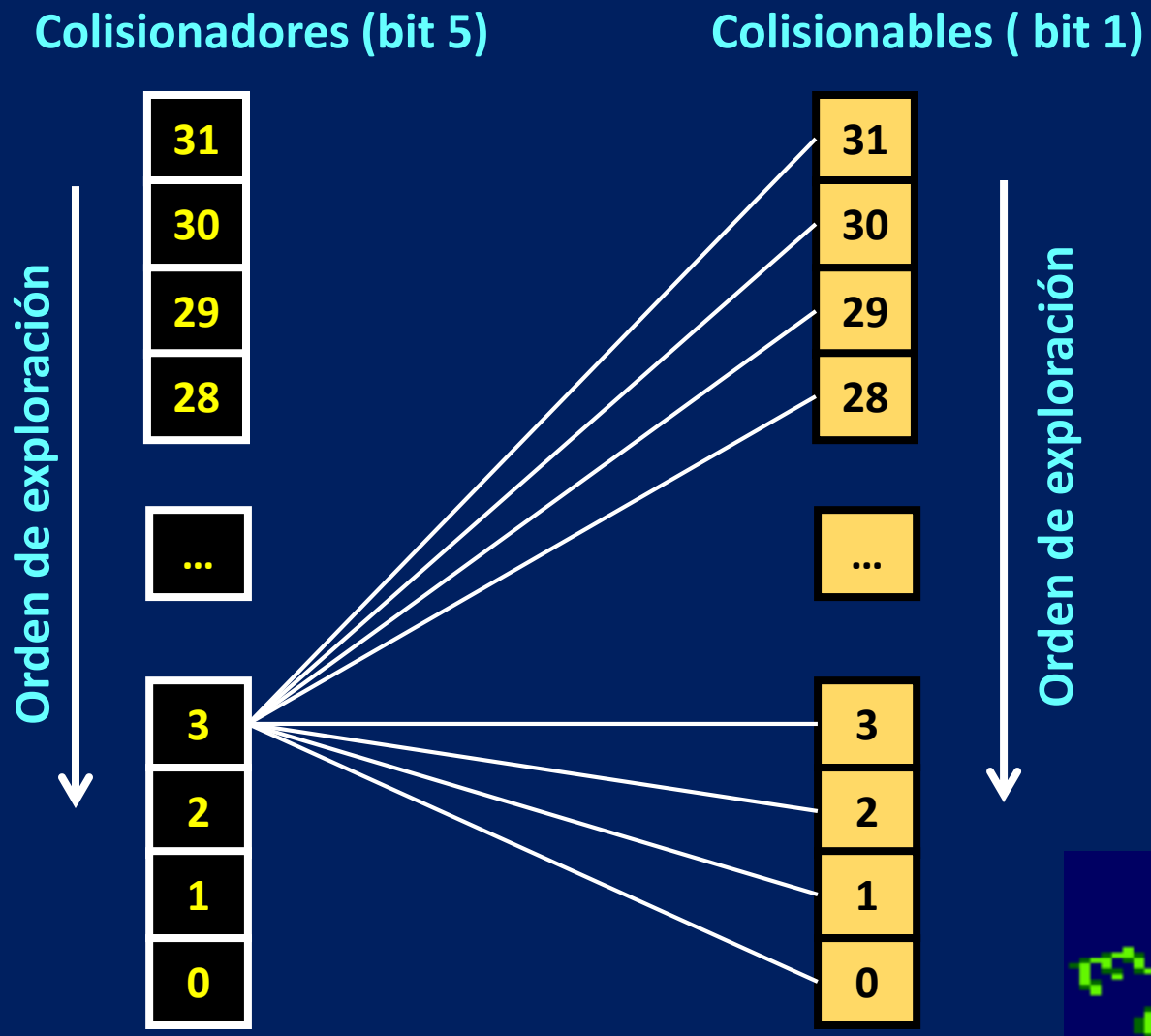
PRINTSPALL,2 : ordenacion parcial de sprites basado en Ymax

PRINTSPALL,3 : ordenacion total de sprites basado en Ymax


DEMO 8BP V36
0 : ORDEN PARCIAL EN Y MINIMA
1 : ORDEN COMPLETO EN Y MINIMA
2 : ORDEN PARCIAL EN Y MAXIMA
3 : ORDEN COMPLETO EN Y MAXIMA
order type (0-3)? 2
cesped no ordenado
resto ordenados
demo
Speed: 100% FPS: 50

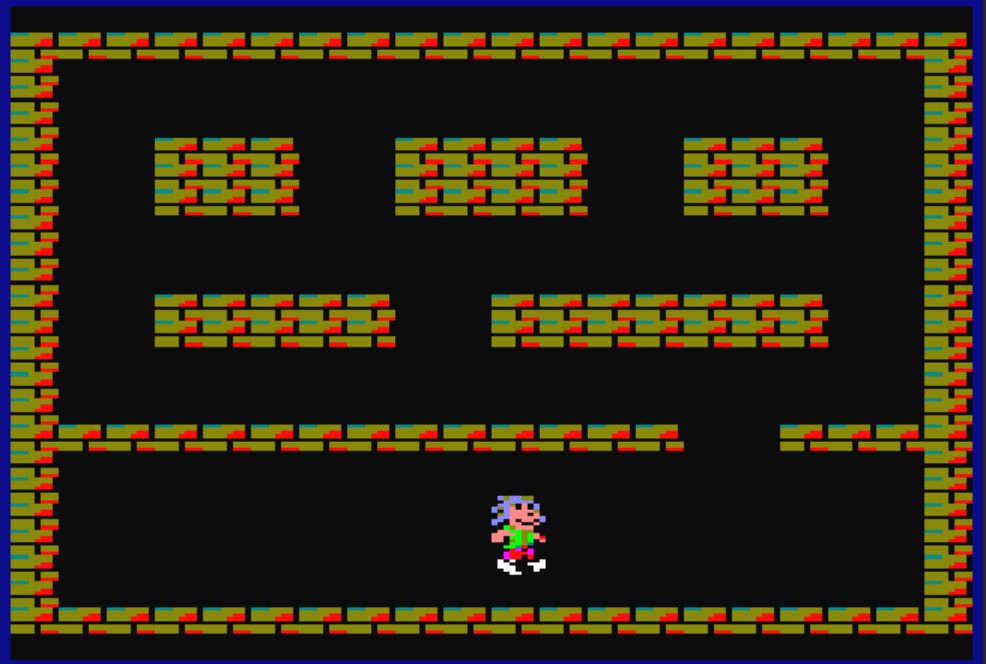
Sprites: colisiones

Funcionamiento de |COLSPALL



Mapa de tiles (layout)

```
100 c$(1)= "ZZZZZZZZZZZZZZZZZZZZ"
110 c$(2)= "Z                  Z"
120 c$(3)= "Z                  Z"
125 c$(4)= "Z                  Z"
130 c$(5)= "Z  ZZZ  ZZZZ  ZZZ  Z"
140 c$(6)= "Z  ZZZ  ZZZZ  ZZZ  Z"
150 c$(7)= "Z  ZZZ  ZZZZ  ZZZ  Z"
160 c$(8)= "Z                  Z"
170 c$(9)= "Z                  Z"
190 c$(10)="Z                  Z"
195 c$(11)="Z  ZZZZZ  ZZZZZZZ  Z"
200 c$(12)="Z  ZZZZZ  ZZZZZZZ  Z"
210 c$(13)="Z                  Z"
220 c$(14)="Z                  Z"
230 c$(15)="Z                  Z"
240 c$(16)="ZZZZZZZZZZZZZ  ZZZZ"
250 c$(17)="Z                  Z"
260 c$(18)="Z                  Z"
270 c$(19)="Z                  Z"
271 c$(20)="Z                  Z"
272 c$(21)="Z                  Z"
273 c$(22)="Z                  Z"
274 c$(23)="ZZZZZZZZZZZZZZZZZZZZ“
' print layout
560 FOR i=0 TO 23:|LAYOUT,i,0,@c$(i):NEXT
```



El comando LAYOUT interpreta cada letra como un sprite.

En este caso la “Z” es el sprite 31 y le hemos asignado una imagen de ladrillos

El espacio es el sprite NINGUNO

Colisión con mapa de tiles (layout) y colisión entre sprites

```
COLAY, sp, @col
```

0 = no hay colisión

1 = hay colisión

```
COLSPALL,@collider , @collided
```

32 = no hay colisión
<32 hay colision

32 = no hay colisión
<32 hay colision

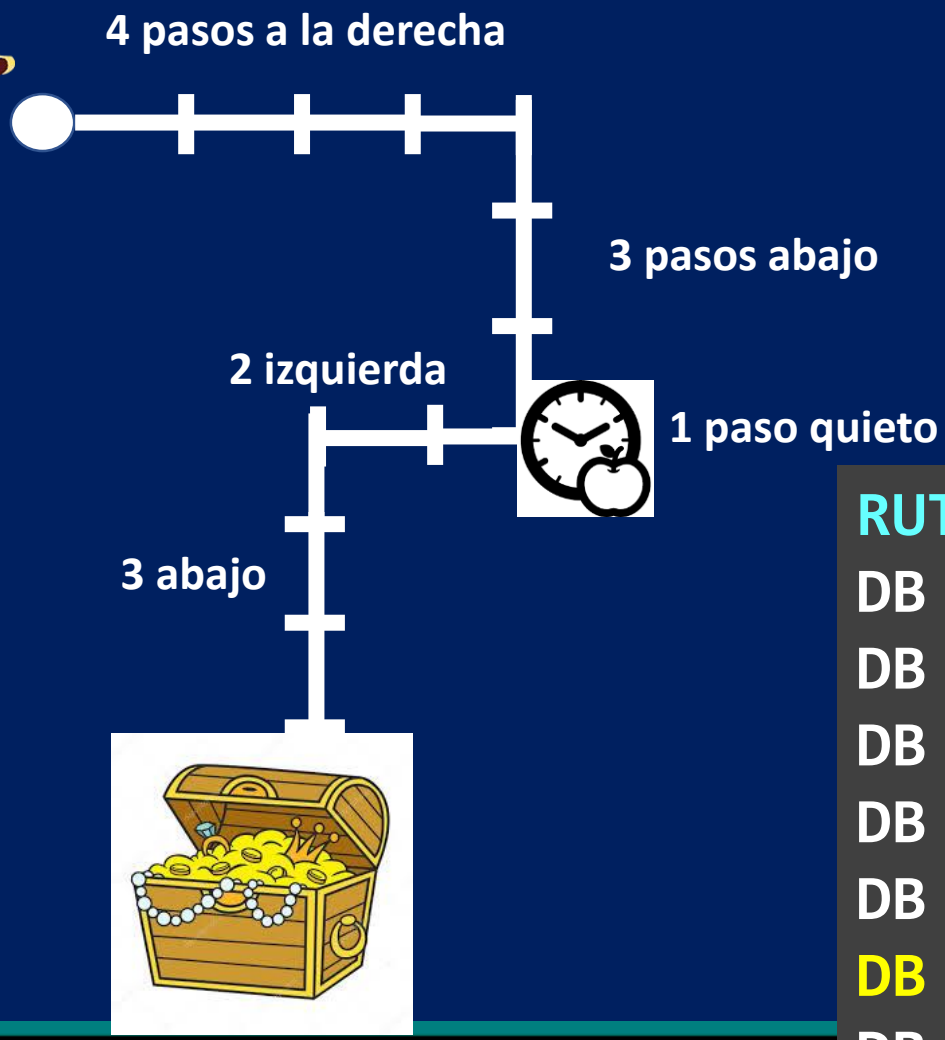
Sprites: secuencias de animacion

```
|SETUPSP, <sprite_id>, 7,  <sequence number>
```

sequences_mygame.asm

```
_SEQUENCES_LIST
dw MONTOYA_R0,MONTOYA_R1,MONTOYA_R2,MONTOYA_R1,0,0,0,0    ;1
```

Capa de Sprites: concepto de ruta

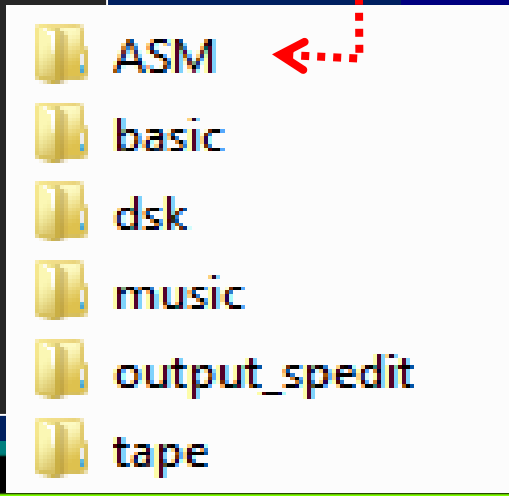



```
RUTA DEL TESORO
DB 4, 0, 1
DB 3, 1, 0
DB 1, 0, 0
DB 2, 0, -1
DB 3, 1, 0
DB 1,-6,-2
DB 0
```

Routes_mygame.asm

```
; LISTA DE RUTAS
;================
;pon aqui los nombres de todas las rutas que hagas
ROUTE_LIST
    dw ROUTE0
    dw ROUTE1
    dw ROUTE2
    dw ROUTE3
…

ROUTE0 ; RUTA DEL TESORO
DB 4, 0, 1
DB 3, 1, 0
DB 1, 0, 0
DB 2, 0, -1
DB 3, 1, 0
DB 1,-6,-2
DB 0
```

```
|SETUPSP,pirata,15, 0
```

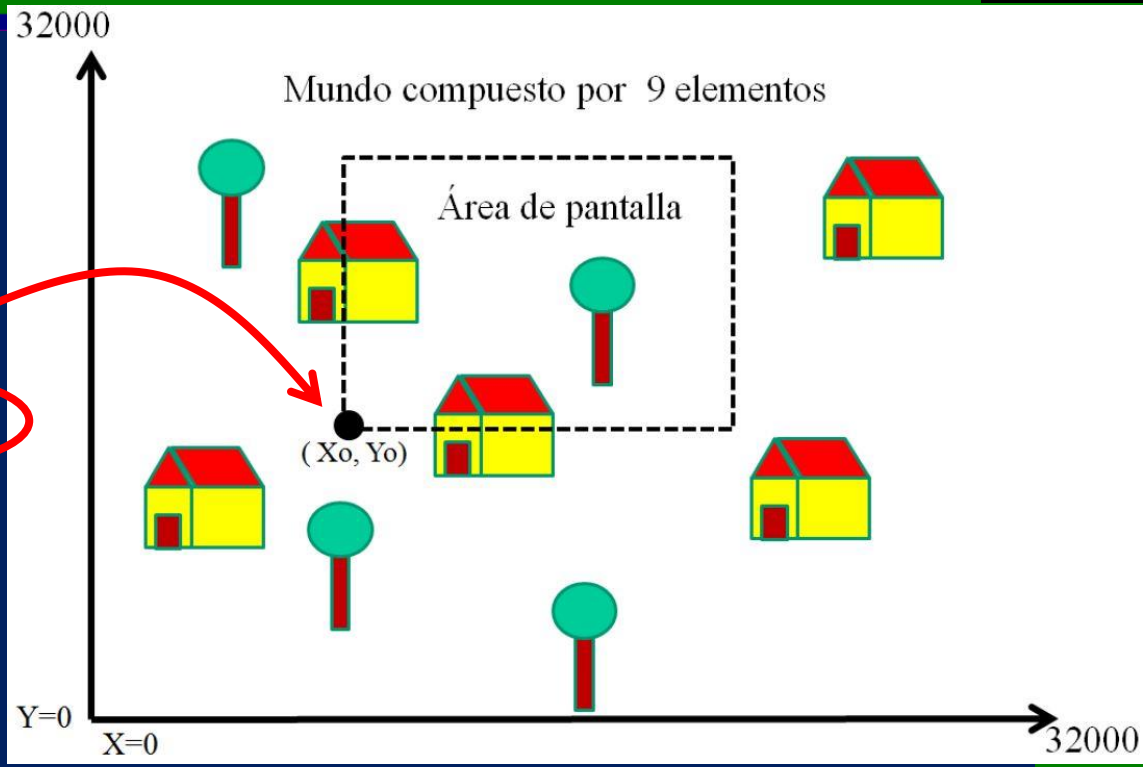


Scroll multidireccional

Scroll basado en una "ventana deslizante"

