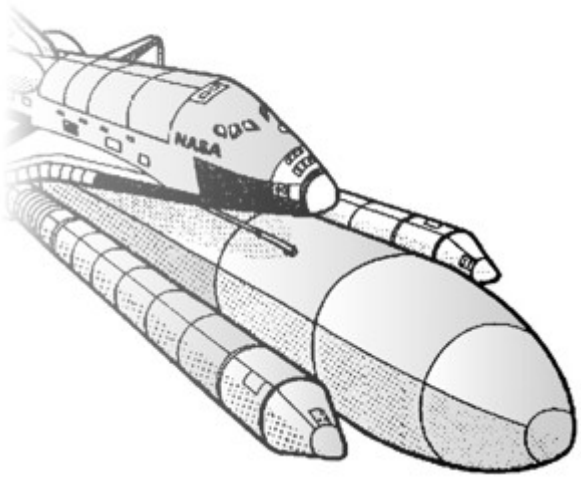


Programación de juegos en BASIC Sinclair

I

Aracnav paso a paso – Creando un juego de marcianitos.

(Compatible con ZX Spectrum 48k)

---

www.speccy.org/czarg

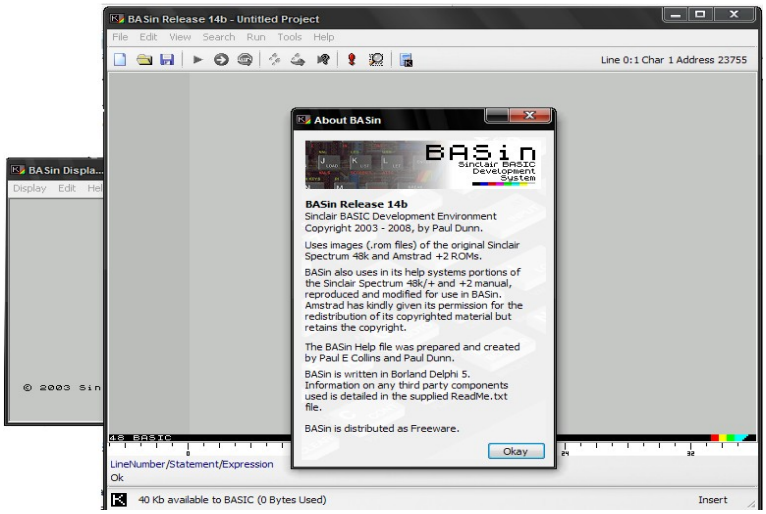
Introducción

Animarme a escribir un tutorial sobre creación de juegos para Spectrum parece una gracia debido a que seguramente no estoy a la altura de otros que podrían enseñarle mas, y no es que me este tirando abajo, simplemente tengo que reconocer que el trabajo en 48k de muchos otros programadores es realmente admirable. Pero desgraciadamente o por suerte, ellos están en otro punto de la escena y pocos son los que prestan atención a los iniciados o aquellos que quieren hacer algo y poseen dudas. Lo que quiero decir es que no soy un programador de alto rango, pero tampoco un iniciado, así que eso es lo que me lleva a tener ganas de escribir este tutorial. Otra pauta a tener en cuenta es la poca o desperdigada información que existe en Internet acerca de programación de esta índole.

Asi que bien, Basic, es un lenguaje muy sencillo y si le ha puesto ya las manos encima en una Spectrum, un emulador o interprete, de seguro no tendrá problema en seguir este sencillo tutorial. ¿Qué no sabe que es Basic?, primero lea la historia en CZArg y luego abra el manual del Spectrum, no tardará en estar nuevamente por aquí.

Para el aprendizaje que iremos viendo en esta y las futuras entregas, he decidido elegir como punto central, juegos que yo haya programado, con el fin de poder otorgarle más detalle acerca de cómo usted puede programar algo similar y modificarlo a su gusto. Sin embargo no descartaré el uso de rutinas externas (de otros juegos) para ejemplificar sucesos. Como editor de Basic Sinclair para PC usaré Basin, el cual, entre otras características, no se basa en los comando por letras, es práctico y realmente flexible.

