2014-2015

MEMORIA PROYECTO NDS

ARQUITECTURA DE COMPUTADORES – ESTRUCTURA DE COMPUTADORES

GRUPO G.10: JON ARZELUS, CARMEN NUÑO - FISS

Introducción

Con esta memoria vamos a tratar de dar una visión general del desarrollo del proyecto de EC sobre el funcionamiento de la programación por interrupciones y el funcionamiento de la NDS.

El proyecto trata sobre crear un juego muy sencillo para aprender los principios básicos del funcionamiento de las interrupciones de un programa informático, al igual que el funcionamiento de dispositivos de entrada y salida.

Contenido

PROYECTO

Descripción general del programa(cambiar fotos)

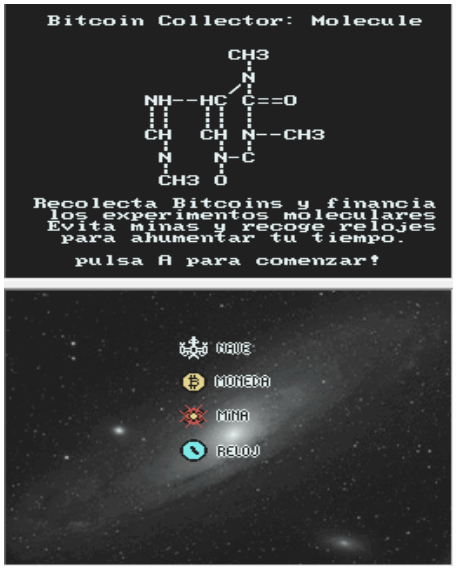

Figura A.1

En la figura A.1 se muestra la pantalla del **estado INICIO**. En ella podemos observar los *sprites* que nos van a aparecer durante la partida junto a su nombre (de arriba abajo: Nave, Moneda, Mina y Reloj).
En la parte superior de la pantalla se muestra el nombre que se ha asignado al juego (Bitcoin Collector: Molecule®), una molécula construida a modo de arte ASCII y una breve descripción de lo que se trata el mismo. Cuando se pulse la *tecla A* se dará comienzo al juego y mostrará la pantalla del **estado PARTIDA**.

Figura A.2

En la figura A.2 se muestra la pantalla del **estado PARTIDA**. En la pantalla superior se nos muestran varios datos: la última tecla que se ha pulsado, las vidas restantes, las monedas recogidas, la posición de la nave y información sobre como terminar inmediatamente la partida.
En la pantalla inferior se muestran los sprites (nave, moneda, mina y reloj), cuya posición se va actualizando periódicamente (en el caso de las minas, monedas y relojes) y tras pulsar una tecla (en caso de la nave).
Si el tiempo se agota, las vidas llegan a cero o se pulsa la *tecla B* la partida finalizará y se mostrará la pantalla del **estado FIN**.

Figura A.3

En la figura A.3 se muestra la pantalla del **estado FIN**. En la pantalla superior se nos muestran varios datos: Las monedas recogidas, el tiempo que ha durado la partida (variable según el número de relojes que se han recogido) e instrucciones de como continuar con el programa.
En la pantalla inferior se muestra el fondo de pantalla, sin ningún sprite.
Si el usuario pulsa la *tecla A* se cambia al **estado PARTIDA**, y si pulsa la *tecla B* se cambia al **estado INICIO**.