Comando

```
|MAP2SP, Yo, Xo
```



El mundo mide 32000 x 32000

Scroll multidireccional

31
30
29
28

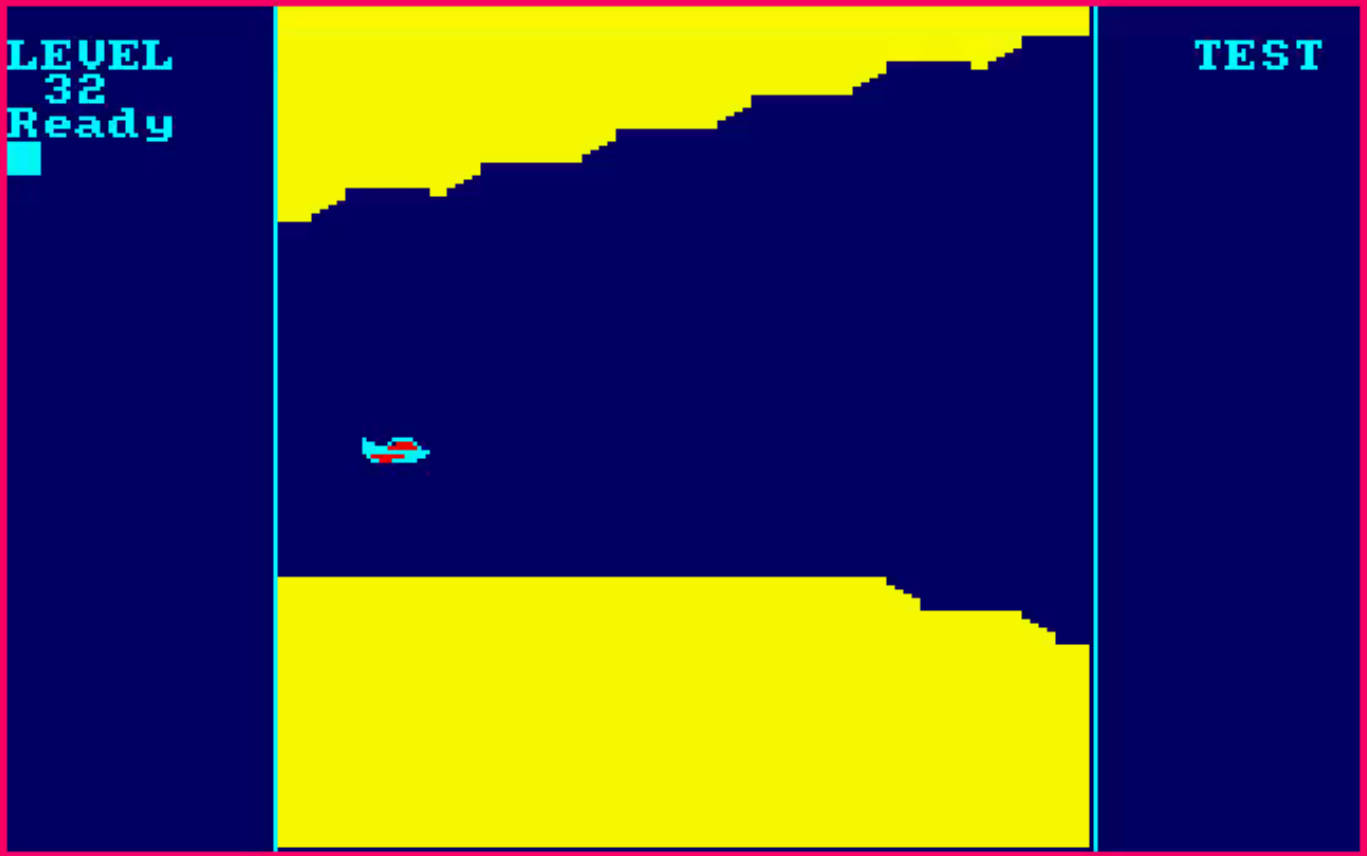
muñecos

...

3
2
1
0

Generados por **MAP2SP** Montañas , casas...

Scroll multidireccional: 32 fps!

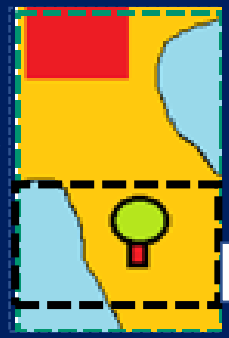


Speed:100% FPS: 50

Scroll multidireccional

Mapa completo en dirección 23000 (por ejemplo)

Mapa parcial ubicado en la dirección 42040

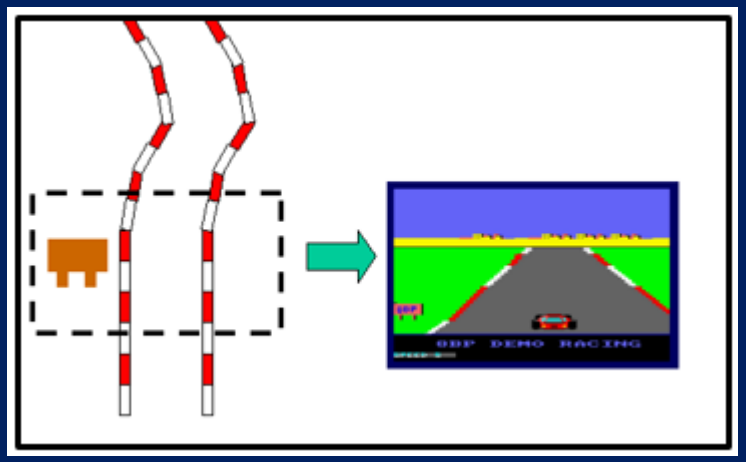


UMAP,23000,24000,200,500,0,80

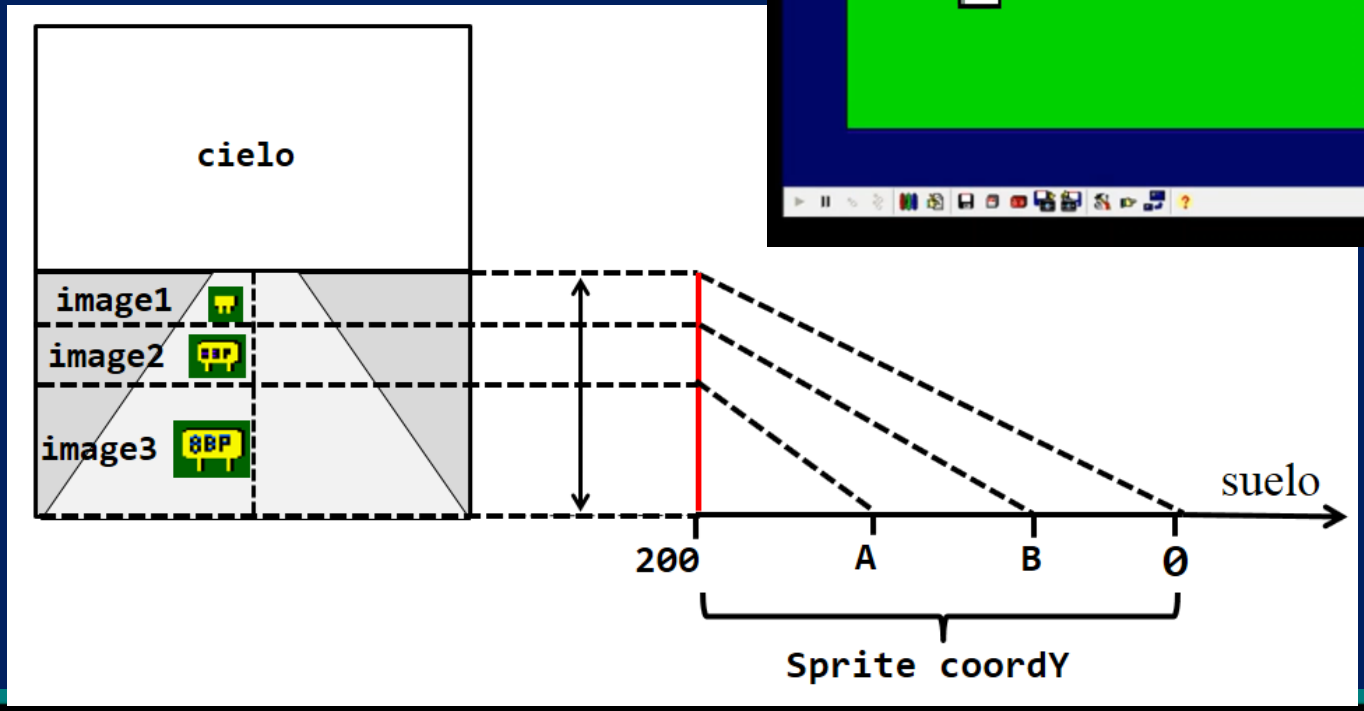
MAP2SP,y,x

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

Pseudo-3D



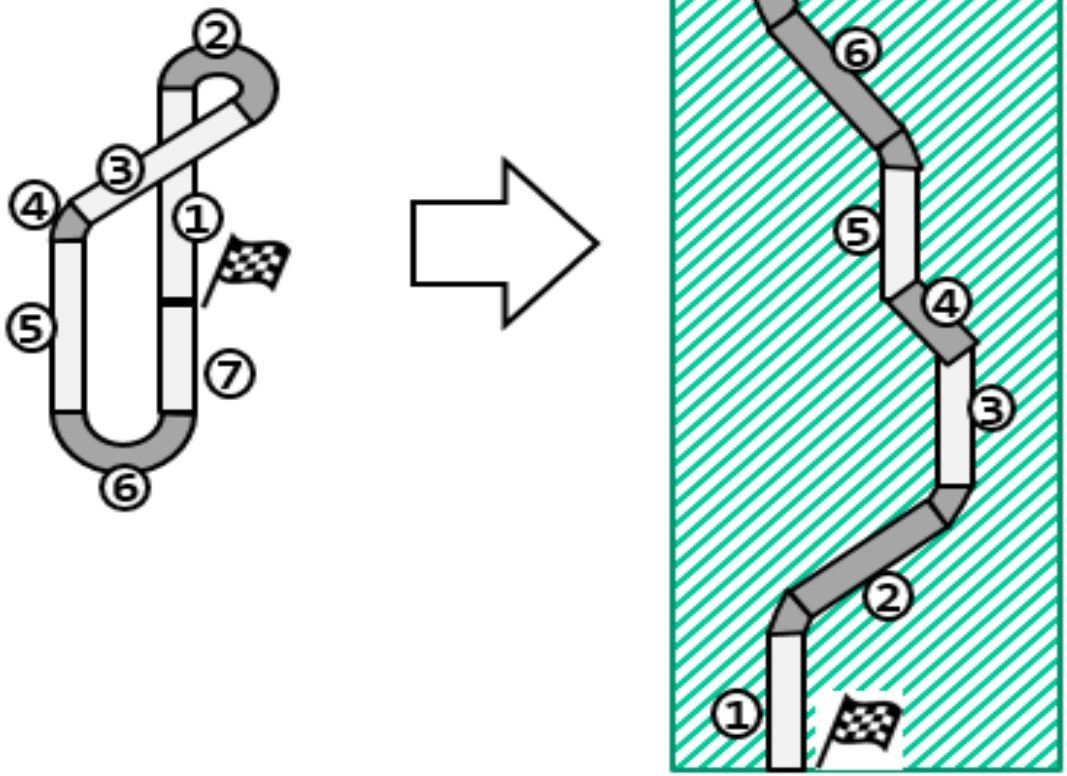
El comando 3D nos permite recorrer un mundo 2D proyectado

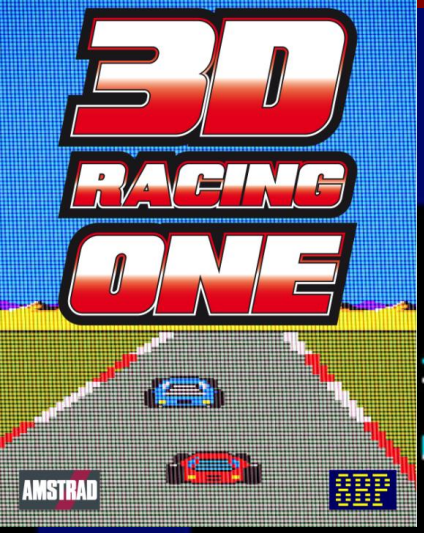


CONCEPTO:

Sólo es 3D el plano del suelo.

Las curvas son una ilusión.