Bien, manos a la obra.



Basin R14d

Nota: Para simplificar y resumir algunas explicaciones, he optado que los caracteres “>>” signifiquen ir o accionar un menú o acción correspondiente. Las notas en Azul serán aclaratorias y las rojas pertenecerán a las advertencias. Las instrucciones (Comandos claves) de Basic se expresarán en Negritas y color azul, mientras que las variables se expresarán en Negritas y rojo. Los datos importantes se remarcarán con Negritas y en cursivas.

Primeros pasos

Vamos a empezar aclarando que juegos como Aracnav son extremadamente sencillos de programar. El juego participó en el concurso 2006 de Basic Bytemaniacos y poseé solo 10 líneas. El juego se basa en no dejar que los arácnidos toquen “el suelo” (la parte inferior de la pantalla). Esto lo logrará poniendo la “nave” debajo de los mismos y disparando para deshacer la bajada. Entenderá mejor la explicación si lo juega.

Sin embargo, explicar un juego con palabras es quizás una de las mejores maneras de empezar, por ejemplo, cuando usted dice “Si estoy debajo del arácnido y disparo, entonces el arácnido muere y entonces sumaré puntos”; en realidad esta mencionando, en este caso, la parte más importante del juego, por lo que Básicamente ya lo tiene.

Una vez que vio cuál es la lógica del juego, comenzará a darle forma. Lo primero será definir los caracteres gráficos del usuario o GDU. Hoy en día la mejor forma que he encontrado para hacer esto es usar el editor GDU de Basin, el cual luego de crear los caracteres, permite generar los **DATA** y los **POKES** correspondientes. En este caso son 2, el de la nave y el de la araña.

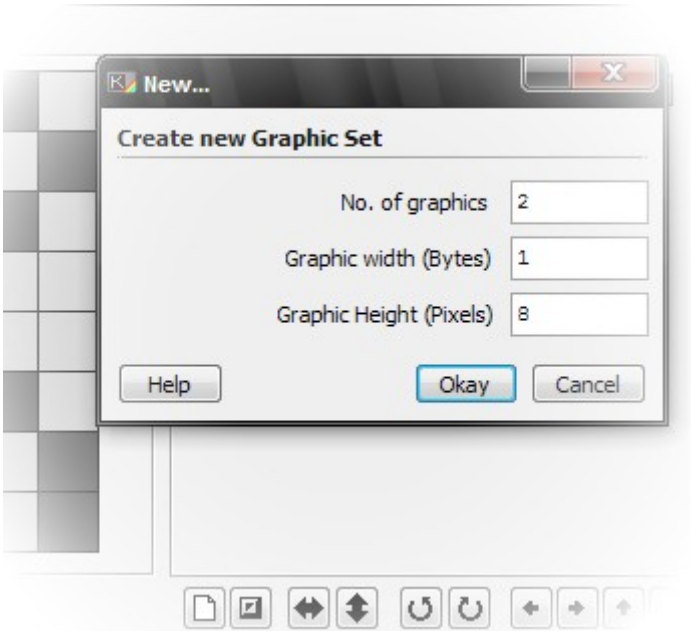
En la práctica es realmente sencillo, abra Basin y siga estos pasos:

>> Tools (Herramientas)

>> Graphics/Sprite Editor (Editor de Gráficos/Sprites).

>> File (Archivo)

>> New (Nuevo)



Pantalla de creación de gráficos GDU en Basin

En la pantalla que aparece cambie el valor de ***No of Graphics*** por el valor de 2. Esto sirve para poder informar al editor que va a redefinir 2 gráficos en vez de 21. Dibujará ambos, asignando la letra A a la nave y B a la araña.

Una vez terminados le resta enviar los datos al programa, para así, cargar los graficos cuando el juego se ejecute.

Para enviarlos vaya al menú:

>> File

>> Export (Exportar).

