Memoria GEOMETRY

- 21715 - Estructura de Computadors II -

Escola Politècnica Superior, 2024.
A. Burguera Burguera

Blanca Atienzar Martínez,
45185593T,
blanca.atienzar1@estudiant.uib.cat

Hai Zi Bibiloni Trobat,
43231443E,
hai-zi.bibiloni1@estudiant.uib.cat

1. Introducción y juego propuesto

Para este proyecto hemos recreado el popular *Geometry Dash* de forma simplificada, es un videojuego diseñado principalmente para jugar en dispositivos móviles. Este juego fue elegido, ya que ambas participantes del grupo jugaban a este juego en la adolescencia.

Consiste en un usuario (cuadrado) que a través de saltos, debe esquivar los objetos que pasan por pantalla de forma deslizante de derecha a izquierda.

El juego es sencillo y asequible, para jugar solo se necesita la barra de espacio, esta tecla hace que el usuario (cuadrado) salte. Entonces, el usuario se encuentra con dos objetos distintos, están los triángulos (triángulos) que si se golpea por cualquier punto de tal se reinicia el juego, y el contador de intentos se incrementa, por otro lado, están las plataformas (objetos rectangulares) que si los golpeas horizontalmente hacen la función ya comentada, pero si saltas encima te deslizas por encima de ella.

Finalmente, el juego acaba cuando se ha recorrido todo el nivel.

2. Método de solución

El proyecto cuenta con un total de 25 clases, 7 de los cuales se encuentran en una carpeta llamada LIB, y son para un uso genérico (“librerías”), por otro lado tendremos 4 clases que definirán las constantes, constantes relativas y las de sistema; el resto de clases tendrán una estructura similar y la mayoría contarán con las subrutinas XUPD y XPLOT, entre ellas. Por otro lado tendremos la carpeta DATA la cual tiene el archivo .bin que contiene el mapdata (MAPDATA.bin) y una clase con los vectores que contienen las imágenes (PICTURE). De forma detallada las clases funcionan así:

- MAIN: el fichero main es el que incluye todos los otros ficheros que se utilizan en el juego, y es el fichero que permite jugar al juego.
- CONST: fichero que incluye las constantes del juego, como la velocidad del usuario o triángulo, entre otros.
- SYSCONST: fichero que incluye las variables que definen el tamaño de la pantalla y que teclas se utilizan en el juego (variables de sistema).
- SYSTEM: fichero que incluye el código que inicializa la pantalla (tamaño) y el teclado para jugar.
- VARS: fichero que incluye las variables del juego, como USUPOSX o PLATPOSX, entre otros.