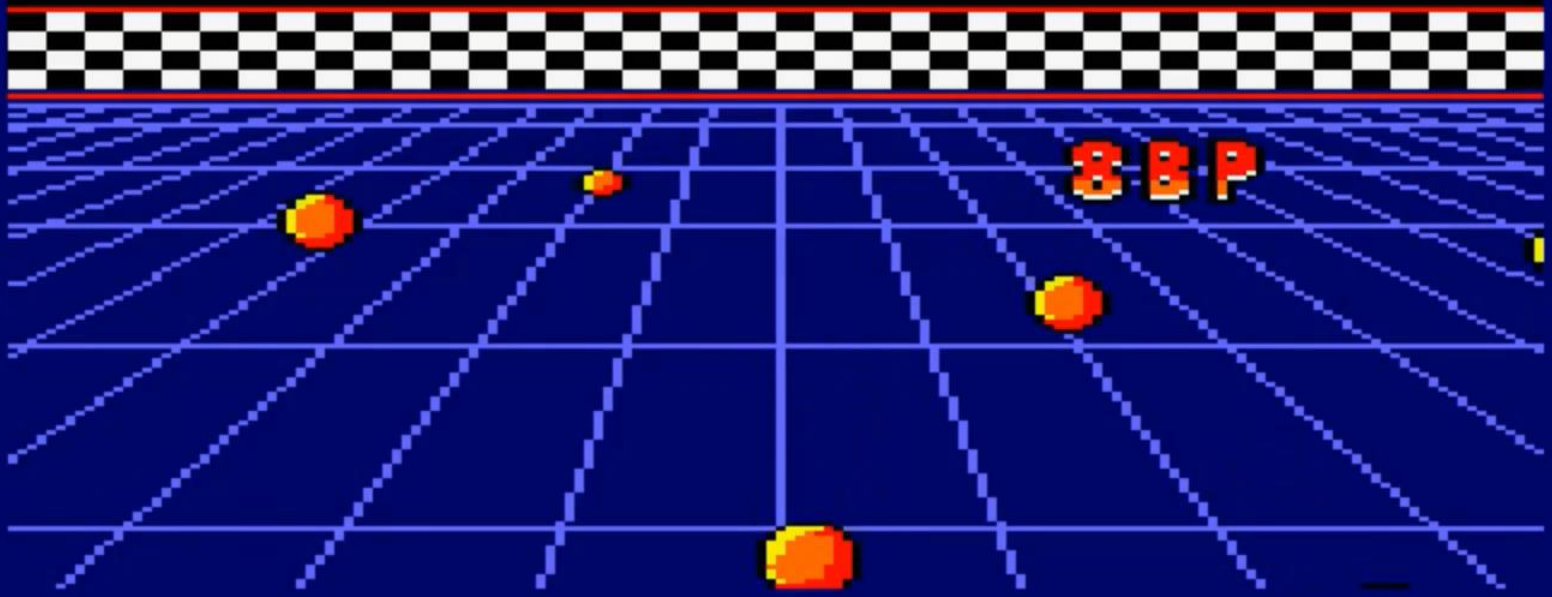




3D RACING ONE

JOSE JAVIER GARCIA ARANDA 2018

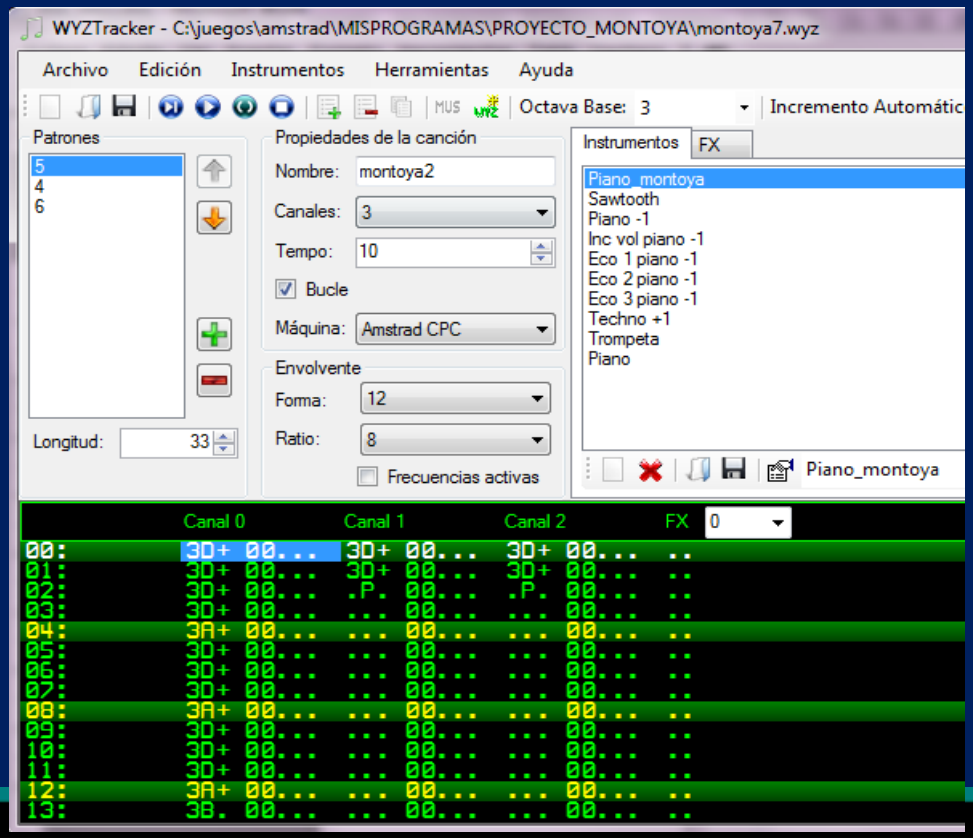
LY BASIC PROGRAM CREATED WITH BBP



Speed:100% FPS: 50

Musica on-game

```
10 |MUSIC,3,6 : comienza a sonar la melodia 3 a velocidad 6

. . .

2000 |MUSICOFF: ' deja de sonar la musica
```



Compones las canciones con el WYZ tracker

El WYZ player esta integrado en la librería

Lógicas masivas

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 o |AUTOALL,1

Ejecución de menos comprobaciones

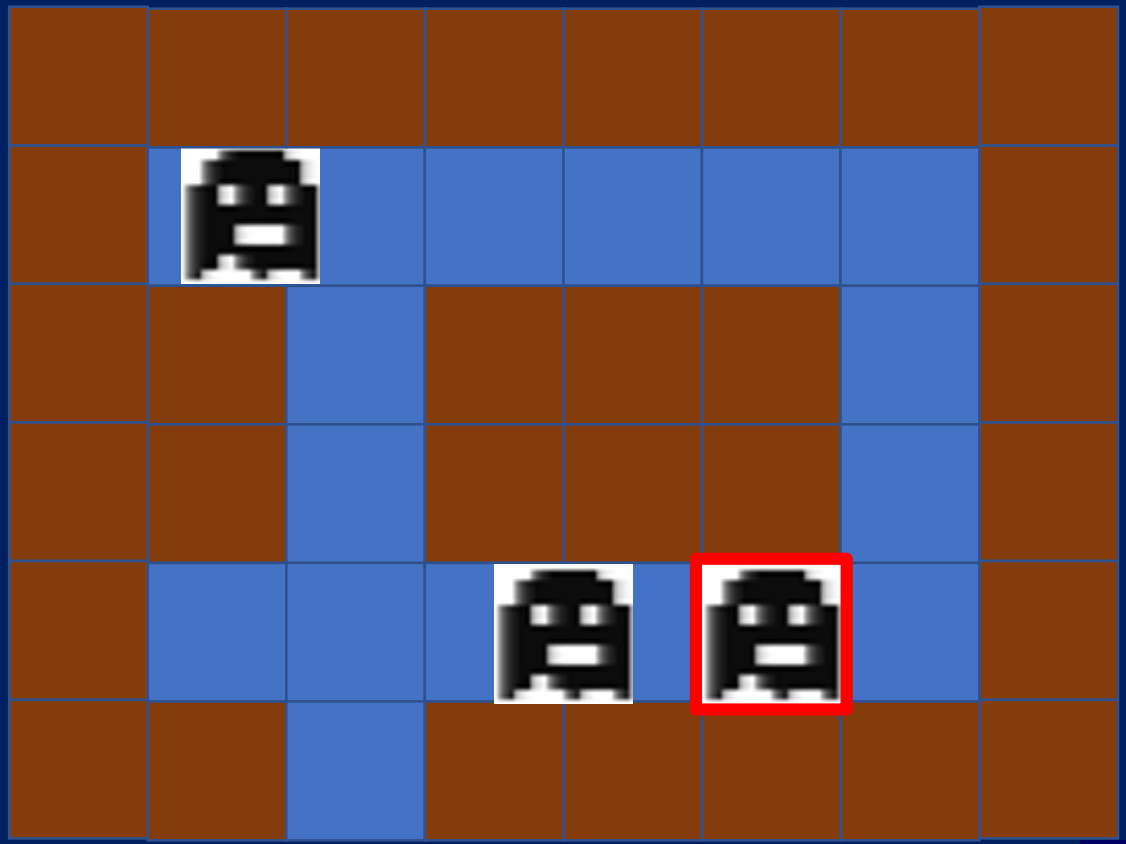
- En cada ciclo de juego ejecutar un subconjunto de comprobaciones
- Todas las comprobaciones tardarán varios frames pero no importa

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**"