La pantalla a continuación muestra la forma en la que puede exportar los gráficos. En este caso enviará los datos como ***DATA Dec.***

Rellene los datos siguientes con estos valores:

Bytes per line: 10 (Bytes por linea)

Start: 9000 (Comenzar en la linea 9000)

Step: 10 (Saltar de 10 en 10)

Y como ultimo debería marcar la casilla ***Include POKEs***, para incluir las sentencias. Le da a Okay y estará listo para seguir.

Ya tiene las primeras líneas listas, al ejecutar el listado obtendrá los caracteres gráficos en A y en B para usarlos en su juego. Debería ejecutarlas, así cada vez que tenga que incluir los caracteres gráficos le será más sencillo.

Más adelante deberá incluir un **Gosub** a la línea 9000 para cargar desde un principio los gráficos.

El resto es comenzar a tipear en el Basin, de forma directa, igual que lo haría en cualquier editor de textos. Las palabras se le irán completando en caso de que introduzca un comando.

IMPORTANTE: Esta rutina puede borrarse mientras la creación del juego está en marcha, pero debe incluirse en la versión final para ver correctamente el juego.

El Bucle

Sea cual sea el juego que este haciendo, tiene que saber que todos los juegos se basan en lo que se llama Game Loop o bucle de juego. Un bucle es algo que se repite una y otra vez hasta que, por alguna razón previamente instaurada por el programador (en este caso usted), se detenga. En dicha repetición se hacen varias cosas, que en general siempre son las mismas:

- Mover los personajes (Gdu u otros)
- Comprobación de cosas (Vidas, tiempo, etc)
- Mostrar gráficos

(No necesariamente en este orden)

Aracnav, pese a tener 10 líneas, no es la excepción. Si mira el listado, dicho bucle se encuentra entre las líneas 10, a partir de la instrucción **For**, y la línea 70 con la instrucción **Next** .

Es ese el lapso en el que el juego sucede, o sea, todo lo que entre esas líneas sucede es lo realmente relevante, el resto es pura decoración. En la línea 5 está la presentación y explicación de las bases del juego, en la línea 9000 empezará la creación de GDUs.

Definir las variables

Ya sabe cómo está estructurado el programa, ahora verá que es lo que hace cada linea.

Entender las variables de un juego es extremadamente importante para saber cómo funciona el mismo.

Algunas variables de Aracnav se encuentran fuera del bucle principal, esto hace que mientras el juego se esté ejecutando, las variables no regresen a su valor inicial.

Las variables especifican varias cosas tales como lugar de la nave, lugar del enemigo, puntaje, etc.

Las variables de Aracnav son:

Definidas primeramente fuera del Bucle principal:

Y= 15 Posición de nuestra nave en horizontal, el valor 15 significa que de un principio apareceremos en la columna 15.

Punt=0 Puntos logrados, en principio 0.

Equiv=0 Equivocaciones en los disparos, en principio 0.

Las últimas dos variables son puestas en pantalla en la línea 10 mediante instrucciones **Print** antes de comenzar el bucle principal

Definidas primeramente dentro del bucle principal

A Valor en vertical que irá tomando la araña (de 0 a 19)

B Valor aleatorio horizontal que irá tomando la araña (de 0 a 30)

C$ Imprime el arácnido.

Quizás las 2 variables más importantes sean **Y** y **B**, ya que cuando estas son iguales y al mismo tiempo se efectúa un disparo, efectivamente el valor de **Punt** aumenta.

El juego

Ya hablamos del bucle y las variables de forma conceptual, ahora le explicaré como conjugarlos para lograr que el juego funcione.

Discriminando la primera linea que dejaré a su imaginación, empezaré por la sentencia **For** de la linea 10, donde comienza el bucle.

