“HAPPY MONTY”

making of

1 pantalla de carga del juego

El videojuego “happy monty” ha sido programado en Locomotive BASIC usando la librería 8BP (8 bits de poder). Creado por 8BP (Jose Javier Garcia Aranda).

La librería 8BP son un conjunto de rutinas escritas en ensamblador que se pueden usar desde BASIC gracias a las extensiones RSX. Es posible igualmente usar estas rutinas desde cualquier otro lenguaje e incluso desde un programa en ensamblador. El videojuego Happy Monty esta creado en Locomotive BASIC y se ejecuta interpretado, no compilado.

8BP mantiene una relación de cooperación con la AUA de modo que ambos logos son mostrados con el objeto de difundir la cultura y valores de el “arte retro”.

La música ha sido creada con WYZtracker 2.0.1.0 ya que dentro de la librería 8BP esta integrado el player de WYZ.

La lógica BASIC del juego ha ocupado 17 KB aproximadamente, aunque el código fuente incluye muchos comentarios de modo que se podría reducir algo. El código fuente será público tras el concurso cpcretrodev, como todos los desarrollos de 8BP (tanto la librería como los juegos son públicos).

Los retos de este desarrollo han sido:

- **Lograr una velocidad aceptable de juego**. Teniendo en cuenta que la lógica esta en BASIC y que el intérprete esta consumiendo muchísimo tiempo, la estrategia de programación de lógicas masivas ha resultado indispensable. La velocidad alcanzada

depende de cada pantalla. Hay pantallas a 27 FPS y otras que solo alcanzan 18 FPS. También se aprecia como al mover el personaje bajan algo los FPS, asi como al activar la música, que es opcional.

- **Incluir un numero elevado de niveles**: se han incorporado 25 niveles mediante una descripción compacta de los mismos que tan solo gasta 160 bytes por nivel. Los niveles se numeran del 0 al 24. El nivel 0 esta inspirado en el primer nivel del videojuego "mutant Monty" creado por John Price en 1984 para la compañía Artic Computing. El juego happy Monty destina en total 25*160bytes= 4kB para todo el mapeado de pantallas.
- **Rutas versátiles:** Para evitar definir decenas de rutas de enemigos de distinta longitud, se ha optado por definir unas pocas rutas básicas (vertical y horizontal de 3 velocidades) y unos sprites invisibles que actúan a modo de "inversores", de modo que cuando un enemigo colisiona con un inversor, cambia la dirección de movimiento. Este mecanismo de los "inversores" ha también posibilitado la creación de rutas con giros de 90 grados, pues se han definido inversores para cambiar a la dirección opuesta y también inversores para cambiar de ruta y asi implementar los giros que dan algunos enemigos como los que hay en el nivel 1.
- **Control del personaje "avanzado":** si pulsas una tecla, el personaje se mueve despacio, pero si la dejas pulsada unos instantes, Monty acelera y va más deprisa.

El videojuego posee 25 niveles, pero una sola lógica para todos ellos.