Lógicas masivas

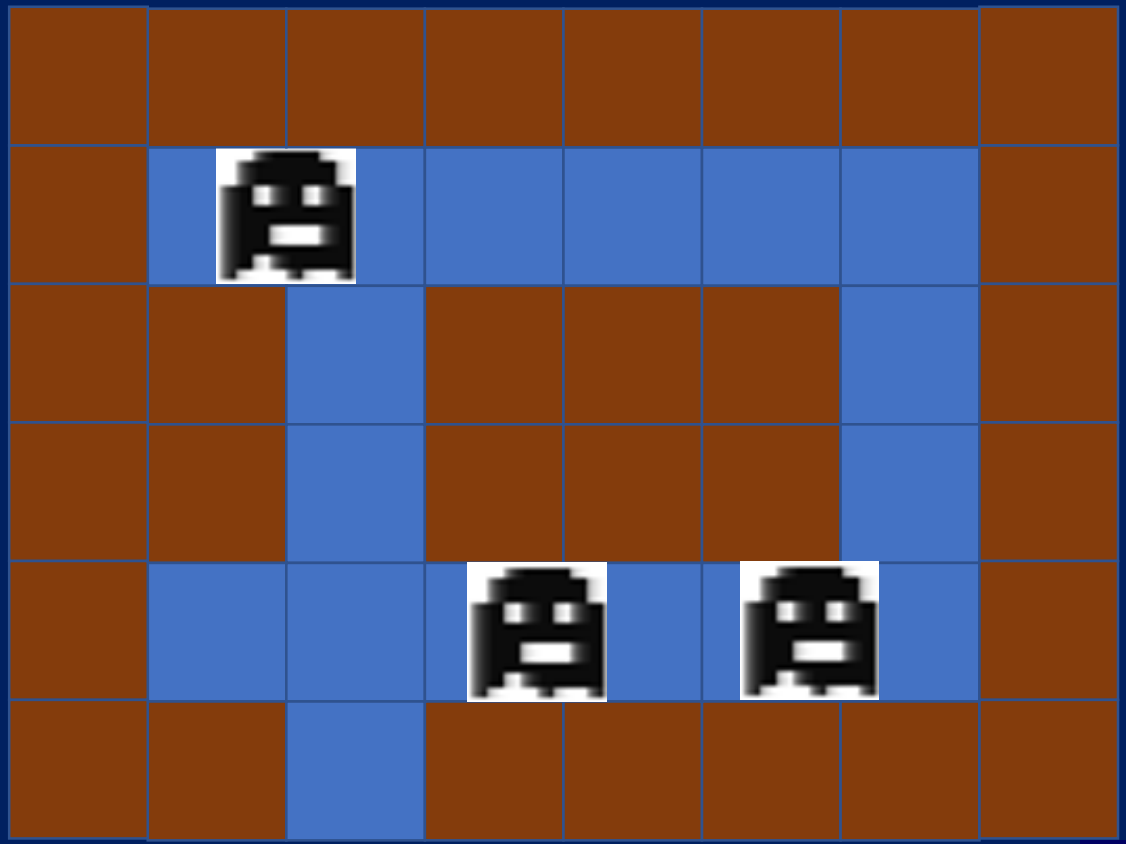
Ciclo= 0



massive logics

Lógicas masivas

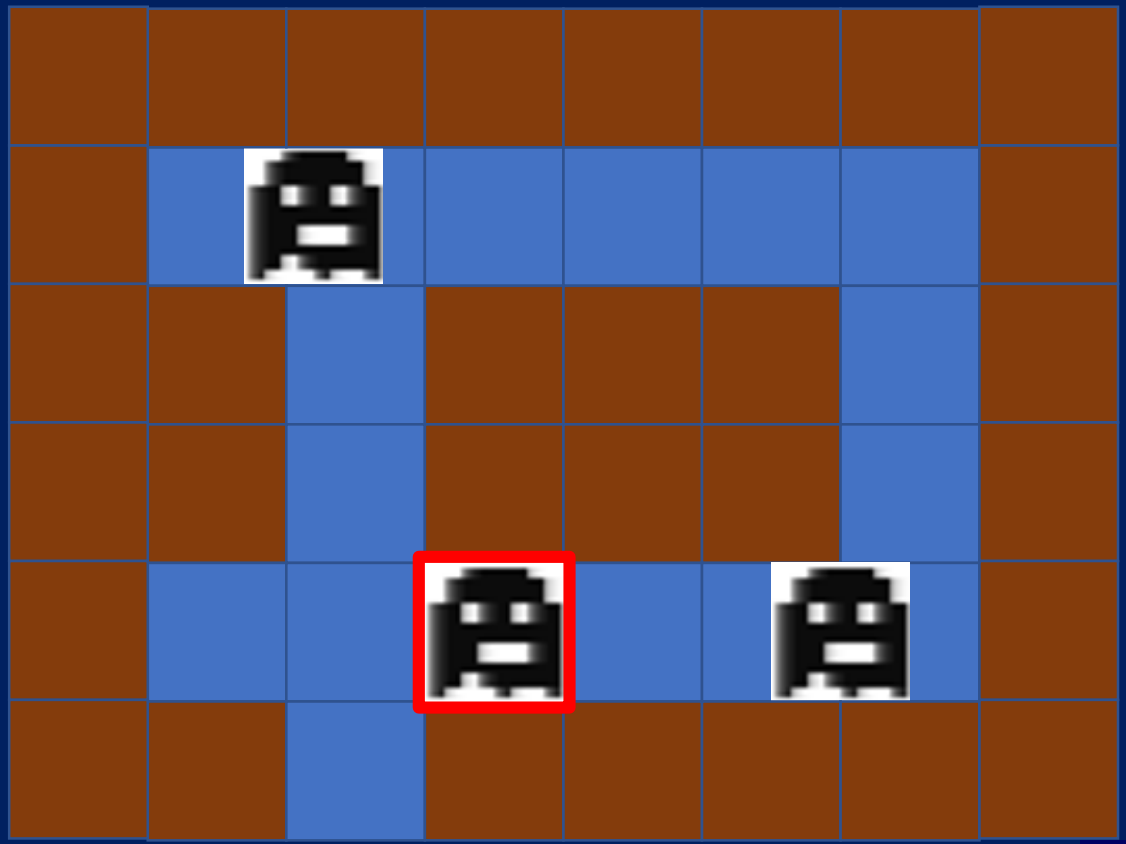
Ciclo= 1



massive logics

Lógicas masivas

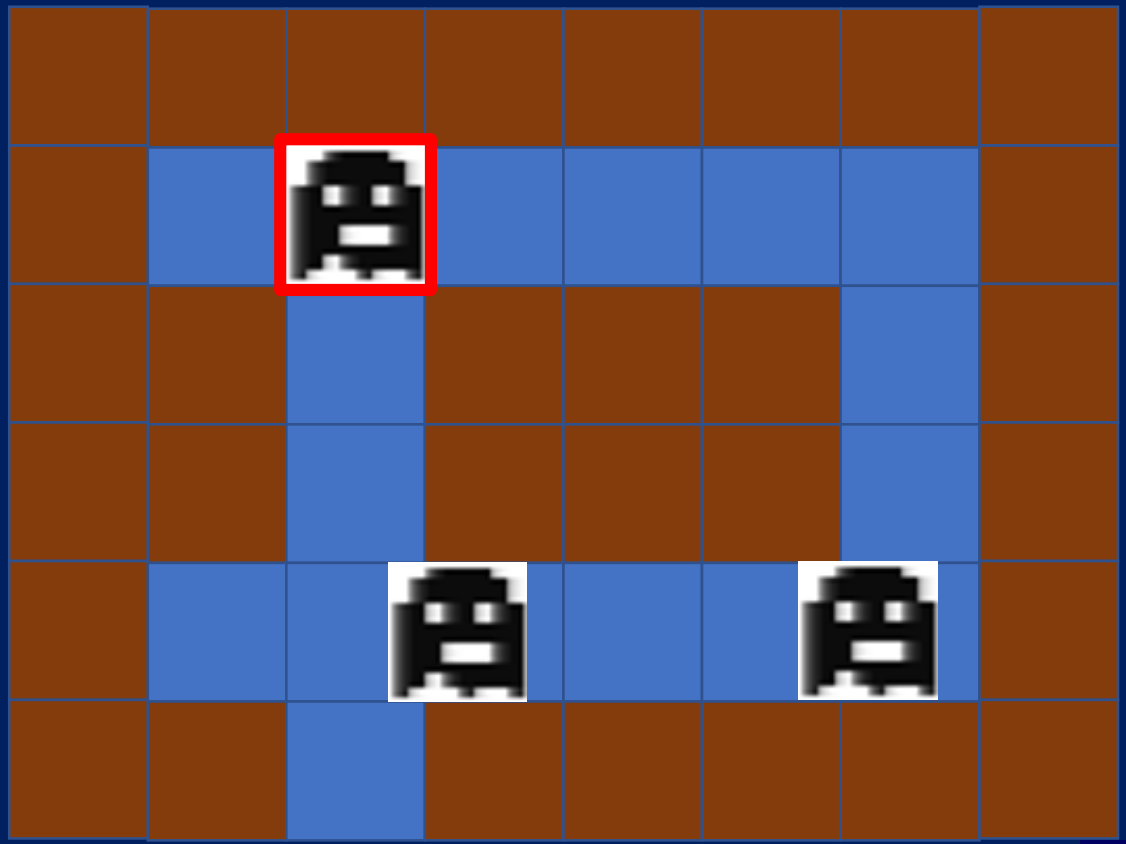
Ciclo= 2



massive logics

Lógicas masivas

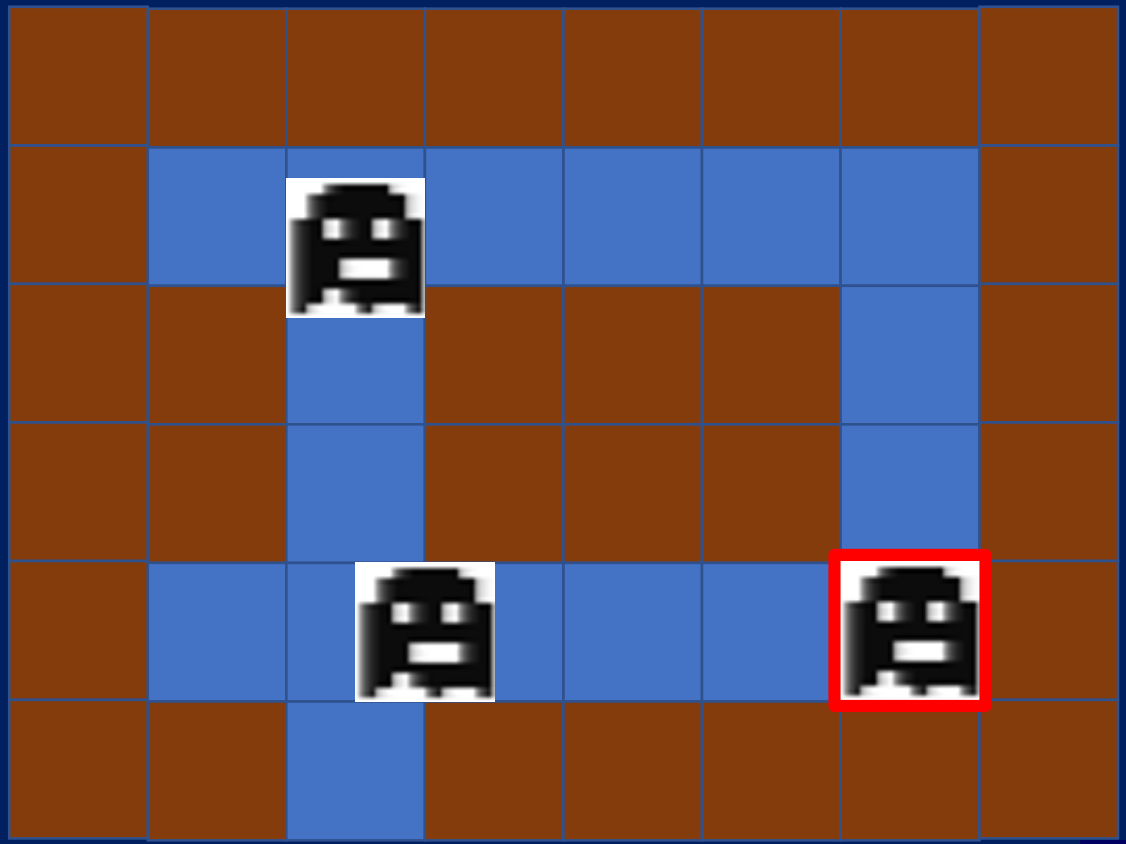
Ciclo= 3



massive logics

Lógicas masivas

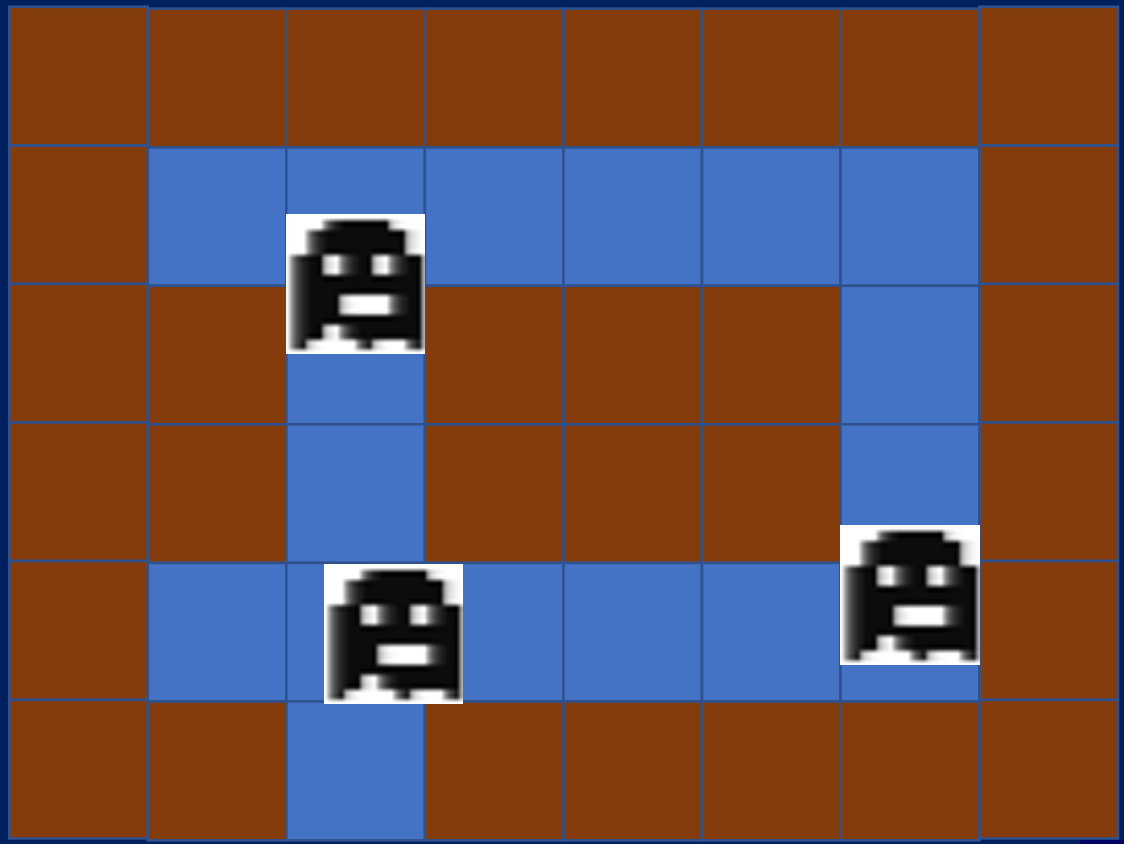
Ciclo= 4



massive logics

Lógicas masivas

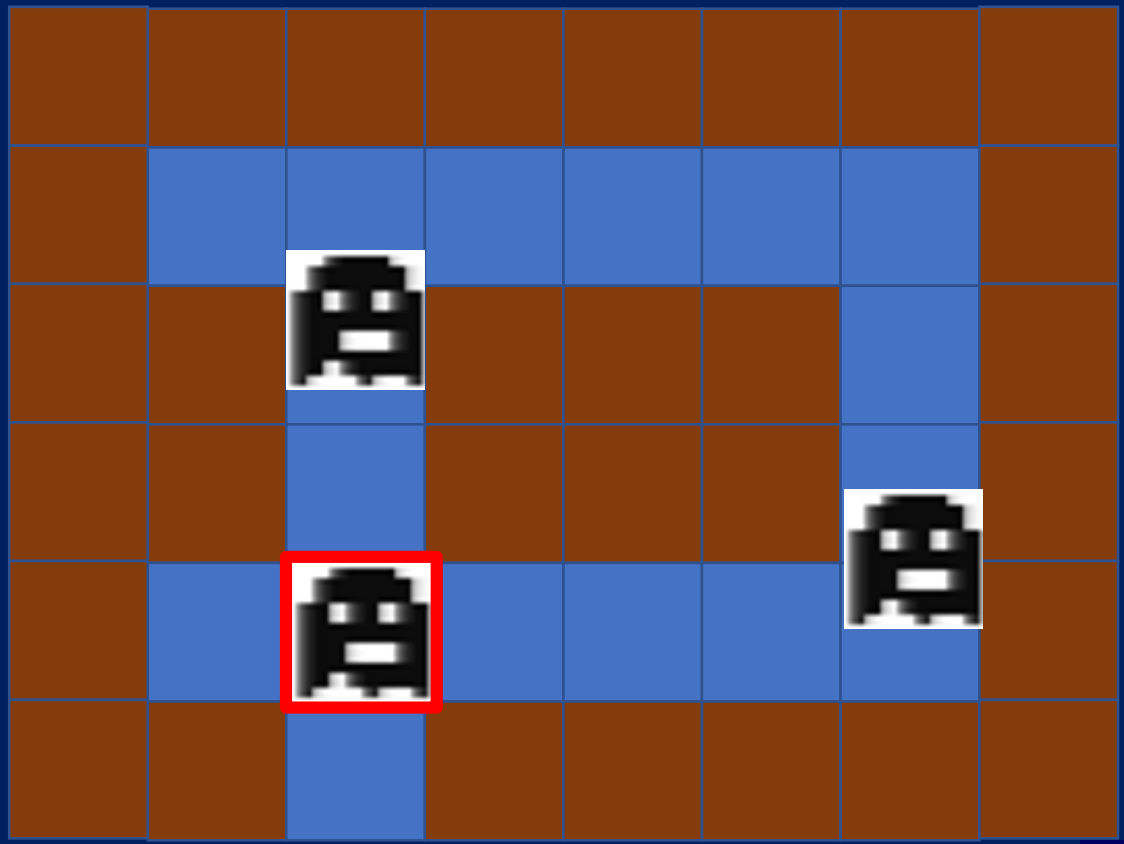
Ciclo= 5



massive logics

Lógicas masivas

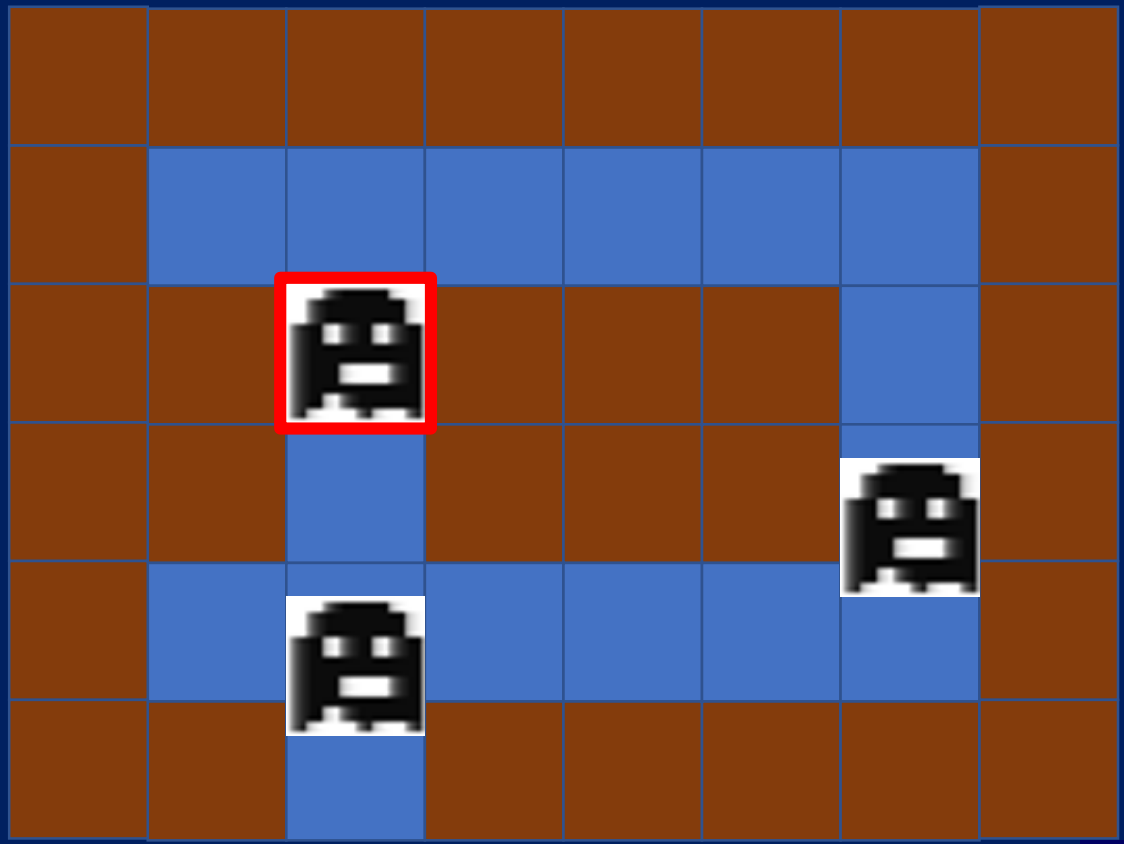
Ciclo= 6



massive logics

Lógicas masivas

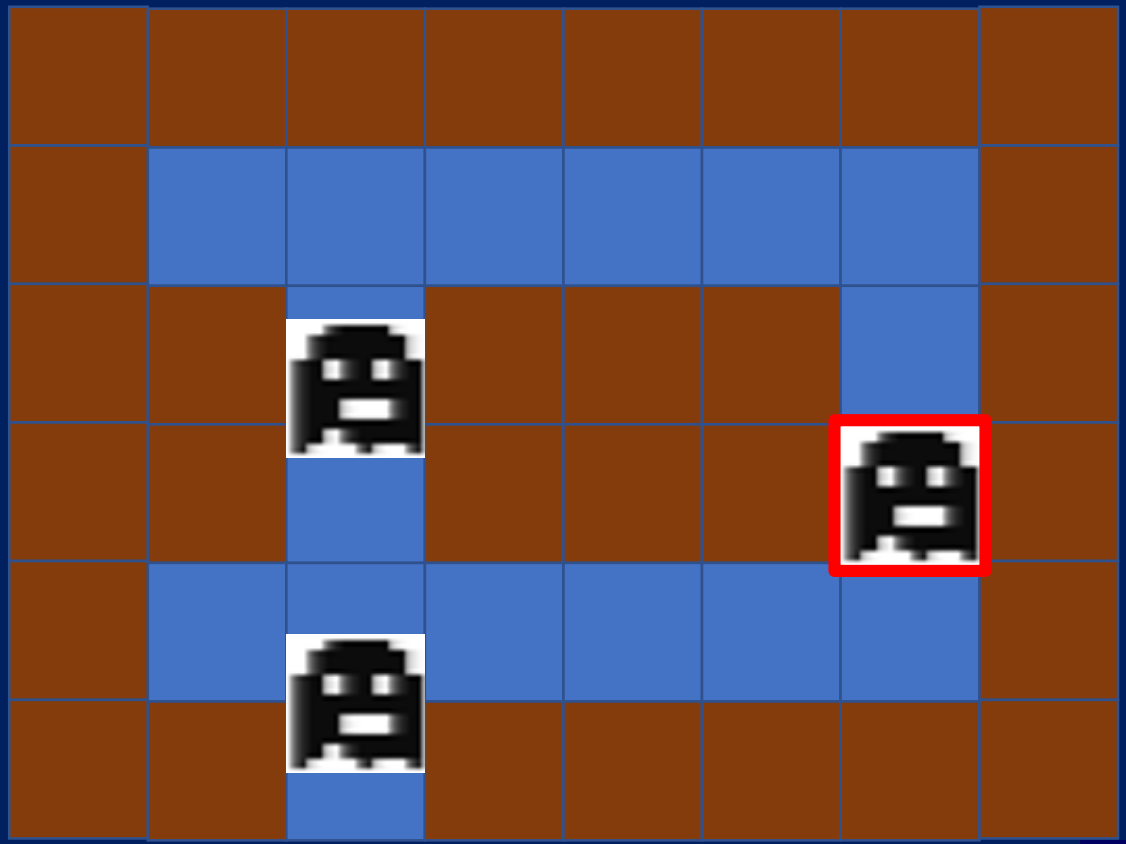
Ciclo= 7



massive logics

Lógicas masivas

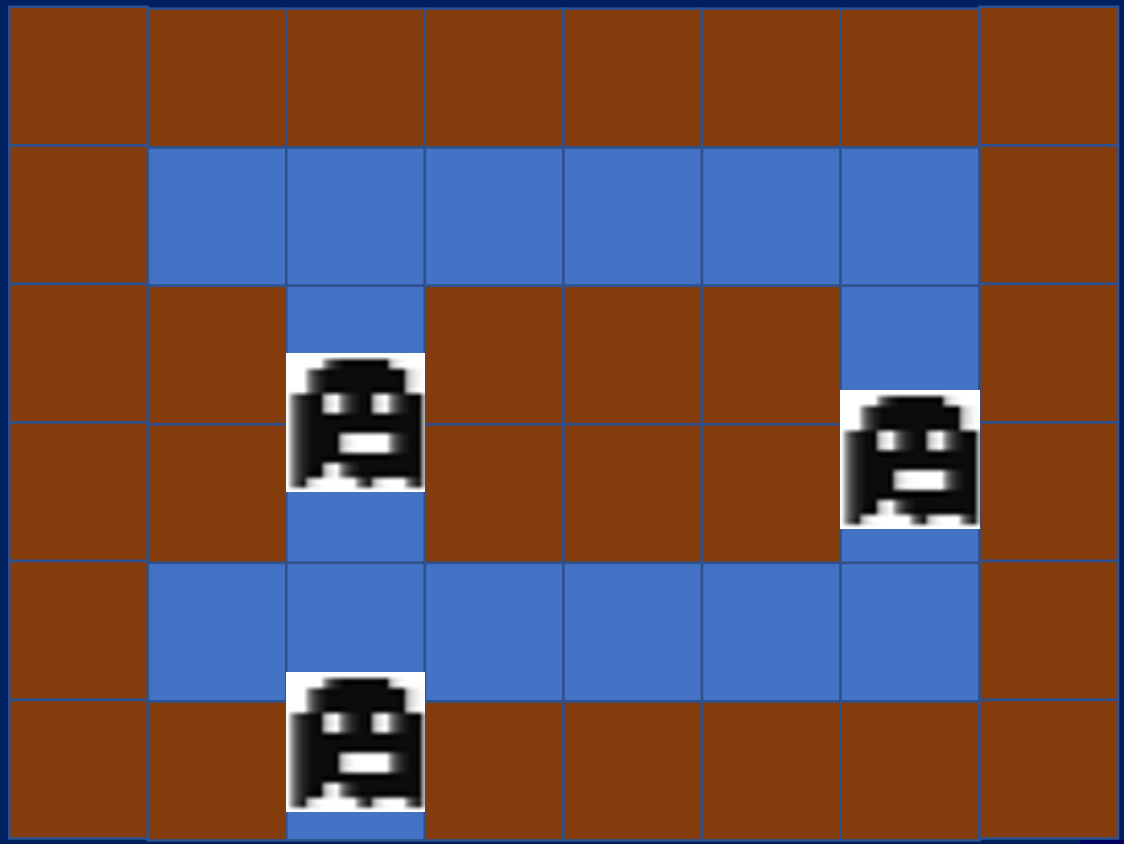
Ciclo= 8



massive logics

Lógicas masivas

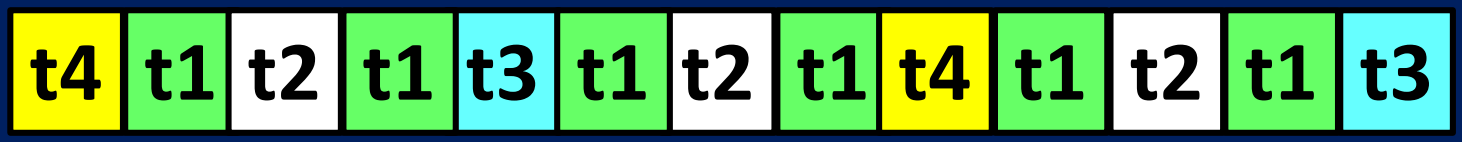
Ciclo= 9



massive logics

Lógicas masivas

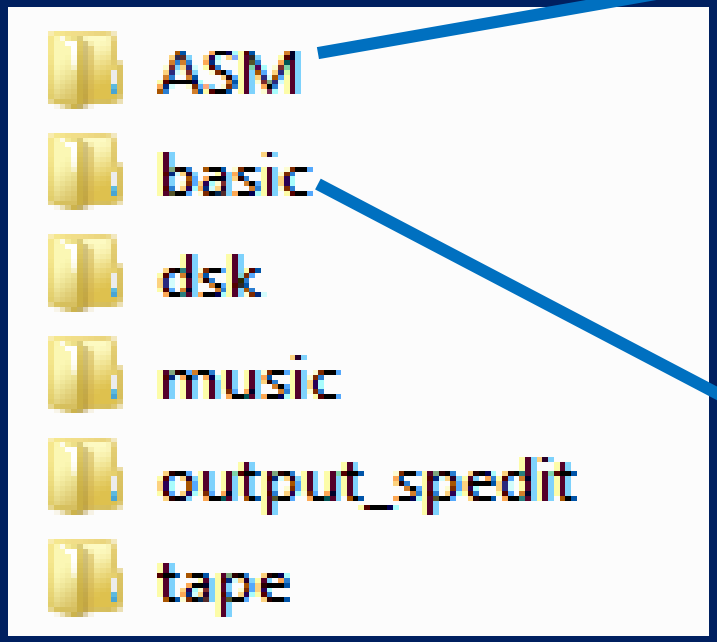
4 tareas, pero tan sólo una se ejecuta cada la vez




```
10 IF ciclo AND 1 THEN 90
20 REM cada dos ciclos entramos aquí
25 IF ciclo AND 3 THEN 80
30 REM cada 4 ciclos entramos aquí
35 IF ciclo AND 7 THEN 70
40 REM cada 8 ciclos entramos aquí
50 <tarea 4> : GOTO 100
70 <tarea 3> : GOTO 100
80 <tarea 2> : GOTO 100
90 <tarea 1>
100 REM --- fin de tareas ---
```


AMSTRAD
Frogger
eterno
EDICIÓN ESPECIAL
AMSTRAD
ETERNO

Carpetas de un juego 8BP



Make_all.asm
Images_mygame.asm
Routes_my_game.asm
Secuences_my_game.asm
…

Loader.bas
Frogger.bas

Diseño de sprites : SPEDIT11

.txt

```
;------ BEGIN IMAGE --------
db 4 ; ancho
db 16 ; alto
db 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0
db 0 , 1 , 8 , 0
db 0 , 3 , 12 , 0
db 0 , 5 , 10 , 0
db 0 , 15 , 15 , 0
db 0 , 15 , 15 , 0
db 0 , 5 , 14 , 0
db 1 , 10 , 15 , 8
db 1 , 13 , 13 , 8
db 0 , 10 , 11 , 0
db 2 , 13 , 11 , 4
db 1 , 12 , 3 , 8
db 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0
;------ END IMAGE --------
```

.asm

```
;------ BEGIN IMAGE ----
FROG_UP1
db 4 ; ancho
