PRÁCTICA FINAL DE ESTRUCTURA DE COMPUTADORES II:

SPACE INVADERS (VERSIÓN SIMPLIFICADA)

Universitat de les Illes Balears: Escola Politècnica Superior

Grado de ingeniería informática

Profesor: Antonio Burguera Burguera

Adrián Ruiz Vidal

Khaoula Ikkene

INTRODUCCIÓN

Para esta práctica hemos desarrollado un juego en el ensamblador EASY68K. Se trata del juego clásico Space Invaders. Su dinámica es notablemente sencilla. La premisa consiste en eliminar a los 30 aliens antes de que alcancen la altura del tanque utilizando el menor número de disparos posibles. La única variable de dificultad radica en la velocidad, la cual se intensifica conforme se progresa a través de los niveles del juego, que comprenden del nivel 1 al nivel 3.

El juego presenta un escenario donde el jugador puede controlar una máquina de disparar (nosotros le hemos llamado tanque) que se puede mover horizontalmente y el objetivo es eliminar todos los aliens (que son las figuras que se van moviendo) lo más rápido posible, porque si alcanzan el tanque (basta que quede 1), el jugador pierde.

¿Cómo jugar? Las teclas necesarias para jugar son solo tres: key up para disparar, key left para mover a izquierda y key right para mover el tanque a la derecha.

Decidimos realizar este juego porque nos pareció un juego divertido además de conocido y que creíamos que podíamos ser capaces de hacerlo.

ESTRUCTURA

El código sigue una estructura definida y ordenada. En la mayoría de las clases del juego, excluyendo las macros y el programa principal, identificamos las siguientes subrutinas fundamentales:

XINIT: Esta subrutina despliega la función de inicializar las variables específicas de cada clase. Por ejemplo, en la de ALIEN esta subrutina establece las posiciones iniciales de dicho carácter, así como su velocidad.

XUPD: En esta subrutina se lleva a cabo la actualización de las variables unidas a la clase, así como aquellas vinculadas al buffer.

XPLOT: Esta subrutina asume la responsabilidad de la representación gráfica de los carácteres pertenecientes a cada clase. Su función principal radica en la generación visual de los elementos del juego.

Dentro de la carpeta del código tenemos varias carpetas, cada uno con su nombre identificador:

- FIC: donde se guarda el fichero que utilizamos para realizar la demo con sus correspondientes números para realizar las acciones.
- SND: contiene los correspondientes sonidos en formato .wav. Hemos incorporado efectos sonoros a las explosiones de los Aliens, así como una canción que se activa al inicio de la INTRO (tuvimos que bajarle la frecuencia de muestreo). Además, introdujimos dos sonidos distintos para representar la victoria y la derrota del jugador.

- LIB: dividida en clases de código y otras para guardar variables y constantes. Es propia del profesor y la hemos empleado para crear una clase agente para el disparo e imprimir texto por pantalla.

Pasando a las clases, tenemos clases de constantes (CONST), que nos posibilitan modificar estas constantes con facilidad, y otras de variables (VARS), que son valores que se van modificando a lo largo del juego y en ellas, encontramos velocidades, marcadores y posiciones.