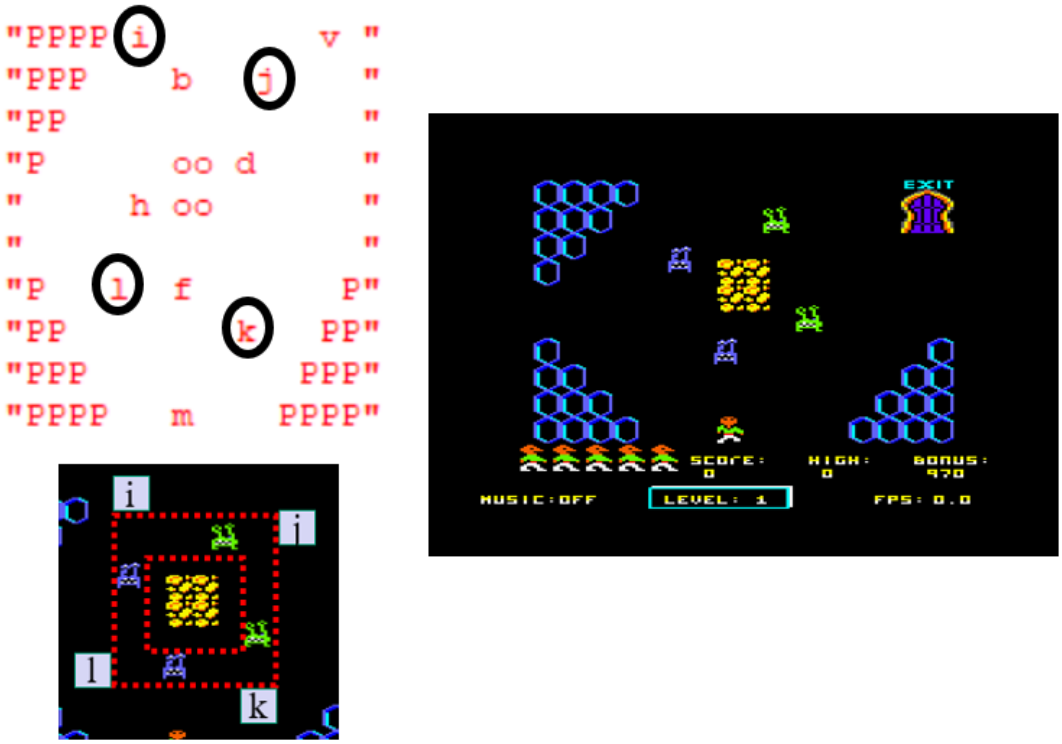
Cada nivel se ha definido con un conjunto de letras tanto los muros y elementos ornamentales como los sprites enemigos ("a", "b,", "c", "d,", "e", "f", "g", "h") el oro("o"), los sprites "Inversores" ("z", "x", "q", "w") y la posición de Monty ("m"). En el siguiente ejemplo podemos ver la definición de un nivel, que como se puede comprobar, ocupa 160 caracteres y es muy fácil de editar.

Era posible definir el nivel de un modo aun mas compacto, gastando unos pocos bits por bloque en lugar de un carácter. Sin embargo, al utilizar caracteres, la creación de nuevos niveles ha sido una tarea mucho más sencilla.

```
"IK m    o     JG"
" IGGGGGH IGGGGHq'
" z  c     xo    '
" C CCGK        d'
" F oC IDD DDDDD  '
" F  C  F      F  '
" Fv C JHz b xoF  '
" F  C IGGGGGGGH  '
" IGGH z    a  xw'
"      EEEEEoEEE "
```

2. Nivel 0 del juego y su definición con 160 bytes

En el siguiente nivel se ven los sprites inversores capaces de hacer girar a los enemigos 90 grados al colisionar con ellos, con lo que se producen rutas de enemigos más "peligrosas". Los sprites "inversores" son los denominados con las letras "i","j","k","l".



3. Sprites inversores de tipo "codo"

Las "lógicas masivas" están presentes en todo el desarrollo: desde las interioridades algorítmicas de los comandos propios de la librería 8BP (tales como la colisión de sprites) hasta la forma en que se lee el teclado (en ciclos pares se leen las teclas Q,A y en ciclos impares las O,P. Además, cada 6 ciclos se comprueban las colisiones con el oro, y la puerta de salida, evitando hacerlo en cada ciclo y de ese modo acelerando el juego. La pulsación de la tecla "m" para activar y desactivar la música se comprueba también cada 6 ciclos. Otras de las cosas que tienen que ver con esta técnica es la detección de colisión con el layout, que solo se realiza cuando las coordenadas (x,y) de Monty son múltiplos de 4 y 16 respectivamente, de modo que solo si se encuentra en una bifurcación se permite el cambio de dirección del personaje. Esto ahorra tiempo de cómputo al tiempo que impide que el personaje pueda quedar parado entre dos muros. Cuando Monty se para, lo hace siempre en una posición múltiplo de 4 (en X) y múltiplo de 16 (en Y).

Para acelerar aun mas, las rutas de enemigos "lentos" (especificados con las letras "a" y "e" en los mapas) , fuerzan cada 2 frames la desactivación del flag de impresión que utiliza el comando PRINTSPALL de la librería 8BP, de modo que hay ciclos en los que no se imprimen si no se han movido, ahorrando algo de tiempo. Esto se ha hecho gracias a las capacidades que tiene 8BP para definir rutas de sprites.