Autómata y descripción del funcionamiento

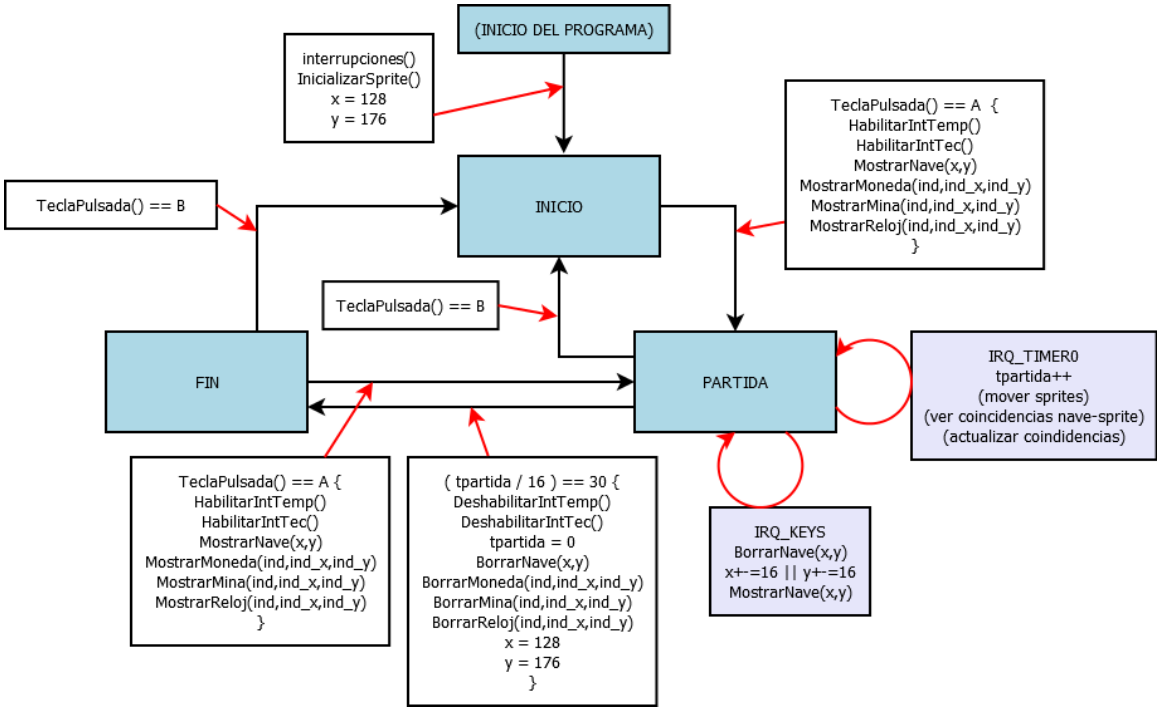

Figura B.1 (Autómata y descripción de cambios de estado)

El funcionamiento del juego según su estado es el siguiente:

- Estado INICIO: es el estado en el que se inicia el juego. Nada mas ejecutar el programa se asignan valores iniciales a las variables globales. Una vez en el estado se espera a que se genere una interrupción a causa de la encuesta de la *tecla A*. Cuando esto sucede se cambia el **estado a PARTIDA**, se comienzan a mostrar sprites en la pantalla y se activan las interrupciones del teclado y del temporizador.
- Estado PARTIDA: Es el estado de la partida, junto a sus rutinas de atención a las interrupciones. La nave se mueve en pantalla y si coincide con un sprite se actualizan las variables pertinentes. Si se pulsa la *tecla B* se sale al menu principal, cambiando al **estado INICIO**, y si el *temporizador* llega a 0 se sale de la partida a la pantalla de despedida, en el **estado FIN**.
- Estado FIN: En este estado se le da al jugador la posibilidad de volver a jugar o salir a la pantalla principal. Si el usuario pulsa la *tecla A* se cambia al **estado PARTIDA**, y si por lo contrario pulsa la *tecla B* se cambia al **estado INICIO**. Cuando se inicia otra partida se vuelven a inicializar las variables pertinentes.

Funciones del código del juego

Ficheros teclado.c y teclado.h	
Nombre de la función	Descripción del funcionamiento
`DetectarTecla()`	Detecta si se ha pulsado una tecla. Devuelve 1 si se ha pulsado alguna tecla, 0 en caso contrario
`TeclaPulsada()`	Identifica la tecla pulsada devolviento el valor de la misma (A, B, SELECT, START…)
`HabilitarIntTec()`	Habilita las interrupciones del teclado poniendo los valores pertinentes en los registros IE y IME
`DeshabilitarIntTec()`	Deshabilita las interrupciones del teclado poniendo los valores pertinentes en los registros IE y IME
`IntTec()`	Rutina de atención al teclado
Ficheros temporizadores.c y temporizadores.h	
Nombre de la función	Descripción del funcionamiento
`HabilitarIntTemp()`	Habilita las interrupciones del temporizador
`DeshabilitarIntTemp()`	Deshabilita las interrupciones del temporizador
`InicializarSprite()`	Asigna valores iniciales a los *sprites* en pantalla
`IntTemp()`	Rutina de atención al temporizador
Ficheros sprites.c y sprites.h	
Nombre de la función	Descripción del funcionamiento
`initSpriteMem()`	(predefinido) Inicializa la memoria de prites
`establecerPaletaPrincipal()`	(predefinido) Establece los colores para usar en los sprites
`establecerPaletaSecundaria()`	
`guardarSpritesEnMemoria()`	(predefinido) Guarda los sprites en memoria para su uso
`MostrarMoneda(int indice, int x, int y)`	(predefinido) Muestra-Borra una moneda en la posicion XY de la pantalla y le asigna el índice de sprite pertinente
`BorrarMoneda(int indice, int x, int y)`	
`MostrarReloj(int indice, int x, int y)`	Muestra-Borra un reloj en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`BorrarReloj(int indice, int x, int y)`	
`MostrarMina(int indice, int x, int y)`	(predefinido) Muestra-Borra una mina en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`BorrarMina(int indice, int x, int y)`	
`Mostrarmina1(int indice, int x, int y)`	Muestra texto en forma de sprite en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`Mostrarmina2(int indice, int x, int y)`	
`Borrarmina1(int indice, int x, int y)`	Borra texto en forma de sprite en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`Borrarmina2(int indice, int x, int y)`	
`Mostrarmoneda1(int indice, int x, int y)`	Muestra texto en forma de sprite en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`Mostrarmoneda2(int indice, int x, int y)`	
`Borrarmoneda1(int indice, int x, int y)`	Borra texto en forma de sprite en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`Borrarmoneda2(int indice, int x, int y)`	
`Mostrarnave1(int indice, int x, int y)`	Muestra texto en forma de sprite en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`Mostrarnave2(int indice, int x, int y)`	
`Borrarnave1(int indice, int x, int y)`	Borra texto en forma de sprite en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`Borrarnave2(int indice, int x, int y)`	
`Mostrarreloj1(int indice, int x, int y)`	Muestra texto en forma de sprite en la posicion XY de la pantalla y le asigna el índice de sprite *indice*
`Mostrarreloj2(int indice, int x, int y)`	
`Borrarreloj1(int indice, int x, int y)`	

