Eugenio Doñaque – 49614987P

Nadal Llabrés Belmar – 41616427C

Introducción

El siguiente documento pretende explicar en detalle el desarrollo de la práctica final de la asignatura Estructura de Computadores II de la carrera Ingeniería informática de la Universidad de las Islas Baleares. La misma consiste en el desarrollo de un videojuego en el lenguaje ensamblador del microprocesador Motorola 68000, utilizando el *software* EASy68K.

Siguiendo las pautas y consejos recibidos a lo largo del cuatrimestre, se ha llevado a cabo el desarrollo del videojuego *Flappy Bird*.

Estructura del código

El profesorado he proporcionado unos archivos básicos para seguir una estructura ordenada y coherente. Estos archivos son:

- MAIN.X68
- SYSCONST.X68
- SYSTEM.X68
- VAR.X68

MAIN.X68

Contiene el bucle principal del juego. En el primer apartado se realiza la llamada a las subrutinas de inicialización, seguidamente a las subrutinas de actualización y, para terminar, a las subrutinas de dibujado.

SYSCONST.X68

Contiene constantes relacionadas con el sistema, el tratamiento de las teclas del teclado y el tamaño de la ventana de juego.

SYSTEM.X68

Implementa rutinas del sistema que permiten realizar lecturas a través del teclado, inicializar los *buffers* de pantalla, establecer la resolución de la pantalla y gestionar la memoria.

SYSVAR.X68

Contiene las variables utilizadas por el sistema para su funcionamiento. Entre ellas, variables utilizadas para gestionar el teclado, la pantalla y la memoria.

VAR.X68

Incluye las variables del videojuego relacionadas con el jugador, el fondo o con los objetos que aparecen en pantalla.

Además de la estructura básica proporcionada, se agregó archivos cuya funcionalidad e implementación en el código hace de engranajes entre las diferentes partes del programa. Con esto, se logra dar un poco de abstracción y permite tener un código fuente más fácil de entender y modificar.

AGENTLST.X68 Y PROPLST.X68

Siguiendo la estructura explicada en clase, se añadió al código dos archivos que gestionan los agentes y utilitarios del videojuego. Se denomina agente a todo ente que forma parte activa en las acciones del juego -en este caso, las tuberías-, y se le llama utilitario o '*prop*' a todo aquello

que forma parte del videojuego, pero su existencia no determina nada en el desarrollo del mismo -Ej. Las nubes de fondo o los detalles en la grama no determinan si el jugador pierde o gana al chocar con ellas. Solo están por una necesidad estética en este caso particular.

La razón para tener ambos separados, es para evitar que en unos casos un *prop* se pinte por encima de un *agent*, causando efectos indeseables como una nube que sobrevuela algunas tuberías.

Estos archivos existen para automatizar el acceso a memoria de cada agente/*prop* y los consiguientes llamados a sus rutinas de *update* y *plot.*

SPAWNER.X68 Y BACKGROUND.X68

Estos archivos existen para localizar una funcionalidad específica necesaria para el juego: la generación automática de agentes y utilitarios -y en el caso de BACKGROUND.X68, añadir también el pintado de la imagen de fondo-.

STATES.X68

Este archivo es el que se encarga de gestionar las acciones a realizar en cada iteración del bucle del juego entre el pintado de nuevas pantallas o *frames*. En el se encuentra el código que detecta cuando ha habido un cambio en el estado de juego y automatiza la ejecución de las diferentes subrutinas específicas para cada estado. Aquí se encuentra el código inicialización, actualización y dibujado de cada *frame* del juego, y, además, este incluye los archivos INTRO_SCREEN.X68, PAUSE_SCREEN.X68, INSTR_SCREEN.X68, Y GOVR_SCREEN.X68, donde se gestiona las rutinas de inicialización, actualización y dibujado de cada una de las pantallas del juego -introducción, pausa, instrucciones y *game over*.

A parte de los archivos mencionados anteriormente, se han creado archivos adicionales para desarrollar el fondo, el jugador, gestionar el sonido, las tuberías y las pantallas de inicio, pausa, y *game over*.

Dificultades encontradas durante el desarrollo

Al empezar a desarrollar el videojuego, una de las principales dificultades fue entender como engranaban cada una de las partes de la estructura y el orden en el cual se llamaban las subrutinas del sistema.

Una vez comprendida esa parte, otra dificultad fue programar el salto del pájaro para que fuese suave y progresivo, de esta forma se da una experiencia mucho más realista para el jugador.

En este apartado también se podría mencionar el detector de colisiones con las tuberías, ya que hubo que plantarse delante del papel para pensar una buena estrategia para hallar las colisiones.

Añadidos principales

Ficheros:

- PLAYER.X68
- PIPES.X68
- CLOUD.X68
- GSSDETAIL.X68
- SOUND.X68
- UTIL.X68

Los cuatro primeros archivos contienen el código referente al jugador y los agentes y utilitarios. Todos incluyen las mismas funciones para cada elemento: *init, update, plot.*

Dentro de PLAYER y PIPES en conjunto está implementada la detección de colisiones, donde dentro de la rutina de *update* de pipes, se verifica que el jugador haya o no colisionado con alguna tubería, y se pasa por una variable el resultado para ser recuperado en la rutina de *update* de *player* para realizar la acción correspondiente y pasar al estado de *game over*.