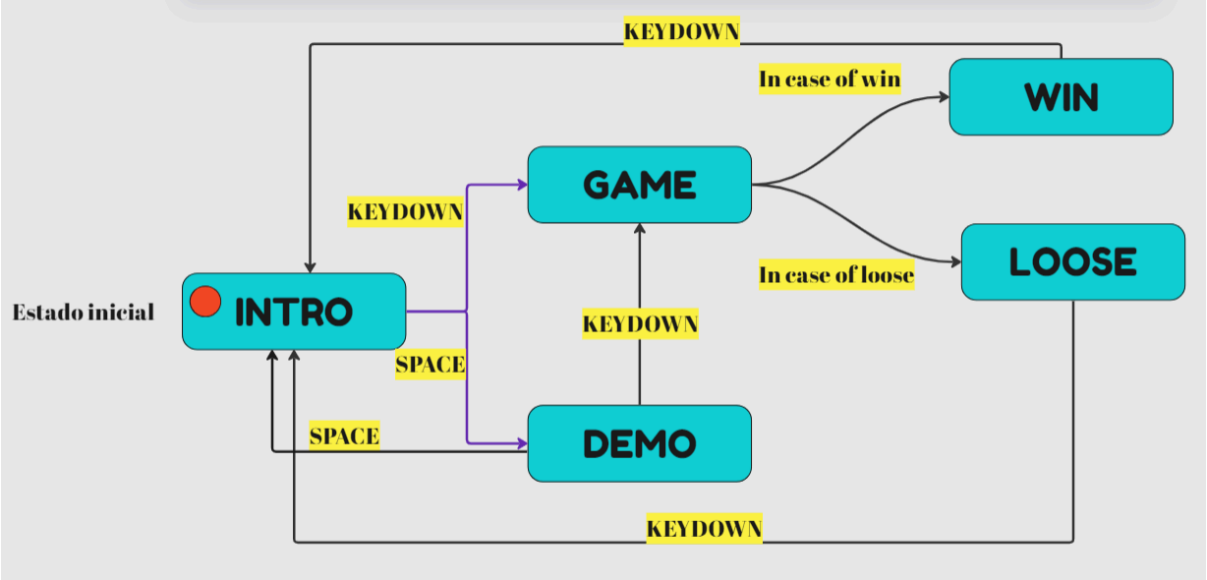
La clase MAIN, es la principal que ejecuta el proyecto. Contiene todos los includes del resto de clases, la inicialización, el update y el plot del estado actual, la inicialización del sistema, algunos traps y la sincronización, que permite que el juego vaya fluido.

En el ámbito de la eficiencia del código, implementamos una clase denominada "MACROS" con el propósito de abordar tareas recurrentes presentes en diversas secciones del juego. Las utilizamos cuando el código se emplea en varias subrutinas y se cambia mínimo un parámetro de entrada.

Hablando de eficiencia y limpieza de código, hemos realizado una serie de subrutinas que se encuentran a lo largo de todas las clases con el objetivo de reducir código redundante. Estas se han realizado cuando teníamos exactamente el mismo código en más de una subrutina.

La clase SYSTEM es de gran importancia, porque en ella están instalados los traps e interrupciones que usamos. Se implementa también la inicialización de la pantalla, la técnica del doble buffer y el sonido, entre otras cosas.

Hay que decir que hemos realizado una estructura de juego con estados. Primero de todo, definimos el diagrama y el paso entre un estado y otro.



La clase STATES es la encargada de manejar la transición de estados mostrada en el diagrama.

Al ejecutar el proyecto, encontramos una presentación (INTRO), que contiene las instrucciones de juego, el objetivo y la posible transición al siguiente estado (DEMO o GAME).

Cabe aclarar que DEMO es únicamente una muestra de la dinámica del juego, por lo tanto, no tiene condición de derrota ni de victoria. En relación con esta clase, podemos mencionar la clase FILE, que se encarga de leer un fichero de texto y a raíz de cada valor que tiene el fichero, la DEMO realiza una de las siguientes acciones: mover el tanque a la izquierda, moverlo a la derecha o disparar.

Para los estados de WIN/GOVER se imprime por pantalla la puntuación final que ha obtenido el jugador, el número de disparos realizados y se reproduce el sonido correspondiente a cada caso. Llegando a uno de estos dos estados, el jugador puede siempre volver a la INTRO y jugar todas las veces que quiera sin tener que salirse.

La clase SCORE contiene una serie de marcadores: puntuación, por cada alien eliminado se obtiene una puntuación de 5 puntos (se puede cambiar y mostrar correctamente la puntuación final); disparos, que cuenta el número de disparos en ese momento; nivel, que va aumentando según los aliens van llegando al límite establecido para incrementar el nivel de dificultad.

