Creando juegos con gráficos

Baltasar, el arquero

Introducción

Está claro que uno de los grandes avances que propone la nueva máquina virtual de Glulx es la capacidad de ofrecer gráficos en las aventuras, utilizando un sistema estándar, basado en una arquitectura fuerte (32 bits), y no otros propietarios (aunque abiertos y completamente gratuitos, por supuesto) como podrían ser, por ejemplo, Iznoguz.

Sin embargo, hoy por hoy no existen demasiados juegos con gráficos. Es cierto que aquí entra la eterna discusión acerca de si los gráficos son necesarios o incluso convenientes en una aventura conversacional: sin embargo, también es cierto, ciñéndonos a argumentos más técnicos, que no está demasiado claro (al margen de algunos ejemplos, por supuesto, en la web de Zak [2]), cómo añadirle gráficos a un juego de una forma sencilla.

El objetivo de este pequeño tutorial es desvelar los misterios [3] de la realización de este tipo de juegos, empezando por reunir las distintas librerías y herramientas que nos hacen falta, paso por paso. Por otra parte, es necesario saber cómo utilizar las librerías desde dentro del programa. Finalmente, será necesario empaquetar todo lo que necesitamos en un fichero *blb* para poder distribuirlo. Quede claro que buscaremos el método de que el programador vea afectada su manera de trabajar en la medida más pequeña posible. El objetivo por ahora es incorporar gráficos a las localidades y posiblemente a algunos objetos, como en las aventuras clásicas y no hacer virguerías. Para complicarnos la vida están el futuro y la librería *Glk* al desnudo.

Reuniendo las piezas

Que quede claro: vamos a utilizar Glulx, por las razones descritas: es una arquitectura de 32 bits, nacida con idea de convertirse en un estándar, y con soporte para todos los posibles adornos que le queramos añadir a nuestra aventura. Iznoguz [8], pese a ser una gran iniciativa en su momento, no deja de ser un arreglo muy interesante, pero dadas las limitaciones de la máquina Z, hay que evolucionar a Glulx. Por otra parte es previsible que poco a poco vayan apareciendo intérpretes Glulx para todas las plataformas, hasta conformar un universo de intérpretes tan tupido como el *frotz* y sus variantes. En este tutorial se tratarán las herramientas y métodos para Windows, por ser la opción más extendida [4].

1. La página de Zak es nuestro punto de partida para llegar al objetivo final: nuestra aventura con gráficos. La primera parada del camino es por supuesto, el compilador *inform* biplataforma, versión 6.21. Ésto se obtiene fácilmente de las páginas de Zak, sin pérdida, pues tiene su propio enlace principal. La librería InformATE! 6.10 biplataforma también es necesaria, y también tiene su propio enlace principal. Es muy importante que ésta librería sea la versión más reciente, es decir, no descargues la librería 6/10 inicial, sino la beta (o incluso otra más reciente), o el juego no compilará correctamente. Todo esto se descomprime junto en un directorio, que a partir de ahora se supondrá es `c:\inform`.
2. El siguiente elemento a recopilar es el paquete de herramientas *iblorb*. Estas herramientas se utilizan para empaquetar los recursos (gráficos, sonidos) que necesitaremos en el juego en un solo fichero *blb*, lo cuál es lo más

conveniente. Este paquete lo encontraremos en la parte final de la página a la que nos lleva el enlace principal de Glulx. Ya que estamos aquí, podemos bajarnos el Glulxe, el intérprete de Glulx necesario para ejecutar cualquier aventurilla que creemos.

3. Nuestro deseo es no complicarnos la vida, así que vamos a utilizar la librería SIX que amablemente pone a nuestra disposición Zak. Es necesario bajarse la Six multiplataforma, para que al compilar las llamadas al sistema se realicen mediante la famosa función glk(fn, arg), y no mediante el ensamblador de la máquina Z. Ésta librería puede encontrarse en la zona de librerías y módulos para InformATE!, en la página principal. Como comprobación, la versión necesaria de SIX es la 2.1. Poned especial cuidado en **no** bajaros la versión 1.0 en las páginas de *Iznoguz*.
4. Vamos a bajarnos también la demo `sixdemo.zip`. Esta demo puede encontrarse de la zona de ficheros de la página de Zak. Puede llegarse a través de la zona dedicada a *Iznoguz*. De esta aventurilla podemos aprender mucho para futura referencia, y además, trae una librería ya hecha (inventar la rueda, no, por favor) que nos facilitará mucho la vida. Se trata de `grafdemo.h`.