CLOUD y GSSDETAILS solo se encargan de generar y mantener las nubes y detalles en la grama mientras existan (subrutina *propupd* se encarga de liberar el espacio en memoria ocupado por ellos cuando estos salen de la pantalla).

El formato de cualquiera de estos archivos sigue este patrón de ejemplo:

PLAYER.X68

Este fichero tiene como función dibujar el jugador, actualizar sus coordenadas y mostrar la puntuación.

Consta de tres partes principales:

Inicialización

Se dan valores iniciales a cada una de las variables.

Actualización

Modificación de los valores de las variables según la acción que se tenga que llevar a cabo. Por ejemplo, actualizar la posición del jugador.

Dibujado

Se pinta el personaje en la pantalla según la configuración en ese instante de tiempo.

SOUND.X68

Este archivo se encarga de cargar todos los sonidos a ser reproducidos durante el juego en memoria de una manera ordenada para que las constantes de sonido sean consistentes con el número de referencia asignado a cada uno. Esto es la función SNDINIT de este archivo.

Como subrutina de actualización, el juego está programado para iniciar la música de fondo 64 ciclos después de iniciar o continuar una partida. De esta manera, se da una pausa para que termine de reproducirse el sonido elegido al presionar SPACE dentro de uno de los menús del juego.

El sonido de fondo ha sido una mezcla entre una sección de la canción T2000 (productores: Mek & Pastemann), y dos sonidos de pájaros cantando en el fondo, para dar un poco de vida a la monotonía de un *soundtrack* típico de videojuego.

El resto de sonidos han sido obtenidos en internet a través de las múltiples páginas webs que ofrecen simples de sonidos para desarrolladores de videojuegos, etc. Y han sido acortadas y modificadas para optimizar su efecto dentro del programa.

UTIL.X68

Este archivo contiene macros y subrutinas de utilidades implementadas a lo largo del código del juego. Ello incluye macros para cambiar el color de relleno y dibujado del procesador, cambiar la fuente de impresión de texto por pantalla y cambiar la localización del cursor para escribir en posiciones deseadas.

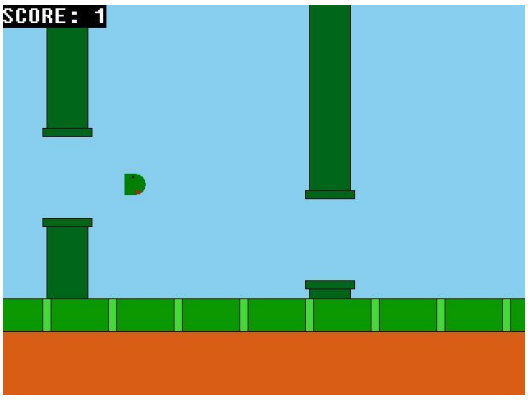
Además, en este archivo se encuentra la subrutina de detección de colisiones -muy importante para el juego-, una subrutina que devuelve un byte pseudo-aleatorio utilizado sobretodo en la inicialización de tuberías y de nubes para la coordenada Y.

En esta también están las subrutinas correspondientes a reproducción y detención de sonido (en el caso del sonido de *background*) y la subrutina encargada de gestionar el acceso a ficheros para guardar en memoria secundaria el mejor resultado de partidas anteriores.

Cómo jugar

Al arrancar el juego aparece una pantalla de inicio para dar la bienvenida el jugador y poder seleccionar la dificultad, ver las instrucciones o empezar la partida.

Para cambiar las opciones se debe hacer uso de las teclas de dirección (↑ ↓ → ←). Para seleccionar una opción hay que presionar SPACE.



Al iniciar el juego habrá que evitar tocar la parte superior e inferior de la pantalla y las tuberías si se quiere ganar. Para subir el pájaro hay que utilizar la tecla de dirección UP (↑).

En la parte superior izquierda aparece el número de tuberías atravesadas.

Se dispone de un menú de pausa donde hay la opción de reiniciar el juego, salir o activar/desactivar el sonido.

Si se produce alguna colisión, el juego habrá terminado y mostrará la pantalla de *game over.*

Conclusión

Desarrollar un videojuego en lenguaje ensamblador ha sido todo un reto por parte de los estudiantes. Se ha aprendido a manejar con fluidez bastantes tareas del TRAP #15 relacionadas con la entrada y salida de datos.

Trabajar con una estructura de inicialización, actualización y dibujado ha servido para ver con más claridad cómo funciona internamente el juego, programarlo siguiendo un orden coherente y detectar posibles errores con mayor facilidad.

Finalmente, hay que decir que ha sido bastante interesante ya que parecía que con el simulador EASy68K sólo se podrían hacer operaciones con registros y memoria. Se ha podido ver que no es así ya que se pueden programar videojuegos con cierta complejidad.