Nos queda por definir las clases que definen el juego:

- TANK es una clase que permite al jugador moverse de lado a lado y disparar. Conlleva un mapa de bits que muestra en pantalla la figura del tanque.
- ALIEN es una clase que contiene el movimiento de estos en pantalla. Es un mapa de bits 3x10, ya que son 30 aliens, y dentro de cada bloque tiene la figura del alien, que también es un bitmap (10x10). El movimiento de los aliens va de lado a lado y cada vez que llega a un extremo (dejamos un poco de margen), baja un poco. Si el disparo toca el alien, este último desaparece.
- SHOT representa el disparo del tanque. Es un agente, donde POSX se representa como (A0) y POSY como 2(A0). Y cada vez que se teclea key up se crea un disparo nuevo, dándole al jugador la oportunidad de tener disparos infinitos. Si un disparo colisiona con el alien, el disparo se elimina, así como si toca el techo de la pantalla.

DIFICULTADES

Sobre los desafíos afrontados, la faceta que presentó mayores obstáculos fue la identificación de colisiones del disparo con el alien, pero gracias a la ayuda del profesor la hemos podido superar.

Asimismo, en las fases iniciales del desarrollo del juego, experimentamos cierta desorientación, y que nos parecía un poco complicado el uso de los sprites, pero al final resultó que era solo miedo de que las cosas no funcionaran más adelante.

En ciertos momentos, también necesitamos la explicación del profesor para saber como podíamos plantear alguna tarea en específico.

No podemos evitar hablar de los típicos fallos, que llevan una gran cantidad de tiempo y al final solo resulta ser una letra mal escrita (este era más fácil de solucionar), una variable mal restaurada, un salto que debíamos hacer y no hacíamos, etc. A esto, hay que decir que los breakpoints y el análisis detallado del código paso a paso nos han ayudado mucho a resolver este tipo de problemas.

TAREAS ADICIONALES

Como la práctica es bastante adictiva, experimentamos un creciente estímulo que nos impulsaba a incorporar cada vez más características. Hay que decir que en un principio intentamos incluir el uso de ratón y los fps, pero de este último no encontramos suficiente información para realizar el cálculo real de los fps y entonces decidimos irnos por la funcionalidad de demo combinado con fichero.

Hemos logrado desarrollar la funcionalidad del *Attract Mode*, donde se muestra la dinámica del juego de manera automática sin necesidad de ningún teclado para moverse o disparar, ya que hemos hecho uso de una funcionalidad más como es el uso de un fichero (fichero.txt) que contiene los movimientos del tanque.

También creemos que hemos hecho la tarea de visualización de imagen dado que los carácteres de nuestro juego (quitando el disparo que es un rectángulo) son sprites.

CONCLUSIONES

Llegando a esta sección creemos que la práctica, como se conoce su objetivo, nos resultó sumamente útil para profundizar en los conceptos prácticos vistos durante el curso como el acceso a periférico, la creación de traps y interrupciones, el mapeo de memoria, el uso de subrutinas y macros entre otras cosas.

Ha sido una práctica que hemos estado semana a semana pendiente y cada semana realizamos un pequeño objetivo. Trabajar en grupo nos ha ayudado a planificarnos mejor y que si algo se le daba peor a uno, el otro lo podía resolver. Ha tenido tanto trabajo individual como trabajo en grupo, ya que solíamos quedar para avanzar la práctica.

Con todo esto, hemos podido sacarlo adelante y realizar un juego que refleja el trabajo que hemos necesitado para hacerlo posible y un código optimizado.

RECURSOS USADOS

SONIDOS
https://www.classicgaming.cc/classics/space-invaders/sounds
https://mixkit.co/free-sound-effects/space-shooter/
OTROS
Firing bullets
M68000 Microprocessors User's Manual