Todos estos ficheros se descomprimirán en el mismo directorio que el compilador de *inform* (que asumimos como `c:\inform`).

Creando una pequeña aventura

Nuestro objetivo es tratar de sentirnos lo menos intimidados posible por el uso de gráficos. De hecho, por el momento, crearemos una aventura sin gráficos. En otro tutorial [5] escrito por el autor [6], se creaba una aventurilla de una sola localidad. Esta aventurilla era *abismo*, para InformATE!. Podrás encontrar el fuente `abismo.inf` (ya corregido para gráficos) en el mismo paquete que el presente tutorial [5]. También en el mismo lugar encontrarás el primer tutorial, donde es posible encontrar el código de abismo sin arreglos para gráficos de ningún tipo.

En esta aventura de ejemplo, basada en una escena de ***Cacahuetes, Sal y Aceite*** [5], el jugador se encuentra ante un abismo que debe cruzar. No parece tener ninguna manera de hacerlo, hasta que examinando el saliente de roca, se encuentra con bastante arena, que lanza hacia delante para descubrir un pequeño puente invisible hasta ese momento.

La forma de compilar este fichero, `abismo.inf`, para Glulx, se explica a continuación. Si compilamos la versión del primer tutorial, no encontraremos ningún problema, sin embargo la versión del presente tutorial, aunque puede compilarse de esta forma, no generará una aventura con gráficos, ya que hay que empaquetar los archivos gráficos. De todas formas, en general, la compilación para Glulx se realiza de la siguiente, simple manera: `informbp –G abismo.inf`. Con esto obtenemos un fichero *.ulx*, que es el fichero de código ejecutable. Por supuesto, este fichero no contiene ningún gráfico, sin embargo, sí que es ejecutable –como parece lógico- desde el intérprete Glulxe. Más adelante, se añadirán los gráficos a esta aventurilla. Cuando se pida compilar el código de la aventura, tendremos simplemente que volver a ejecutar esta línea de código, desde el *prompt* o símbolo de sistema, que debería ser `c:\inform>`. Por tanto, la compilación tendría el siguiente aspecto (aunque veremos más adelante que podemos facilitarnos la vida con la herramienta *front*):

```
c:\inform> informbp –G abismo.inf
```

Ésta forma de hacer las cosas no es casualidad. No debe olvidarse que los gráficos, sonidos ... etc., sólo son adornos (bueno, más o menos, o al menos de la forma

introductoria con la que los vamos a tratar aquí), y que siempre (ésto siempre es cierto) se debe prestar una atención especial al texto, el guión y la interactividad del juego.

Añadiendo los gráficos

El siguiente paso es planear, sin duda, cuáles van a ser los gráficos, por objeto y por localidad. En este caso, se plantearán tres: un gráfico para la localidad del abismo, otro para la misma localidad, cuando el puente aparece, y otro para el objeto arena. Los gráficos pueden hacerse en cualquier editor que soporte el formato o bien *jpg* o bien *png*. En este caso, los ficheros serán `abismo.jpg, abismopuente.jpg` y `arena.jpg`, respectivamente. Pueden encontrarse, de nuevo, en la misma descarga de este tutorial [5] [7].

Para complicarnos la vida lo menos posible, se utilizará la librería *grafdemo*, para la utilización de gráficos de una forma sencilla.

Así, sólo debemos poner el *include* de la librería al principio del código *.inf*, y un *include* de unas definiciones que serán generadas más tarde. Es **muy** importante colocar los *includes* de la siguiente manera (los nuevos *includes* son los tres del medio):

```
Include "EParser";
Include "grafdemo";
Include "six";
Include "abismo.bli";
Include "Acciones";
```

Abismo.bli todavía no existe, pero se generará a partir del fichero de recursos que debe llamarse *abismo.res*. Se trata de las asociaciones entre nombres simbólicos y gráficos o sonidos, como veremos enseguida.