Programa:

```
10 PRINT #0;"   ARACNAV   ";"RES:
";equiv;"  PUNTOS: ";punt:
FOR a=1 TO 19:
BEEP .001,a+10:
IF punt=200 THEN
PRINT AT 10,4;"GANASTE, MISION
CUMPLIDA!":
BEEP .1,6:
BEEP .6,2:
BEEP .2,5:
BEEP .4,3:
INPUT #0;"Teclea m para volver a
empezar ";l$:
IF l$="m" THEN CLS :
GO TO 1
20 IF a=1 THEN
LET b=INT (RND*30)
30 LET c$="A":
PAUSE 2:
PRINT AT a,b; INK 2;c$:
PRINT AT 21,y; INK 1;"B":
LET equiv=equiv+1:
IF equiv=400 THEN
```

```
PRINT AT 10,0;"   PERDISTE, LOS
ARACNIDOS          HAN TOMADO LA
CIUDAD":
BEEP .1,-20:
INPUT #0;"Teclea m para volver a empezar
";l$:
IF l$="m" THEN
CLS :
GO TO 1
```

El **For** de la linea 10 hace variar a la variable **A** de 1 a 19, eso hace que el bucle se repita 19 veces, en esas veces pasarán varias cosas.

- Se ejecuta un sonido que va en escala a medida que se modifica **A** (de 0 a 19)
- Se verifica si la variable **Punt** es igual a 200 y muestra un mensaje, o sea, si hizo 200 puntos, entonces se mostrará en pantalla que ha ganado, se ejecutará una rutina musical mediante Beeps y se esperará a que teclee “m” para recomenzar.
- En la linea 20 encontrará un **If,** el cual verifica si la variable **A** esta en uno (o sea al principio del bucle); si es asi elige una posición (número) de 0 a 30 al azar y la almacena en **B**, para luego posicionar el arácnido, el cual esta almacenado en la variable **C$**, en la posición A,B. Por ejemplo, sabe que **A** va a ser 1,2,3,4…19; esto simulará que la araña venga “cayendo” y **B** al ser aleatorio hará que la araña caiga en 2,8,9,4, o en cualquier otra posición que no supere los 30. Claro esta que este ultimo número se da por el tamaño de pantalla en baja resolución de la Spectrum.
- En la linea 30 se imprime el arácnido y luego la nave. La posición de la nave no es aleatoria como la del arácnido. La nave tiene el valor vertical estipulado en 21, la última linea de pantalla (abajo) y un valor horizontal manejado por el usuario, el cual se guardará en la variable **Y**. Es importante que dentro de las comillas en la que colocamos la nave exista un espacio entre la primera comilla y el gráfico y la segunda y el gráfico. Esto es bastante común, ya que la nave va a moverse, el espacio evita que si por ejemplo se mueve hacia la izquierda, la nave no quede pintada hacia la derecha. Digamos que se da movimiento mientras se deja un espacio en blanco para borrar el resto donde estuvo.
- Luego se suma la variable **Equiv.** Ya expliqué que el valor de **Equiv** es igual a 0, y esto es importante. Tenga en cuenta que la variable se va a modificar dentro del bucle, por lo que cada vez que acabe el bucle y vuelva a empezar, dicha variable habrá sumado en realidad 19 equivocaciones. Fíjese que esto es curioso, porque hace que el juego premie al jugador que pegue al aracnido apenas sale, ya que si le pega cuando el bucle esta en 5, el bucle se reiniciará y no existirán 19 equivocaciones sino 5.
- Verifica si la cantidad de equivocaciones es igual a 400 hace que el juego se termine.

El movimiento

He decidido detenerme debido a que las últimas 3 líneas del bucle son realmente importantes. No significa que el resto no lo sea, pero en este punto aprenderá algo que seguramente aplicará de ahora en más a muchos juegos. Y eso es el movimiento de nuestra nave y el impacto del disparo contra el enemigo. Esta pequeña rutina es clásica y puede ser ampliada y modificada de miles de formas.

La rutina empieza en la línea 60 y hace uso de la instrucción **Inkey$ (If)** para "averiguar" si la tecla que se presiona es, en este caso, la "n". Si esto es correcto, toma la variable **Y** (posición de nuestra nave) y le resta 1. Al repetir el bucle, la línea 30 volverá a imprimir la nave, ahora un lugar desplazada a la izquierda.