- SYSVARS: fichero que incluye las variables relativas al teclado entre otras.
- RECTA: fichero que incluye la subrutina RECPLOT, la cual se encarga del dibujo de imágenes en cada azulejo perteneciente del marco del nivel.
- TRIANGULO: fichero que incluye las subrutinas TRIUPD y TRIPLOT; la primera subrutina mencionada comprueba la colisión del usuario sobre este haciendo comparaciones de los diversos puntos geométricos de ambos objetos (triangulo y usuario), si hubiera colisión el juego se reinicia, con las subrutinas de inicializacion XINIT.
- PLATAFORMA: fichero que incluye las subrutinas PLATINIT, PLATUPD y PLATPLOT; con las mismas funcionalidades que el anterior.
- MAP: fichero que incluye las subrutinas MAPINIT, MAPUPD, MAPPLOT y MAPOBJETOS; la subrutina MAPUPD va actualizando la posicion del mapa para ir desplazando a la izquierda, por otro lado la subrutina MAPOBJETOS es la encargada de segun el numero que indique el mapa crear un objeto o simplemente pintar el fondo.
- USUARIO: fichero que incluye las subrutinas USUINIT, USUUPD, USUPLOT y USUFLOOR; la subrutina USUUPD, por un lado, actualiza la posicion de la coordenada x del usuario para moverlo horizontalmente al principio y final del juego, y por el otro actualiza los saltos que son dirigidos por la tecla de espacio del usuario, también llama a la subrutina USUFLOOR, donde su funcionalidad es asignarle un valor a la variable NWFLOOR, que se irá actualizando a medida que el usuario cambie de coordenada y, es decir, si el usuario salta a una plataforma NWFLOOR indicará que el usuario debe acabar el salto encima de la plataforma, una vez la plataforma llegue a su fin el usuario podrá cambiar de “suelo imaginario” y volver a la coordenada y principal.
- SCORE: fichero que incluye el código que muestra por pantalla dos mensajes: número de intentos y número de saltos.
- STATES: fichero que incluye el orden en que se deben ver las pantallas en los diferentes momentos, en nuestro caso es introducción, instrucciones, juego y ganar.
- INTRO: fichero que incluye el código que escribe por pantalla la página principal de *Geometry Dash*. Es la primera pantalla que se ve al dar “play” al juego, que incluye el nombre del juego y un mensaje que explica como se pueda empezar a jugar (pulsar la barra espaciadora).
- GAME: fichero que inicializa todos los objetos que forman parte del juego, por ejemplo el usuario (cuadrado).
- INSTRUCCIONES: fichero que incluye el código que escribe por pantalla las instrucciones de cómo se juega al *Geometry Dash*. Para poder ver la pantalla de instrucciones, es necesario esperar unos segundos en la pantalla principal del juego, y para poder jugar, solo hay que volver a esperar o pulsar la barra espaciadora y te llevará a la pantalla principal del juego.

- WIN: fichero que incluye el código que solo se mostrará en pantalla en el caso que se gane el juego. Posteriormente, se podrá volver a la página principal del juego desde la pantalla de WIN pulsando la tecla de espacio una vez.

En cuenta a los ficheros de la carpeta LIB:

- DMMCONST: fichero que incluye el código sobre las constantes de la memoria dinámica.
- DMMCODE: fichero que incluye el código sobre la gestión de la memoria dinámica.
- UTLCONST: fichero que incluye el código sobre la gestión del código útil.
- UTLCODE: fichero que incluye el código sobre las constantes útiles.
- UTLVARS: fichero que incluye el código sobre las variables útiles.
- DMMVARS: fichero que incluye el código sobre las variables de la memoria dinámica.

3. Dificultades

Durante la realización de la práctica nos encontramos con estas varias dificultades, pero sin embargo algunos con mayor grado que otras, o con mayor tiempo de enfoque en ella, pero sin embargo las más laboriosas fueron:

- Nuestra primera dificultad fue la creación de objetos, los cuales aún no estaban predeterminados por el mapa, es decir, tuvimos que programar su movimiento, y lo más trabajoso sin duda el tema de las colisiones. Para comenzar el proceso y entenderlo hicimos, paso a paso, primero que se pintara todo el movimiento, y finalmente, que se borrara el camino que hacía.

- Lo segundo más dificultoso fue introducir todo lo que ya habíamos creado al mapa, para que funcionase correctamente. Por otro lado también nos fue difícil el tema de guardar el mapa en un archivo .bin, ya que desconociamos totalmente este formato, y lo que debíamos hacer para implementarlo en el easy68k. Sin embargo, cabe destacar otros problemas minoritarios como: la lectura del nivel, se crearán de forma correcta los objetos y el desplazamiento horizontal de este.

- Funcionalidad objetos: cada objeto del juego tiene una funcionalidad, en el momento de asignar sus funcionalidades a través de la asignación del mapa. El objeto más difícil de implementar funcionalidad fue la plataforma.