Por otra parte, ahora hay que añadir la propiedad '*numgráfico*' a aquellos objetos que queramos que dispongan de gráfico.

Así, en la localidad del abismo (la única que hay), incluimos la línea:

```
Object  elAbismo "Al borde del abismo."
with irrelevante 'rocas' 'roca',
   arena 0,
   numgrafico graf_abismo, ! ésta es la nueva línea
...
has luz;
```

Y en la definición del objeto arena, la siguiente:

```
Object -> arenaloca "arena"
with
      numgrafico graf_arena, ! ésta es la nueva línea
      nombre 'arena' 'granos' 'montoncitos',
...
;
```

Ahora, sólo es necesario un detalle: cuando el puente aparece, debemos cambiar el gráfico del abismo por el gráfico del abismo con puente.

En el código referente a la acción lanzar arena, en el objeto arena al que acabamos de dar su número de gráfico:

```
     descripcion
           "Montoncitos de arena gorda ...^",
     antes [;
               echaA: if (otro==negroabismo){
                         elAbismo.numgrafico = graf_abismopuente;
                          elAbismo.arena = 1;
                          ++puntuacion;
                          remove arenaloca;
                              "Al caer la arena al abismo, parte queda
sobre una .....
```

De esta forma, cuando el jugador realice la acción esperada, el gráfico de la localidad cambiará justo antes de pedirle el siguiente turno al jugador.

Estos nuevos nombres simbólicos, que hemos utilizado a lo largo del programa, todavía no están “conectados” con los gráficos reales. Esto es lo que es necesario hacer, empaquetando los gráficos con la aventura. Las relaciones necesarias son:

Fichero gráfico	*Nombre simbólico*
Arena.jpg	graf_arena
Abismo.jpg	graf_abismo
Abismopuente.jpg	graf_abismopuente

Y para eso comenzamos el proceso de empaquetado de la aventura con sus gráficos. Y para esto, el primer paso es escribir un fichero de recursos, acerca de los ficheros que vamos a meter en el mismo fichero *blorb*. El fichero de recursos es como sigue, teniendo en cuenta la anterior tabla, y las necesidades de la librería grafdemo.h:

```
CODE abismo.ulx
PICTURE graf_abismo abismo.jpg
PICTURE graf_arena arena.jpg
PICTURE graf_abismopuente abismopuente.jpg
PICTURE Sin_imagen sin_imagen.jpg
PICTURE Oscuridad oscuridad.jpg
PICTURE Fondo fondo.jpg
PICTURE Marco marco.jpg
```

Yo he cambiado *oscuridad_jpg* por *oscuridad*, *Fondo_jpg* por *Fondo*, y *marco_png* por *Marco*. Esto lo he hecho simplemente abriendo el fichero grafdemo.h y editándolo, cambiando las apariciones de *oscuridad_jpg* por *oscuridad*, y así sucesivamente. En el *zip* [5] de este tutorial puedes encontrar el fichero grafdemo.h cambiado con estos detalles. Si no quieres cambiarlo, no tienes porqué hacerlo.

La primera línea del *.res* indica qué fichero contiene el código ejecutable Glulx de la aventura. Las siguientes, indican los recursos que utiliza la misma: las tres primeras se corresponden con los que ya hemos referenciado en la aventura, *graf_abismo*, *graf_arena* y *graf_abismopuente* (que de nuevo pueden encontrarse en el sitio de descarga [5]). Los siguientes gráficos están ahí por requerimiento de *grafdemo.h*, y son el gráfico a pintar cuando no hay gráfico en una localidad (cosa que no sucede en nuestra aventura, pero que es útil para el caso genérico), el gráfico que se utilizará como fondo, y el gráfico que se utilizará para pintar el marco del gráfico.