Programa:

```
60 IF INKEY$="n" THEN
LET y=y-1
65 IF INKEY$="m" THEN
LET y=y+1
```

En la línea 65 sucede lo mismo, en este caso la variable es sumada para que ahora la nave se desplace 1 lugar a la derecha. Obviamente en dicha línea se cambia la tecla a pulsar, en este caso será la "m".

Cuando mencioné que Aracnav era un juego extremadamente sencillo, no lo hice en vano ya que es en esta parte donde queda demostrado. Aracnav no se basa en ningún sistema de colores como otros juegos para saber si el disparo le ha dado o no a la araña. Aracnav lo hace de una manera simple pero no por eso menos efectiva. Verá otros métodos en otro tutorial.

El juego compara la posición vertical de la nave con la del arácnido y si en ese momento se efectúa un disparo, se habrá acertado el tiro. Las dos variables a comparar son: **Y** (la ubicación de la nave) y **B** (posición de salida del arácnido). Claro esta que para esto hay que primero recordar cómo la línea 30 ha impreso la nave en pantalla: **comillas, espacio, nave, espacio, comillas.** Esto hace que cuando los dos estén en la misma posición, en realidad la nave este gráficamente desplazada un lugar a la izquierda (por el espacio que imprime para lograr el movimiento). Para solucionar esto esta la primera instrucción de la línea 68.

Programa:

```
68 LET g=b-1
```

Lo que en realidad se hace en este punto es declarar la variable **G**, la cual será igual a **B** (la posición del arácnido) menos 1. Como se ve, lo que se hace es hacer que la variable **G** sea la que decida la posición real para la comparación. Entonces, para que un disparo sea válido la nave debe estar en una posición y el arácnido debe estar en esa posición menos un lugar, para saltear el espacio y estar justo debajo de la nave.

Nuevamente se hace uso del **Inkey$** y se comprueba que si se pulsa “z” e **Y=G** entonces imprime un mensaje de “BANG” en la misma posición en la que el arácnido se encuentre en ese momento. Inmediatamente se suman 10 puntos y seguidos de unos sonidos, se limpia la pantalla y se regresa a la linea 10.

Programa:

```
IF INKEY$="z" AND y=g THEN
PRINT AT a,b;"BANG!":
BEEP .1,40:
BEEP .1,50:
LET punt=punt+10:
CLS :
GO TO 10
```

Fin del Bucle, fin del juego

Hasta aquí ha llegado el bucle, con la instrucción **Next A**, que le dice a la Spectrum que regrese nuevamente al bucle siempre y cuando no se haya acabado por alguna razón. En ese caso el programa volverá a comenzar.

Notas finales

Como verá, esto es mucha explicación y poco juego, pero le aseguro que le será útil para entender básicamente la lógica de un juego. Basin resulta un poco complicado al principio, más si usted siempre ha programado en Spectrum y no en la PC. Sin embargo resulta indispensable para programar en Basic y seguramente una vez que le tome la mano no podrá dejar de usarlo. Quizás ya se ha puesto a pensar en cómo modificarlo para lograr su propio juego, si aún no lo hizo, sepa que existe una comunidad que lo está esperando, y que hoy en día es una buena oportunidad para participar en concursos y ganar muy bueno premios, así que adelante ¿Qué espera?.

Dónde conseguir las herramientas:

Basin

http://paul.dunn.googlepages.com/BASin_r14d.exe

Aracnav y otros juegos de Palamar Software

http://www.speccy.org/czarg

Spectaculator (El mejor emulador de Spectrum)

http://www.spectaculator.com

http://oscomp.hu/dlgame.php?276

(versión gratuita)

Fuentes

Curso Basic de la revista Microhobby.

Motor de plataformas en basic, blog TCYR:

http://tcyr.wordpress.com/2007/11/20/programacion-1-motor-de-plataformas-basic-spectrum/

Zx Spectrum, cómo obtener el máximo rendimiento; Diaz de Santos; Ian Sinclair.

Dudas y/o comentarios a: palamarweb@yahoo.com.ar

Tutorial Modificado el Sábado 10 de Enero de 2009