Para acelerar un poco mas el juego, se ha perfeccionado la librería 8BP, acelerando algunos comandos como COLAY y MOVER. En el caso de COLAY (rutina que detecta colisiones con el layout), se le ha dotado de "memoria", de modo que no es necesario invocarlo con parámetros si son los mismos que la ultima vez. Esta es una característica que comparten muchos otros comandos de 8BP, tales como PRINTSPALL o STARS, pero faltaba en COLAY. EL paso de parámetros es una de las cosas mas costosas en BASIC pues el interprete analiza cada parámetro, su formato, etc. Y conviene evitarlo si es posible.

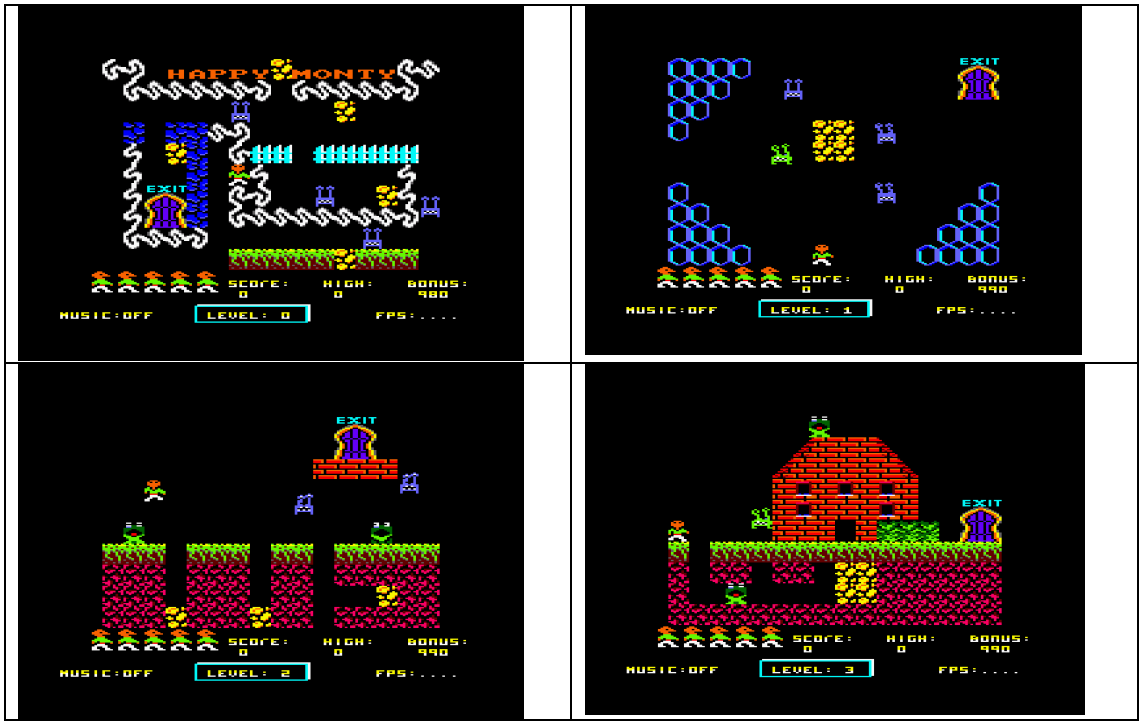
Por último, muchos de los comandos RSX de la librería 8BP en el ciclo de juego se han reemplazado por su correspondiente CALL, lo cual permite acelerar algo su ejecución, aunque a costa de que el código fuente sea algo menos legible.

El juego incluye un guiño al juego “Astro Marine Corps”: cada vez que Monty es alcanzado con un enemigo, el monstruo muestra durante unos instantes el famoso “gronf!!”, mientras suena el fatídico sonido que te indica que has perdido una vida.

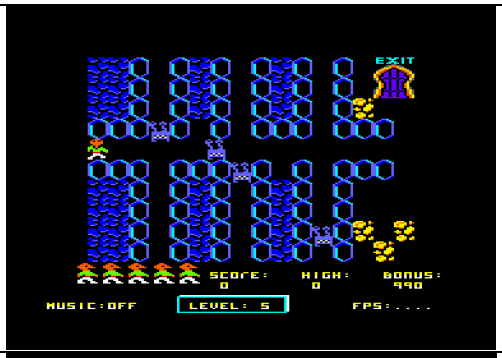
4. Guiño al famoso juego AMC

En el último nivel aparece la princesa, que no está presente en el resto de niveles. Tras recoger el oro y dirigirnos a ella nos espera un épico final

Los 25 niveles de Happy Monty son los siguientes:




EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 4
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 5
FPS:....