Sólo queda hacer que en la rutina de inicialización se inicialice la ventana gráfica. Esto lo conseguimos con el siguiente código, que utiliza los *.jpg* de fondo y el marco para dibujar la ventana base. Antes de eso, se realiza una pequeña comprobación acerca de que el intérprete acepte gráficos.

```
[ Inicializar;
      if (~~ComprobarInterprete())
      {
            print "^^Lo siento, este juego precisa un
                     intérprete Glulx con posibilidades
                     gráficas.^^";
            EsperarTecla();
            quit;
      }
      else {
      print      "^Te adentraste en la cueva del tesoro,
       y ... recuerdas
       perfectamente como al llegar a la cámara secreta del rey,
       activaste una trampa que podría haber sido mortal ...^
       Buscabas la riqueza, pero ahora te conformas con salir vivo ...
      ^^";
       EsperarTecla();
      }
```

```
      ! Inicialización de la aventura
      localizacion = elAbismo;
      move mapa to jugador;

      ! Inicialización gràfica
      PonerAltodeVentanaGrafica(200);
      BorrarGraficos();
      PintaMarco(true);
];
```

Tal y como se ve, las modificaciones son mínimas, y se pueden tomar como base para cualquier otra aventura. La línea más importante es la primera de "inicialización gráfica", dependiendo de los píxeles que utilicemos en los gráficos el tamaño de la ventana.

Ahora es el momento de compilarlo todo. Es necesario disponer de los *.jpg* que utiliza la aventura [5]. Podemos hacerlo por pasos, pero nuestro objetivo es complicarnos la vida lo menos posible. La línea siguiente compilará la aventura generando el fichero *.blb*, que ya es el ejecutable Glulx. El primer nombre se trata del fichero de recursos (sin el *.res*), y a continuación todas las opciones que le pasarías normalmente al compilador *inform*. En este caso, le pasamos –G para que genere código Glulx y el nombre del código fuente.

```
front abismo –G abismo.inf
```

Para probar la aventura, tan sólo es necesario arrastrar y soltar el fichero *abismo.blb* generado sobre el intérprete Glulxe.

Conclusiones

Las aventuras con gráficos tienen tantos partidarios como detractores. Pero difícilmente se podrá juzgar algo sin conocerlo (es decir, en su estado actual), y tienen que aparecer suficientes aventuras con gráficos como para poder obtener unas conclusiones válidas.

Además, la aparición de la máquina Glulx como una evolución de la máquina Z, con posibilidades avanzadas, una arquitectura más sólida y un largo etcétera, da la posibilidad desde el estándar de crear juegos de este tipo.

El presente tutorial tiene el objetivo de proponer una forma sencilla y básica de añadir gráficos al juego, aportando al lector una solución guiada paso a paso que podrá tomar como los cimientos de su propia aventura con gráficos.

De esta forma, se espera poner al alcance de todos los escritores de aventuras los mecanismos suficientes para producir este tipo de aventuras, y una vez puestos los medios .... que cada uno decida.

Ejercicios

1. Crea un gráfico para el cartel podrido en el saliente de roca.
2. Crea un gráfico de localidad nuevo que aparezca cuando ganas (es decir, crea una localidad final nueva con su propio gráfico, o símplemente visualízalo).
3. Crea un gráfico que aparecerá como presentación antes de comenzar la aventura (consúltese el código *sixdemo.inf*).

Referencias

[1] Iznoguz puede encontrarse en uno de los apartados principales de la web de Zak.
[2] La página de Zak es http://www.geocities.com/~spinf
[3] A mí el deseo de incluir gráficos en uno de mis juegos me llevó a una frustrante búsqueda donde cada elemento encontrado (en los sitios más dispares) parecía requerir la presencia de otros elementos más, y así sucesivamente.
[4] Por otra parte, a los usuarios de Linux se les asume una mayor experiencia en el uso de sus sistemas operativos, por lo que necesitarán menos ayuda. Probablemente tengan que compilar la mayoría de las herramientas, si bien encontrarán los enlaces a los código fuente multi-plataforma en el mismo sitio que los enlaces a los ejecutables win32.
[5] Lo encontrarás en http://usuarios.lycos.es/elarquero/
[6] baltasarq@yahoo.es
[7] Los he hecho yo mismo, así que por favor, que nadie espere obras de arte.
[8] Espero que nadie piense que tengo algo contra Zak.