db 16 ; alto
db 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0
db 0 , 1 , 8 , 0
db 0 , 3 , 12 , 0
db 0 , 5 , 10 , 0
db 0 , 15 , 15 , 0
db 0 , 15 , 15 , 0
db 0 , 5 , 14 , 0
db 1 , 10 , 15 , 8
db 1 , 13 , 13 , 8
db 0 , 10 , 11 , 0
db 2 , 13 , 11 , 4
db 1 , 12 , 3 , 8
db 0 , 0 , 0 , 0
db 0 , 0 , 0 , 0
;------ END IMAGE ------
```

Diseño de sprites: fichero “images_mygame.asm”

Images_mygame.asm

```
IMAGE_LIST
;-------------------------------------------------------
; pondremos aqui una lista de las imagenes que queremos usar
; se empiezan a numerar en 16
; podemos usar hasta 255 imagenes especificadas de este modo
;-------------------------------------------------------
DW FROG_UP1;16
DW FROG_UP2;17
DW STONE;18
DW STONE2;19
DW WATER;20
DW SAND;21
DW CAR1_L;22
DW MONEDA;23
DW FROG_L1_DEL;24

...
DW FROGGER;38
```

```
280 tronco=30: mosca=36:watermosca=37:frogger=38
...
600 |SETUPSP,0,0,1:|SETUPSP,0,9,frogger:|PRINTSP,0,0,0
```

Diseño de sprites: flipeo horizontal

Images_mygame.asm

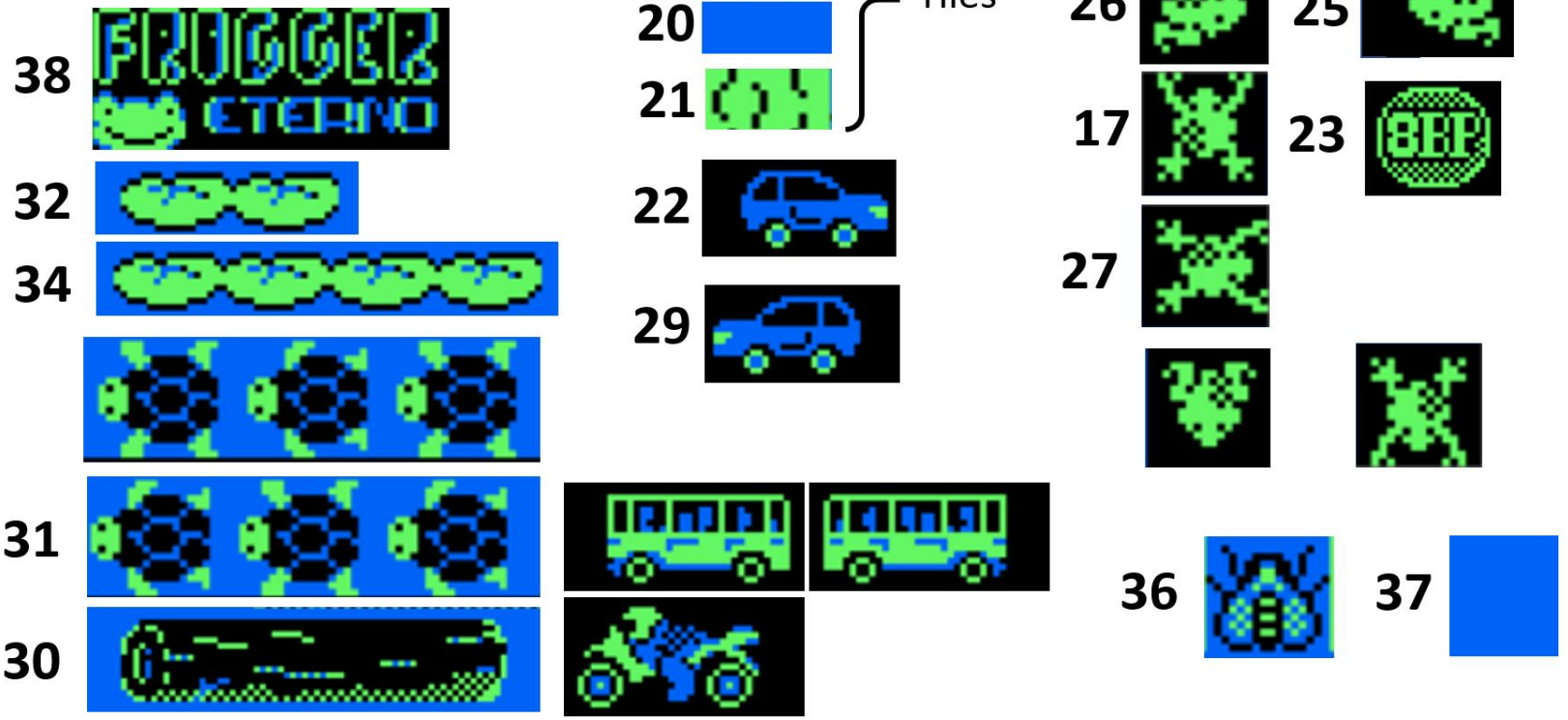
```
;=============================================
_BEGIN_FLIP_IMAGES
;=============================================
; aqui pon las imagenes que se definen como otras
; existentes pero flipeadas horizontalmente.
FROG_L1   dw FROG_R1
FROG_L2   dw FROG_R2
FROG_L1_DEL dw FROG_R1_DEL
;=============================================
_END_FLIP_IMAGES
;=============================================
```

FROG_L1

Se obtiene flipeando:

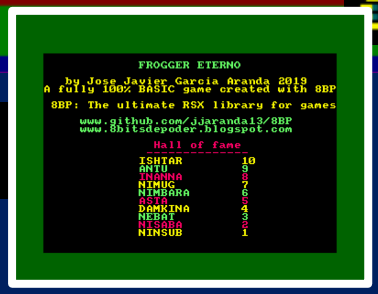
FROG_R1

Imágenes de “Frogger Eterno”



No todas necesitan identificador numérico

Inicio y Presentación



Lógica del programa principal

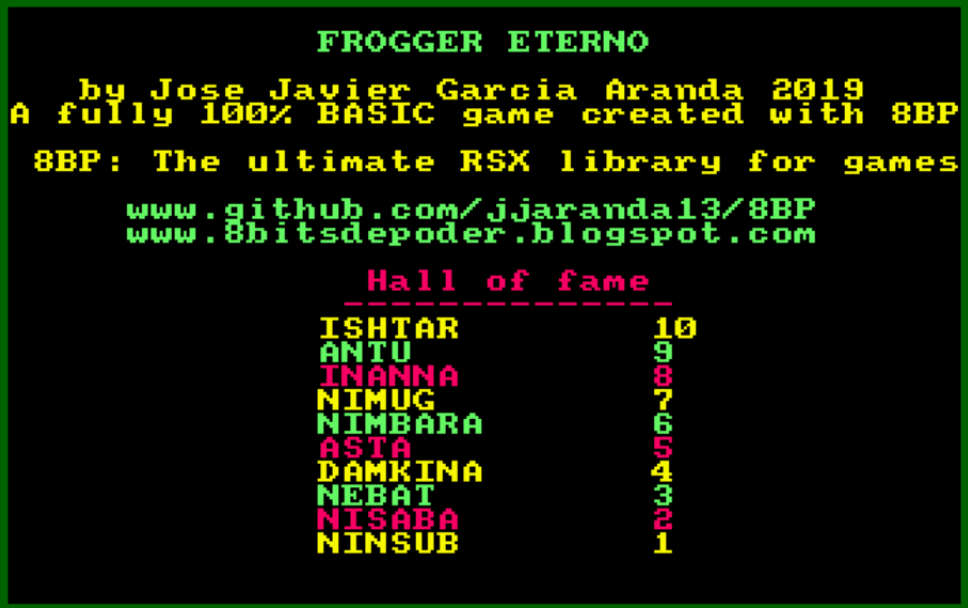
- GOSUB pantalla 1
- GOSUB pantalla 2
- GOTO FIN

Pantalla N:

- inicialización de sprites
- pintado de escenario
- ciclo de juego:
  - leer teclado y mover personaje
  - decisiones logicas
  - mover sprites
  - imprimir sprites
  - detectar colisiones y lógica asociada