Para concluir, una cuestión que nos ha dificultado en la elaboración, ha sido la falta de velocidad del juego a la hora de hacer las pruebas. Explicado de otra forma, a causa de nuestros dispositivos electrónicos (ordenadores), el programa cuando implementamos el mapa y consecutivamente las imágenes, comenzó a paralizarse y tener 1 FPS. Esto ha llevado a que realmente nunca sabremos si el programa funciona correctamente en la parte final que implica la pantalla “WIN”. En caso de que el juego en otro ordenador fuese demasiado rápido o lento, la velocidad es ajustable con la variable MAPSPEED en la clase CONST.

4. Características adicionales

Para la realización de la práctica se deben implementar mínimo tres características adicionales, por lo que para este juego (“geometry dash”) estas tareas se implementaron en el siguiente orden: la visualización de imágenes, el uso de ficheros y los frames per second:

- Visualización de imágenes: Para la ejecucion de las imagenes, creamos un programa en JAVA el cual leia pixel por pixel de una imagen .png y extraia su codigo de color en BGR. Una vez en ello creamos un vector con cada imagen de cada objeto i/o título (están en la clase DATA/PICTURE.X68), los cuales son llamados por cada clase de su respectivo objeto y son dibujados a través de sus subrutinas nombradas con el sufijo XPLOT. Debemos dejar claro que las imágenes y el programa fueron creados por nosotras.

- Uso de ficheros: Nuestro videojuego accede a un fichero externo para cargar el mapdata a un vector en el easy68K. Esto sucede en la clase MAP en su primera subrutina MAPINIT, donde inicializamos el mapa y por lo tanto cada vez que inicialicemos sucederá esto. Para poder dibujar por pantalla nuestro mapa del juego, se han extraído los datos de un fichero binario creado por nosotras a través de un programa en Python (imagenes.py) llamado “MAPDATA.bin”. Dentro del fichero se encuentran los objetos de nuestro juego codificados por 0 (el fondo del mapa), 1 (marco del juego) y 2 (triángulos). Para poder leer el fichero hemos utilizado las tareas 50, 51, 53 y 56 del TRAP #15.
  * El programa de python mencionado anteriormente, tiene como parametro un array formado por el mapdata y devuelve este mapdata introducido en un archivo .bin .

- Frames per second: Para el cálculo de “Frames per second” se ha tenido en cuenta que en "PLOT" nosotras solo realizamos el dibujo y el doble buffer. En cada iteración del bucle se pinta un frame por segundo del juego. Para eso hemos necesitado saber cuantas veces se repite el bucle por segundo. Así conseguimos que cada segundo se enseñe el número de frames que se han pintado cada segundo (FPS). Para poder conseguir los FPS primero de todo en el apartado de inicialización hemos inicializado los frames por segundo. Posteriormente en el apartado PLOT con la tarea 8 del TRAP #15 hemos actualizado los FPS cada segundo. Finalmente para mostrar estos FPS, utilizamos la subrutina SCOPLOT en la clase SCORE junto al marcador de intentos, también cabe destacar que el usuario puede cambiar el frame rate deseado a través de la constante FPSGOALS situada en la clase CONST.

Estas características se fueron implementando también según nuestro criterio de dificultad, es decir, nos fue más fácil la implementación de imagenes en .png para el título e objetos, hasta la implementación del frame per second que fue el más dificultoso, ya que al principio no supimos captar bien la idea.

5. Conclusion

En conclusión, esta práctica ha resultado más difícil de lo que esperábamos, ya que durante todo el proceso de realización nos íbamos encontrando dificultad tras dificultad.

Dividimos todo el trabajo en las posibles tareas que había que llevar a cabo, y empezamos a crear el juego, el problema llegaba cuando había que implementar las tareas en el Easy68k, dado que teníamos la idea de como poderlo hacer, pero no sabíamos cómo plasmarlo.

Además hemos tenido que añadir 3 características adicionales, ya que era un requisito para la recuperación de la práctica, las cuales algunas nos han resultado más fáciles de implementar que otras.

Finalmente, lo conseguimos, y el resultado es el trabajo presentado.