`Borrarreloj2(int indice, int x, int y)`	Borra texto en forma de sprite en la posicion XY de la pantalla y le asigna le asigna el índice de sprite *indice*
`MostrarNave (int x, int y)`	(predefinido) Muestra-Borra una nave en la posicion XY de la pantalla y le asigna el índice de sprite 127
`BorrarNave (int x, int y)`	
Ficheros rutserv.c y rutserv.h	
Nombre de la función	Descripción del funcionamiento
`interrupciones()`	Función para configurar los periféricos y el interrupt-dispatcher.
Fichero main.c	
Nombre de la función	Descripción del funcionamiento
`main()`	Función principal del programa, donde se realiza encuesta constante del *estado actual* y la *tecla A*

Descripción detallada del código

El código fuente tiene funciones con sentencias if bastante complicadas y código extenso; a continuación se muestra una pequeña explicación de dichas funciones.

sprites.c

Se han definido más colores en la paleta principal, al igual que varios nuevos Sprites. Los sprites Borrarreloj1, Mostrarnave1… Son simplemente imágenes de texto para poder escribir en la pantalla secundaria.

temporizadores.c

`if ((ind_x[i]<=(x+8)&&(ind_x[i]>=(x-8)))&&(ind_y[i]<=(y+8)&&(ind_y[i]>=(y-8))))`, con esta sentencia condicional se ve si la posición de la nave está a 8 píxeles o menos de distancia de cualquier sprite.

`if (ind[i] < 51) { … } else if (ind[i] < 117) { … } else { … }`, se define la probabilidad de que aparezca una mina, moneda o reloj. Los índices 0..50 están reservados para monedas; los índices 51..116 están reservados para minas; los índices 117..126 están reservados para relojes. Así se evita que aparezcan demasiadas monedas o demasiados relojes. Si se cambian estas proporciones se podría modificar la dificultad del juego.

`if ((tpartida / 16) == 30)`, *tpartida* es una variable que se incrementa en 16 unidades cada segundo, para un fluido movimiento de los sprites. Es por ello que para ver si ha llegado a 30 segundos se divide la variable por 16. Cuando el usuario recoge un reloj se le resta 64 a *tpartida*, o sea se, se le dan 4 segundos más al jugador para finalizar la partida.

OBSERVACIONES DEL PROYECTO

Varios sobre el proyecto

Hemos tenido varios problemas en la realización del proyecto. El principal problema ha sido que realizar un trabao como este es completamente nuevo para nosotros, y por ello hemos tardado más de lo esperado en completarlo. Por lo demás el proyecto se ha realizado de una manera bastante fluida, una vez habiendo establecido el rumbo que este iba a tomar.

Estimaciones sobre el proyecto

A continuación se muestra una tabla con las estimaciones de tiempo para cada tarea realizada en el proyecto:

TAREA	TIEMPO ESTIMADO (HORAS)
ESTUDIO DEL PROYECTO (VER LO QUE SE PIDE ETC.)	3
ORGANIZACIÓN DEL GRUPO	2
PLANIFICACIÓN DEL FUNCIONAMIENTO (AUTÓMATA, ESTADOS, PANTALLAS, SPRITES…)	6
ESCRITURA DEL CÓDIGO (PRINCIPAL)	52
MEJORAS DEL CÓDIGO (NUEVOS SPRITES, REDUCIR CÓDIGO, SIMPLIFICAR LECTURA ETC.)	23
AJUSTES FINALES DEL CÓDIGO (CORRECCIONES)	5
ESCRITURA DE LA MEMORIA DEL PROYECTO	14
TIEMPO TOTAL	105

ANEXO

Contenidos del anexo

A continuación se incluyen toda la documentación complementaria del proyecto:

- Carpeta con todo el código fuente, que contiene los fondos y archivos usados para la correcta compilación del juego. Listado de archivos:
  - Nave/ : Makefile, Nave.arm9, Nave.elf
  - Nave/build : Varios archivos de compilación (no se usan)
  - Nave/gfx : Fondo.grit, fondo.png
  - Nave/include : defines.h, fondos.h, graficos.h, rutservs.h, sprites.h, teclado.h, temporizadores.h
  - Nave/source : defines.c, fondos.c, graficos.c, rutservs.c, sprites.c, teclado.c, temporizadores.c
- Ejecutable en formato *.nds, para el uso del juego : Nave.nds
- Programa para la ejecución del archivo *.nds : NO_GBA.EXE

FACULTAD DE INFORMATICA DE SAN SEBASTIAN UNIVERSIDAD DEL PAIS VASCO