Inicio y Presentación

```
10 MEMORY 23999
20 CALL &6B78
30 DEFINT A-Z
```

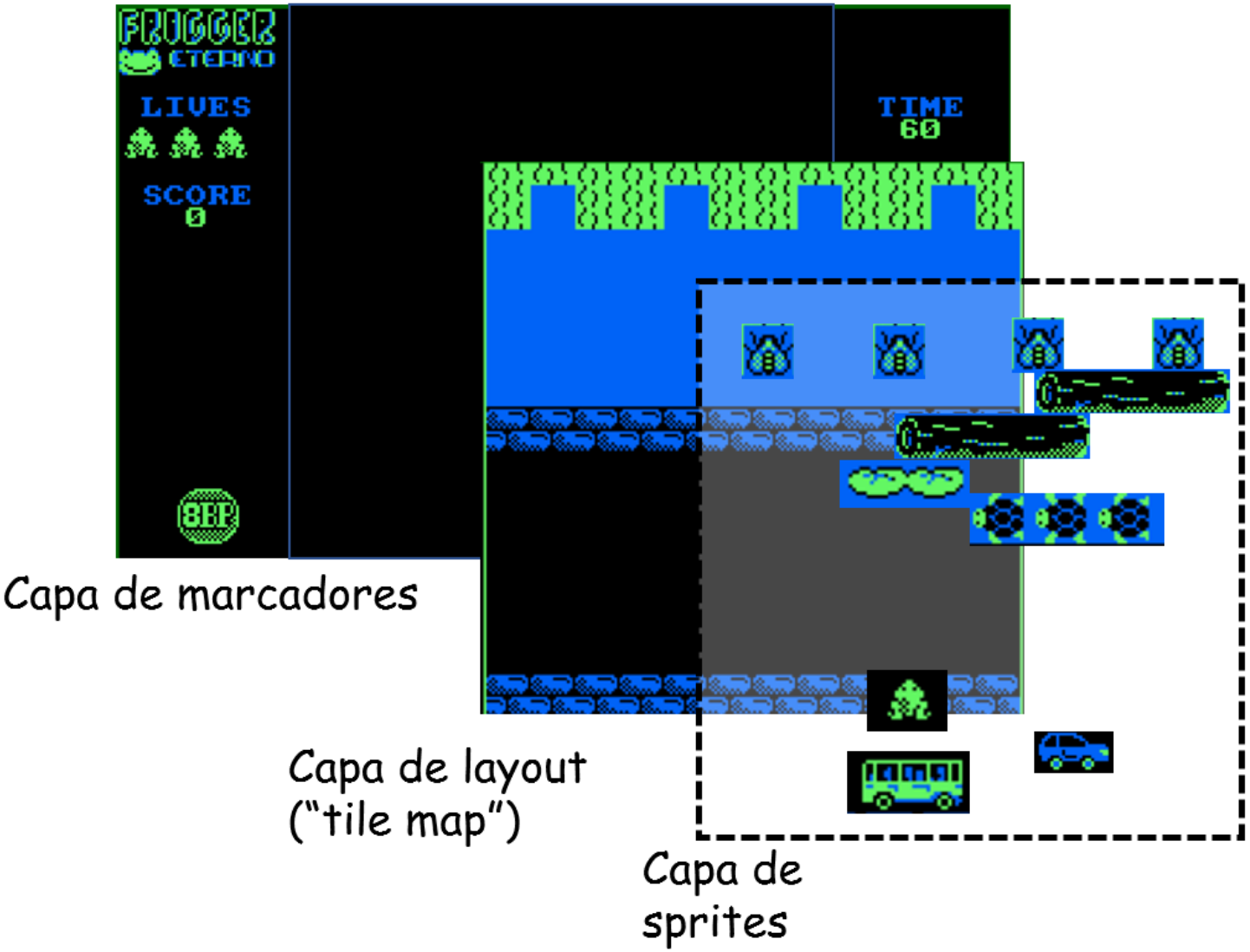


Lógica del programa principal

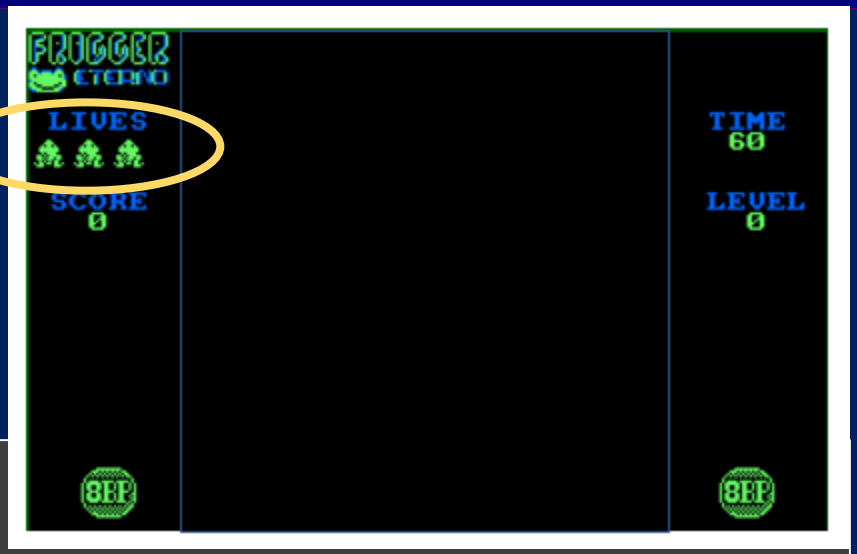
```
360 '---init game
370 lives=3: level=0:score=0:dificultad=30
380 'game logic
390 mosca(0)=1:mosca(1)=1:mosca(2)=1:mosca(3)=1:moscas=0:t=tmax
400 WHILE (lives>0 AND moscas<4)
410 FOR i=0 TO 31:|SETUPSP,i,0,0:NEXT:'reset
420 CLS:GOSUB 540
430 WEND
440 IF moscas<4 THEN 520
450 IF dificultad>15 THEN dificultad=dificultad-5
460 'clean buffer teclado
470 b$=INKEY$:IF b$<>"" THEN 470
480 CLS:PEN 2: LOCATE 12,10:PRINT "CONGRATULATIONS!"
490  LOCATE 11,12:PRINT "READY FOR LEVEL :";(30-dificultad)/5
500 PEN 1: LOCATE 14,20:PRINT "PRESS ANY KEY"
510 b$=INKEY$:IF b$="" THEN 510
520 IF lives>0 THEN 390
530 GOTO 1800:'fin del juego
```

De aquí retornas al perder una vida o al comer las 4 moscas

Diseño general



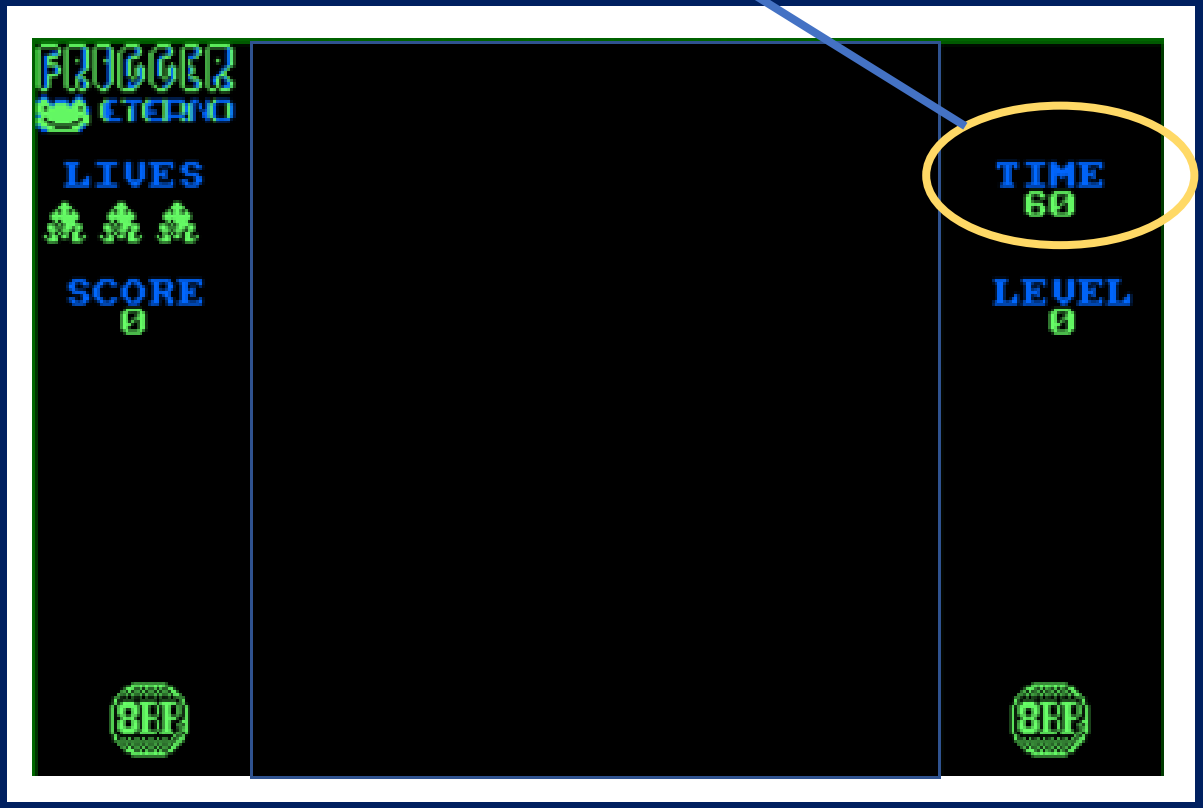
Capa de marcadores




```
560 |SETUPSP,31,9,frogup1
...
580 '--- marcadores
590 |SETLIMITS,0,80,0,200
600 |SETUPSP,0,0,1:|SETUPSP,0,9,frogger:|PRINTSP,0,0,0
610|SETUPSP,0,0,1:|SETUPSP,0,9,23:|PRINTSP,0,172,4:|PRINTSP,0
,172,68: |SETUPSP,0,0,0
620 PEN 1:LOCATE 2,5: PRINT "LIVES": LOCATE 35,5:PRINT
"TIME": PEN 2: LOCATE 35,6: PRINT t
630 LOCATE 2,9: PEN 1:PRINT "SCORE": LOCATE 3,10:PEN 2:PRINT
score
640 LOCATE 35,9: PEN 1:PRINT "LEVEL": PEN 2:LOCATE
36,10:PRINT (30-dificultad)/5
650 FOR i=0 TO LIVES-1:|PRINTSP,31,40,i*4:NEXT
```

Reducción del tiempo

```
1280 IF ciclo AND 31 THEN 1300
1290 t=t-1:LOCATE 35,6:PRINT t: IF t=0 THEN t=tmax: GOTO 1520
1300 …
```



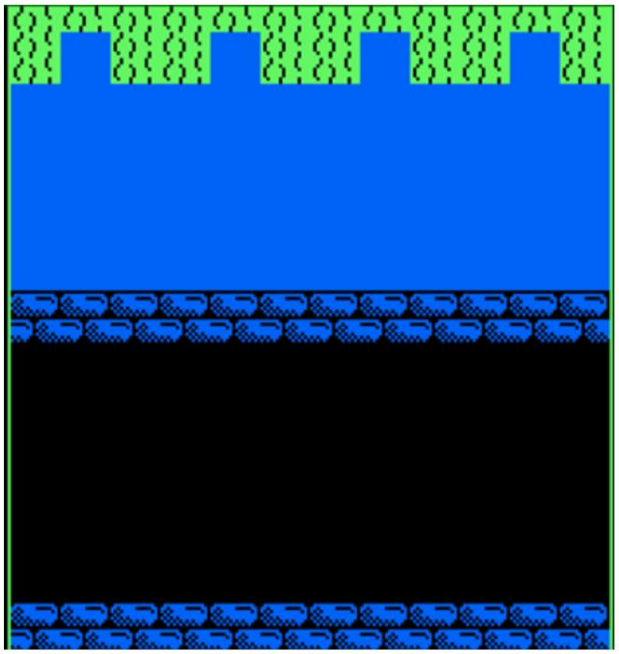
Capa de layout

|SETUPSP,6,9,Stone → Letra “A”

|SETUPSP,7,9,water → Letra “B”

|SETUPSP,8,9,stone2 → Letra “C”

|SETUPSP,9,9,sand → Letra “D”



```
690 c$(0) ="    DDDDDDDDDDDDDD    "
700 c$(1) ="    DDBDDBDDBDDBDD    "
710 c$(2) ="    DDBDDBDDBDDBDD    "
720 c$(3) ="    DBBBBBBBBBBBBD    "
730 c$(4) ="    DBBBBBBBBBBBBD    "
740 c$(5) ="    DBBBBBBBBBBBBD    "
750 c$(6) ="    DBBBBBBBBBBBBD    "
760 c$(7) ="    DBBBBBBBBBBBBD    "
770 c$(8) ="    DBBBBBBBBBBBBD    "
780 c$(9) ="    DBBBBBBBBBBBBD    "
790 c$(10)="    DBBBBBBBBBBBBD    "
800 c$(11)="    DAAAAAAAAAAAAD    "
810 c$(12)="    DCCCCCCCCCCCCD    "
820 c$(13)="    D            D    "
830 c$(14)="    D            D    "
840 c$(15)="    D            D    "
850 c$(16)="    D            D    "
860 c$(17)="    D            D    "
870 c$(18)="    D            D    "
880 c$(19)="    D            D    "
890 c$(20)="    D            D    "
900 c$(21)="    D            D    "
910 c$(22)="    D            D    "
920 c$(23)="    DAAAAAAAAAAAAD    "
930 c$(24)="    DCCCCCCCCCCCCD    "
```

```
940 |SETLIMITS,16,64,0,200
950 FOR i=0 TO 24:|LAYOUT,i,0,@c$(i):NEXT:'print layout
960 PLOT 31*4+2,0,2:DRAW 31*4+2,400
970 PLOT 640-32*4,0,2:DRAW 640-32*4,400
```

```
670 col%=0:|COLAY,67,31,@col%
```

Umbral de colisión

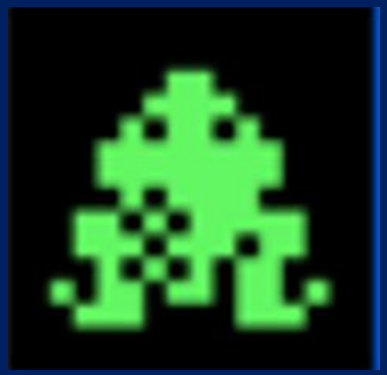
Caracter	Sprite id	Codigo ASCII
“ “	NINGUNO	32
“;”	0	59
“<”	1	60
“=”	2	61
“>”	3	62
“?”	4	63
“@”	5	64
“A”	6	65
“B”	7	66
“C”	8	67
“D”	9	68
“E”	10	69
…		

Capa de Sprites

```
540 '--- setup frog
550 |SETUPSP,31,0,1+64+32:'colider,transp,printable
560 |SETUPSP,31,9,frogup1
```

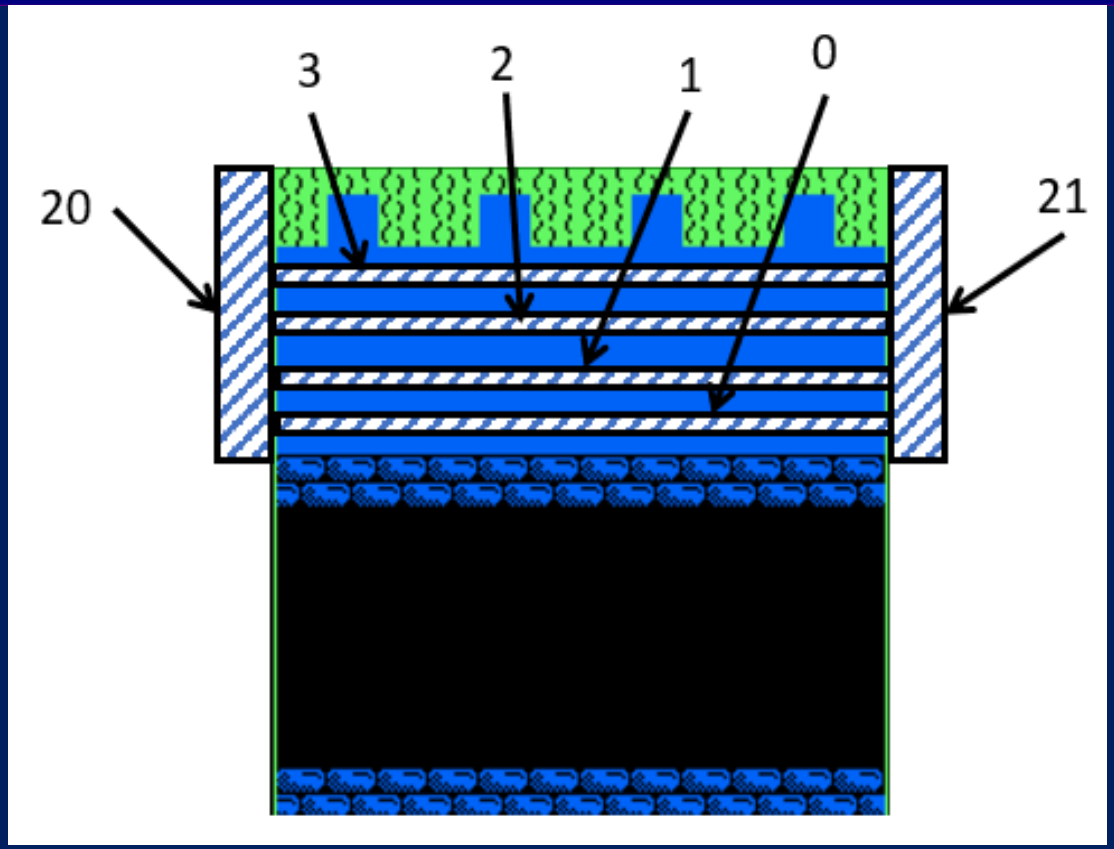
7	6	5	4	3	2	1	0
ROUTEALL lo ruta	Sobre-escritura	COLSPALL collider	MOVERALL lo mueve	AUTOALL lo mueve	ANIMALL lo anima	COLSP collided	PRINTSPALL lo imprime