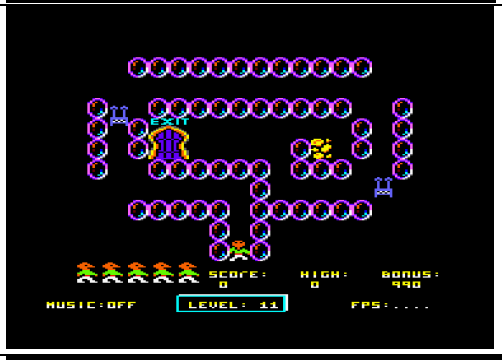

EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 6
FPS:....

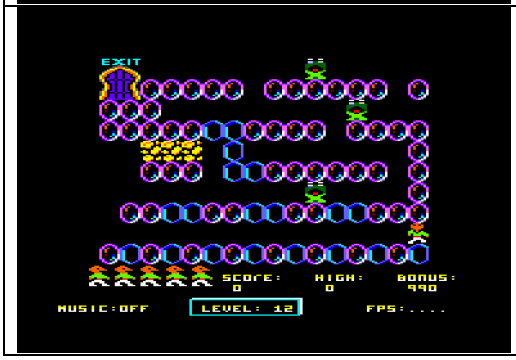

EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 7
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 980
MUSIC:OFF
LEVEL: 8
FPS:....

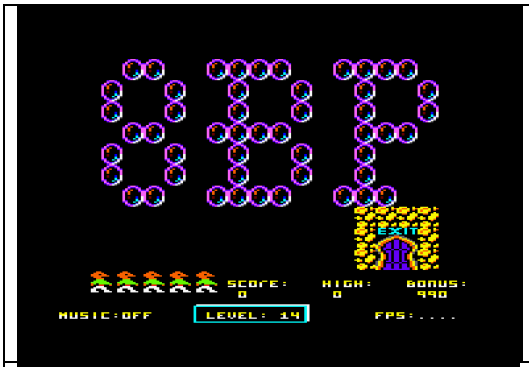

EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 9
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 10
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 11
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 12
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 13
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 14
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 970
MUSIC:OFF
LEVEL: 15
FPS: 0.0

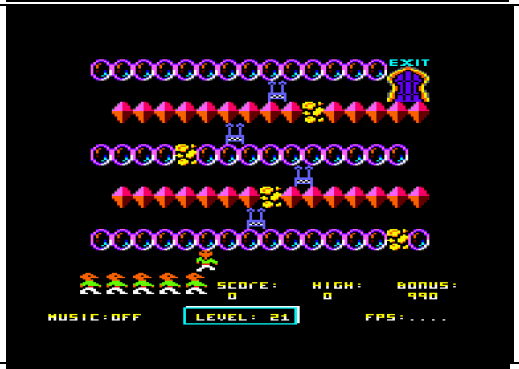

EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 16
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 17
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 18
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 980
MUSIC:OFF
LEVEL: 19
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 20
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 21
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 990
MUSIC:OFF
LEVEL: 22
FPS:....


EXIT
SCORE: 0
HIGH: 0
BONUS: 980
MUSIC:OFF
LEVEL: 23
FPS:....

Todos los diseños de sprites y decorados han sido construidos con la herramienta SPEDIT, que viene incluida en 8BP. SPEDIT es una herramienta sencilla escrita en BASIC que permite editar en mode 0 y mode 1, permite espejar sprites, editar y generar código fuente para hacer paletas, etc. Se ejecuta en el mismo AMSTRAD.

Por último, la pantalla de carga ha sido realizada con la herramienta convImgcpc v0.14

Respecto al mapa de memoria del juego, es el siguiente:

- 0000 hasta 17000 : el videojuego en basic
- 20000 hasta 240000: los 25 niveles, a 160 bytes cada uno
- 24000 hasta 32200: librería 8BP
- 32200 hasta 33600 : musicas (hay 2 musicas, la segunda suena al final del juego)
- 33600 hasta 42040: graficos
- 42040 hasta 42540: aquí se almacena el layout que se este ejecutando. Aunque una pantalla ocupa 160 bytes, se usan 500 bytes como buffer de layout en 8bp
- 42540 hasta 42619 : banco de estrellas (se usan en el final del juego)