64 32 1



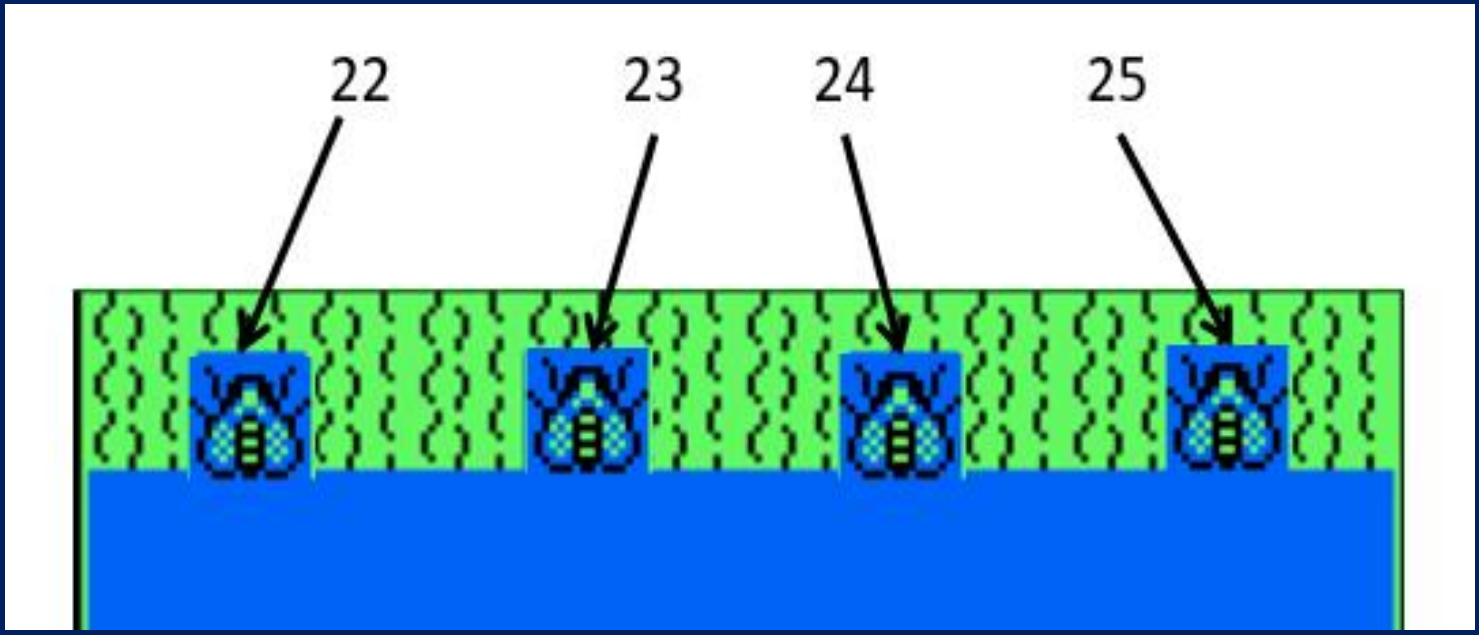
- 8BP soporta 32 sprites de cualquier tamaño
- 16 bytes por sprite (9 parámetros)
- Comienzan en 27000
- Podemos leer con PEEK o con |PEEK
- Podemos escribir sus parámetros con POKE o con |POKE o con |SETUPSP
- El primer byte es el de estado
- Los puedes colocar con |LOCATESP

	1byte	2 bytes	2 bytes	1byte	1byte	1byte	1byte	2 bytes	1byte
sprite	status	coordy	coordx	vy	vx	seq	frame	imagen	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
7	27112	27113	27115	27117	27118	27119	27120	27121	27127
8	27128	27129	27131	27133	27134	27135	27136	27137	27143
9	27144	27145	27147	27149	27150	27151	27152	27153	27159
10	27160	27161	27163	27165	27166	27167	27168	27169	27175
11	27176	27177	27179	27181	27182	27183	27184	27185	27191
12	27192	27193	27195	27197	27198	27199	27200	27201	27207
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



son
Sprites
no visibles
Pero
colisionables

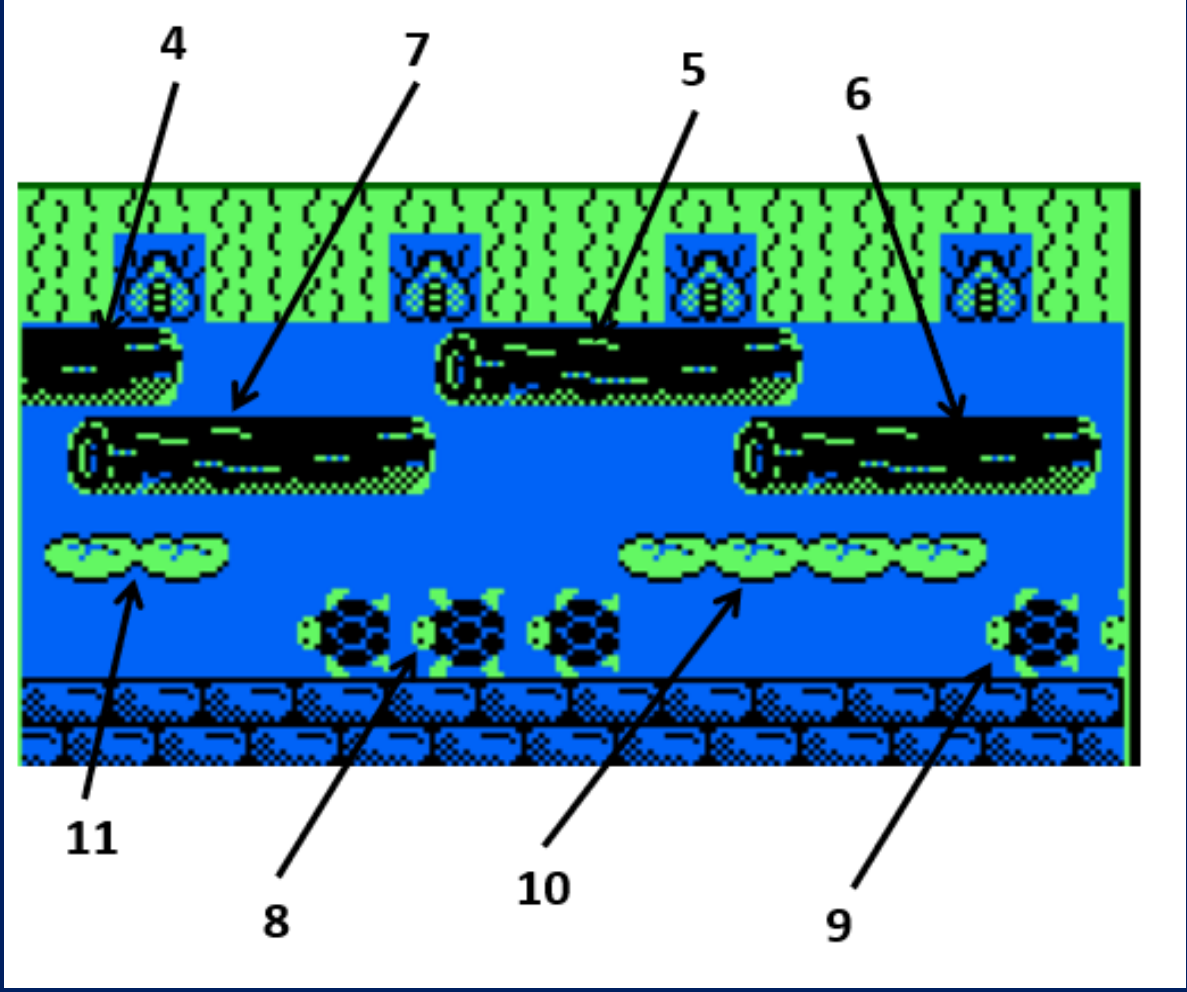
```
1070 |SETUPSP,0,0,2:|SETUPSP,0,9,river:|LOCATESP,0,72+4,16:'rio 4
1080 |SETUPSP,1,0,2:|SETUPSP,1,9,river:|LOCATESP,1,56+4,16:'rio 3
1090 |SETUPSP,2,0,2:|SETUPSP,2,9,river:|LOCATESP,2,40+4,16:'rio 2
1100 |SETUPSP,3,0,2:|SETUPSP,3,9,river:|LOCATESP,3,24+4,16:'rio 1
1110 |SETUPSP,20,0,2:|SETUPSP,20,9,muro:|LOCATESP,20,24,8:'MURO L
1120 |SETUPSP,21,0,2:|SETUPSP,21,9,muro:|LOCATESP,21,24,64:'MURO R
```



Si no te has comido aun a la mosca 0 , tendremos que crearla

```
1130 IF mosca(0) THEN
|SETUPSP,22,0,2:|SETUPSP,22,9,mosca:|LOCATESP,22,8,20:
|PRINTSP,22: ELSE |PRINTSP,31,8,20:'MOSCA 1
```

```
980 '--- setup river ---
990 |SETUPSP,4,0,128+8+2+1:|SETUPSP,4,9,tronco:|SETUPSP,4,15,10:
|LOCATESP,4,24,0:'tronco 1A
```



Lógica de ciclo de juego

```
1190 |PRINTSPALL,0,0,0,0:|COLSP,34,3,3:|COLSPALL,@collider%,@collided%
1200 |COLSP,32,0,25:'mas alla del 25 no hay colisionables
```

```
1201 '--- game cycle
1210 ciclo=ciclo+1: |AUTOALL,1:|PRINTSPALL:|COLSPALL
1220 IF demora THEN demora=demora-1
1230 IF demora GOTO 1260
1240 GOSUB 1600
1260 IF ciclo MOD dificultad THEN 1280
1270 GOSUB 1550
1280 IF ciclo AND 31 THEN 1300
1290 t=t-1:LOCATE 35,6:PRINT t: IF t=0 THEN t=tmax: GOTO 1520
1300 IF PEEK(27147)<90 THEN 1320
1310 |LOCATESP,8,72,60:|LOCATESP,9,72,90:'recoloca turtle
1320 IF collider<32 THEN 1340
1330 GOTO 1210
```

Control de la rana

- `IF INKEY(67) THEN 1660`
- `IF INKEY(67)>0 THEN 1660`

} Son equivalentes

```
1620 IF INKEY(67) THEN 1660:
    <Instrucciones si se pulsa Q y RETURN>
1660 IF INKEY(69) THEN 1710
```

```
1600 '-- Keyboard control
1610 y=PEEK(27497):x=PEEK(27499)
1620 IF INKEY(67) THEN 1660:
1630 |MOVER,31,-16,0:|COLAY,31,@col%:|MOVER,31,16,0:IF col% THEN RETURN
1640 POKE 27499,(x+1) AND 252:SOUND 1,638,0,0,1
1650 |SETUPSP,31,0,233:|SETUPSP,31,15,0:demora=maxdemora: arrastre=0:
RETURN
```

El numero 233 en binario es 11101001

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

```
ROUTE0; jump up
        db 253
        dw FROG_UP2
        db 1,0,0
        db 8,-2,0
        db 253
        dw FROG_UP1
        db 8,0,0; stop
        db 255,64+32+1,0
        db 1,0,0
        db 0
```

Asigna la imagen de rana saltando

Se queda quieta un ciclo

Se mueve durante 8 ciclos hacia arriba, de 2 en 2 líneas y nada en coordenada X

Asigna la imagen de rana en reposo

Se queda quieta durante 8 ciclos

Cambia el estado, quitando flag de ruta

Quieta un ciclo

fin

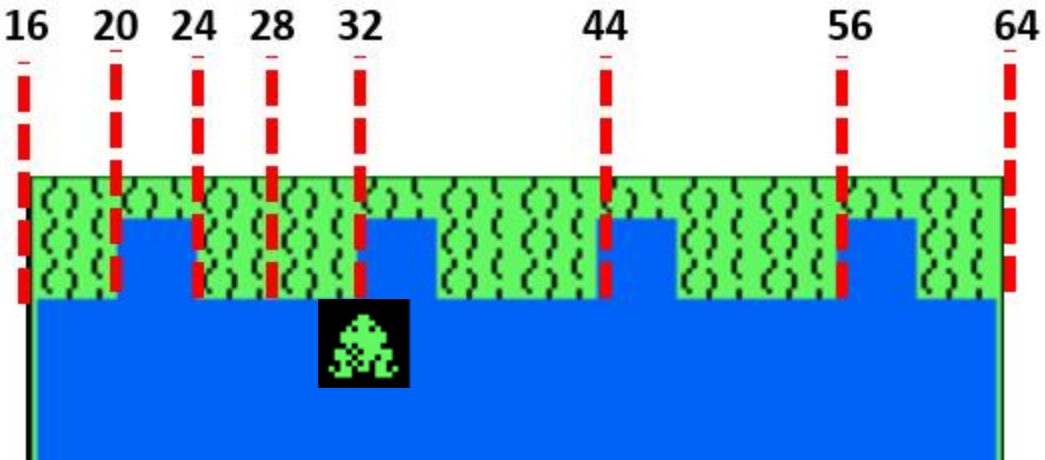
Control de la rana. alineación

POKE 27499,(x+1) AND 252

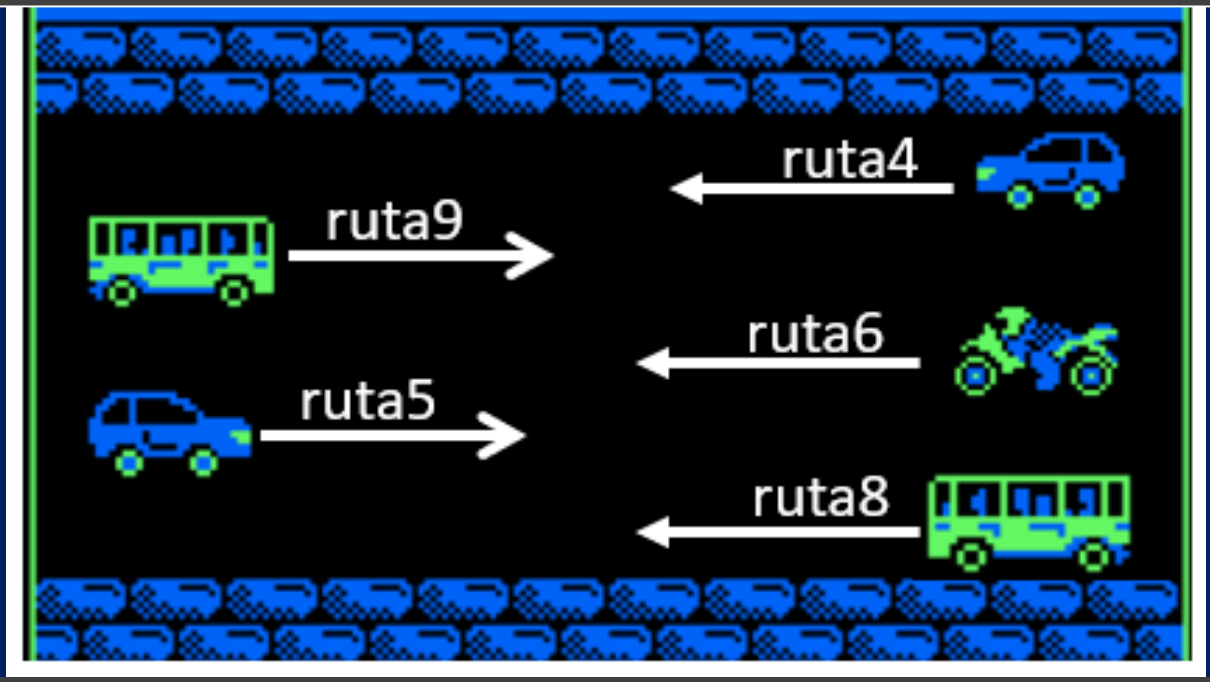
Básicamente esto lo que hace es sumar un 1 a la coordenada X y eliminar los 2 bits menos significativos. Por ejemplo:

- Si X= 3 (binario &x11) entonces X = (&x11) AND &x11111100 = &x100 =4
- Si X= 4 entonces X seguirá valiendo 4
- Si X= 5 (binario &x101) entonces X = (&x110) AND &x11111100 = &x100 =4
- Si X= 6 (&x110) entonces X = (&x111) AND &x11111100 = &x100 = 4
- Si X= 7 (&x111) entonces X =(&x1000) and &x11111100 = &x1000 =8
- Si X=8 se seguirá valiendo 8

3,4,5,6 → 4
7,8,9,10 → 8
11,12,13, 14→12
…



```
1540 '--- new car
1550 car=1+car AND 7:'max simultaneos cars = 8
1560 carsp=car+12:'12 en adelante 12+7=19, el 20 ya esta libre
1570 p=(p+1 + RND*5) MOD 5:'pista
1580 |SETUPSP,carsp,9,16:|SETUPSP,carsp,0,128+8+2+1:
|SETUPSP,carsp,15,r(p): |LOCATESP,carsp,yc(p),xc(p)
1590 RETURN
```




```
330 xc(0)=64:xc(1)=12:xc(2)=64:xc(3)=12:xc(4)=64
340 yc(0)=104:yc(1)=120:yc(2)=136:yc(3)=152:yc(4)=168
350 r(0)=4:r(1)=9:r(2)=6:r(3)=5:r(4)=8
```

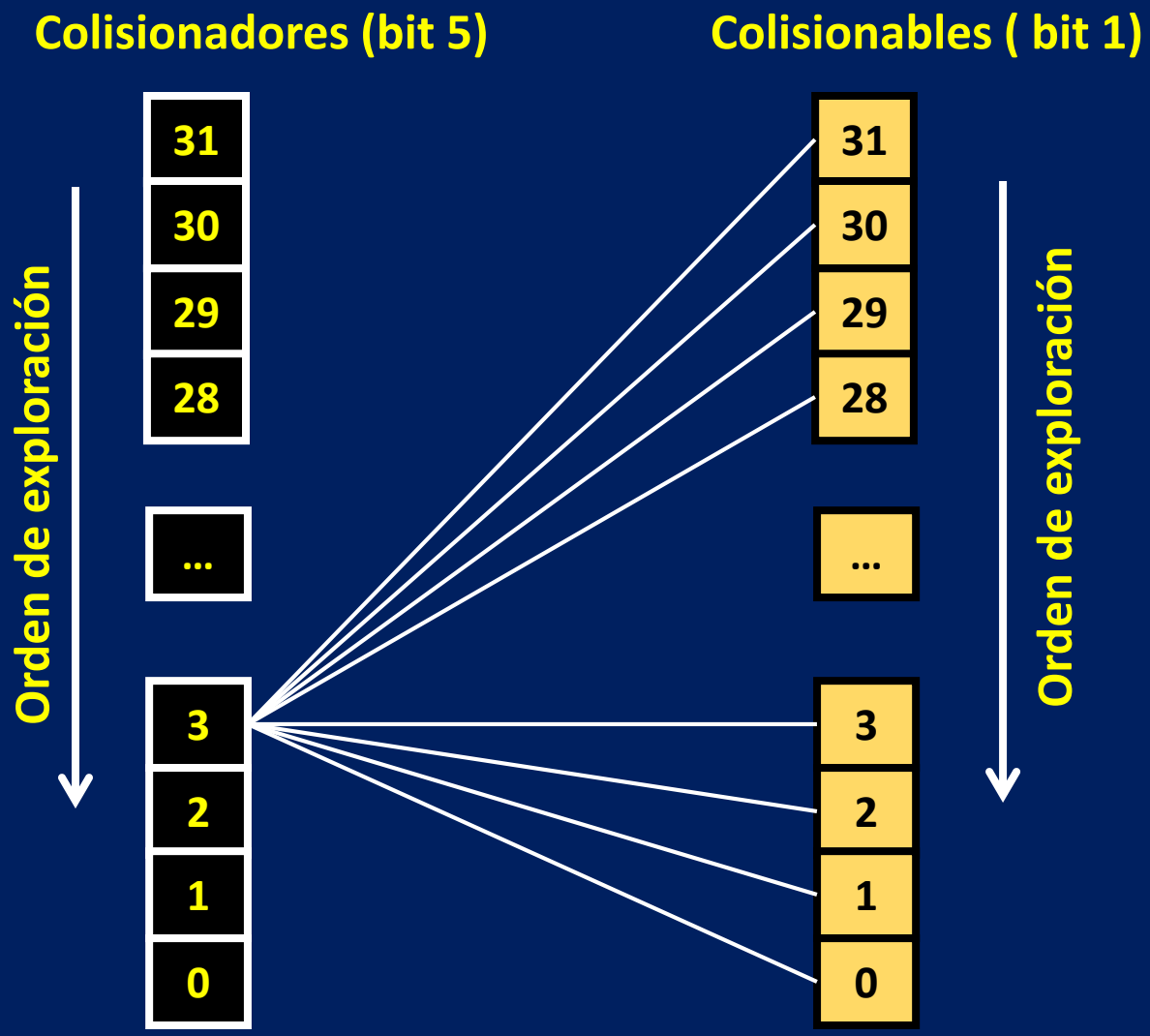
Rutina de colisión

```
1320 IF collider<32 THEN 1340
```

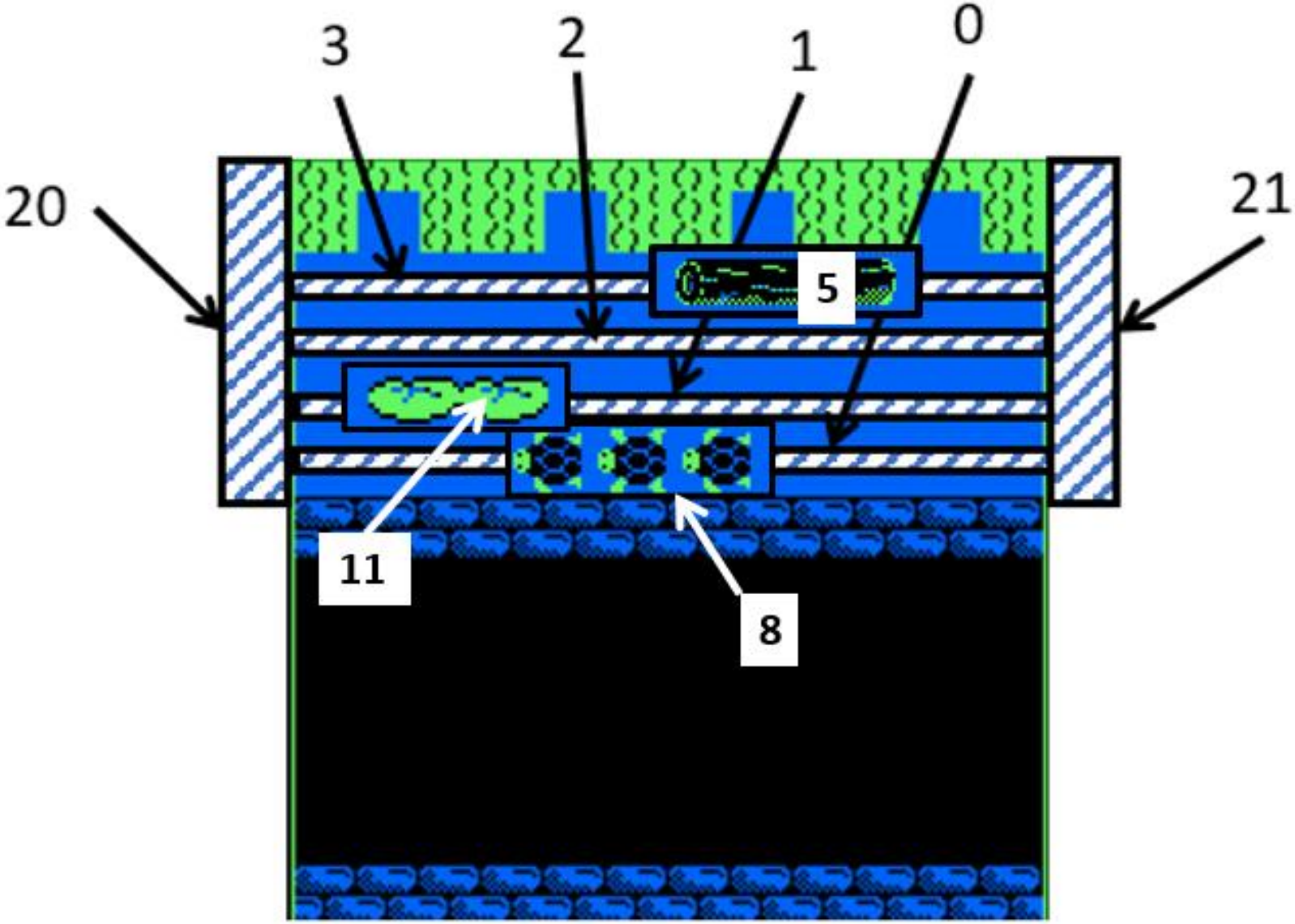
La rutina de colisión debe tener en cuenta todos los casos:

- **Colisión con enemigo (coche):** debe morir
- **Colisión con mosca:** debe comérsela y volver abajo del todo
- **Colisión con tronco, hojas o tortugas:** debe asignarse una ruta de "arrastre" a la rana en función de la velocidad y dirección que lleve el elemento al que se ha subido
- **Colisión con rio:** en el caso de que colisionemos con uno de los 4 ríos, debemos morir
- **Colisión con muros invisibles :** debe morir

Funcionamiento de |COLSPALL

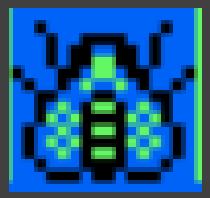


Funcionamiento de |COLSPALL

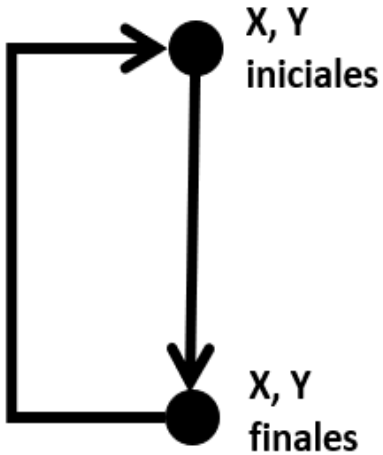


```
1340 '--- collision routine
1350 IF collided>11 THEN 1440
1370 IF collided<4 THEN IF demora=0 theb 1510 else 1210:'rio

1371 'hemos subido a un tronco.
1380 IF (demora+arrastre) THEN 1210
1390 IF collided<6 THEN |SETUPSP,31,0,233:|SETUPSP,31,15,14: arrastre=1:GOTO
1210:'tronco R slow
1400 IF collided<8 THEN |SETUPSP,31,0,233:|SETUPSP,31,15,13: arrastre=1:GOTO
1210:'tronco L fast
1410 IF collided<10 THEN |SETUPSP,31,0,233:|SETUPSP,31,15,16: arrastre=1:GOTO
1210:'turtles L slow
1420 IF collided<12 THEN |SETUPSP,31,0,233:|SETUPSP,31,15,18: arrastre=1:GOTO
1210:'hojas R fast
1430 GOTO 1210

1440 'colision con mosca o con enemigo
1450 IF collided<22 THEN 1520:'check mosca
1460 |SETUPSP,collided,9,watermosca: |PRINTSP,collided:|SETUPSP,collided,0,0
1470 SOUND 1,638,30,15,0,1: mosca(collided-22)=0
1480 score=score+10:LOCATE 3,10:PEN 2:PRINT score
1490 WHILE demora :|AUTOALL,1:|PRINTSPALL: demora=demora-1:WEND:'termina salto
1500 moscas=moscas+1:IF moscas=4 THEN RETURN ELSE GOTO 1180

1510 'colision enemigo
1520 SOUND 1,142,100,15,0,3:FOR i=1 TO 20:BORDER 7:CALL
&BD19:|PRINTSPALL,0,0,0,0,0:BORDER 0:CALL &BD19:NEXT
1530 lives=lives-1:RETURN
```

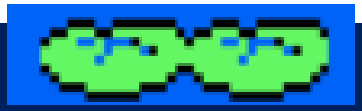


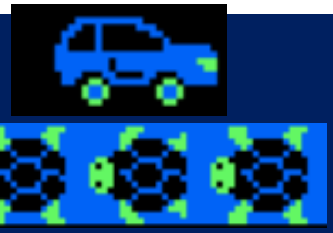

Tipos de rutas de sprites

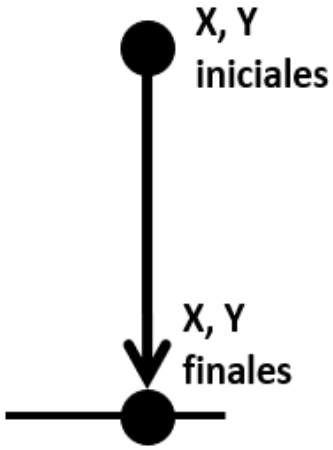


Ruta sin fin
cíclica

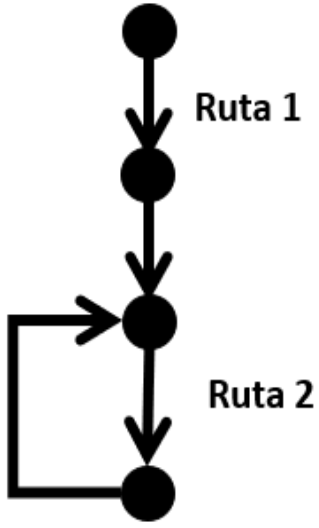


Ruta sin fin
no cíclica





Ruta con fin



Rutas
encadenadas

Rutas (fichero routes_mygame.asm)

```
ROUTE9; CAR SLOW R
        db 253
        dw BUS_R
        db 1,0,1
        db 1,0,0
        db 0
```

Asigna la imagen de bus

Mueve hacia derecha 1 byte

quieto

Vuelve a empezar

```
ROUTE6; CAR FAST L
        db 253
        dw MOTO_L
        db 28,0,-2
        db 255,0,0
        db 1,0,0
        db 0
```

Asigna la imagen de moto

Mueve hacia izquierda 2 bytes durante 28 ciclos

Cambia estado a inactivo

quieto

fin

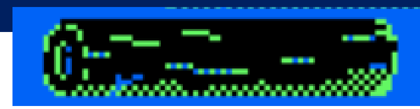
Ruta sin fin no cíclica. Requiere recolocación desde BASIC

```
ROUTE15; turtles slow L
        db 253
        DW turtlesa
        db 1,0,-1
        db 1,0,0
        db 253
        DW turtlesb
        db 3,0,0
        db 0
```

Mueve hacia izquierda 1 byte

quieto

Ruta sin fin cíclica. NO Requiere recolocación desde BASIC



```
        ; rutas de troncos
        ;-----------------
ROUTE10; tronco  R lento
        db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0
        db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0
        db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0
        db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0
        db 252,11,0; pasamos a la ruta 11
        db 1,0,0
        db 0

ROUTE11; continuacion de ruta 10 (una ruta no puede medir mas de 255)
        db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0
        db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0
        db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0
        db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,0,0
        db 1,0,-64
        db 252,10,0
        db 1,0,0
        db 0
```

recoloca

```
db 1,0,1,2,0,0,1,0,1,2,0,0,1,0,1,2,
db 252,11,0; pasamos a la ruta 11
db 1,0,0
db 0
```

Empieza
la aventura de programar!

consejos:

1. No escatimes tiempo en la creación de gráficos
2. Empieza por hacer algo sencillo y hazlo crecer

https://8bitsdepoder.blogspot.com
https://github.com/jjaranda13/8BP