PROYECTO. DESARROLLO DE UNA APLICACIÓN DE E/S PARA LA NINTENDO DS

1 Introducción

El proyecto consiste en implementar un juego sencillo para una consola **Nintendo DS**. El objetivo del mismo es manejar los conceptos estudiados en el Tema 3 de Entrada/Salida de la asignatura Estructura de Computadores, por lo que el juego debe:

- Llevar a cabo el control de sus periféricos mediante distintos tipos de sincronización: por encuesta y por interrupción.
- Hacer uso, al menos, del teclado y de la pantalla táctil.
- Realizar algún tipo de control de tiempo con temporizadores.

Todo ello con el uso mínimo de librerías de alto nivel, concretamente en lenguaje **C**. Como el sistema gráfico de la consola es bastante complicado por lo que su definición ya está implementada en la plantilla del proyecto.

El desarrollo de este proyecto tiene un peso del **20%** de la calificación de la asignatura.

El proyecto se realiza en **grupo de tres estudiantes**. Además de las competencias específicas del tema de Tema 3 de Entrada/Salida, en este proyecto se van a valorar dos competencias transversales, (a) la capacidad de trabajo en equipo y (b) la capacidad para comunicar los resultados de forma escrita.

2 Planificación prevista del proyecto

La dedicación prevista para el desarrollo del proyecto es de **12 horas presenciales** (6 sesiones de 1:30 horas y una sesión, en la semana de horario agrupado, de 3 horas) y otras **12 horas no presenciales** (el 20% de la asignatura).

La siguiente tabla muestra la planificación del proyecto. Las clases dedicadas al desarrollo del proyecto son las destacadas en azul.

(*)	***MARTES*** ***9:00 – 10:30***	***MIÉRCOLES (desdobles)*** ***9:00 – 10:30 / 10:45 – 12:15***	***JUEVES*** ***12:30-14:00***
9	***24 de marzo*** ***Proyecto E/S****: Plantear el proyecto. Autómata.*	***25 de marzo*** ***Proyecto E/S****: Teclado en NDS. Funciones asociadas.*	***26 de marzo***
10	***31 de marzo***	***1 de abril*** ***Proyecto E/S:*** *Pantalla y temporizadores.* ***1ª Entrega:*** *diseño*	***2 de abril***
11	***6 de abril*** ***(horario miércoles)*** ***Proyecto E/S***	***7 de abril*** ***(horario de jueves)*** ***Control: Teoría de E/S***	***8 de abril*** ***(horario de viernes)***
Vacaciones			
12	***21 de abril (martes) -semana de horario especial-*** ***Proyecto E/S***		
13	***28 de abril***	***29 de abril*** ***Proyecto E/S***	***30 de abril***
14	***10 de mayo*** ***Tema 4***	***11 de mayo*** ***Proyecto E/S***	***7 de mayo*** ***Tema 4***
15	***11-15 de mayo*** ***Entrega final del Proyecto E/S***		

3 Descripción del proyecto

Como hemos indicado, en este proyecto tenéis que desarrollar un juego muy sencillo para una consola **Nintendo DS**. Cada grupo va a poder diseñar el juego que quiera siempre que se pueda realizar en el tiempo estimado y que cumpla las condiciones exigidas.

Debéis utilizar la **pantalla superior** (también denominada **pantalla secundaria**) para mostrar texto y la pantalla inferior (**pantalla táctil** o **pantalla principal**) para mostrar los gráficos.

Como mínimo **dos teclas** deben controlarse **por interrupción** y otras **dos por encuesta**. La **pantalla táctil** debe controlarse por **encuesta** y el **temporizador** por **interrupción.**

4 Los gráficos y el juego

En este trabajo se trata de hacer un juego gráfico muy rudimentario (de hecho ni lo llamamos videojuego) y muy alejado del nivel gráfico de los videojuegos actuales que han evolucionado considerablemente en los últimos cuarenta años. Pero de cara a la realización de esta práctica conviene conocer los siguientes conceptos:

- **Pixel**.

Un **pixel** (acrónimo en inglés de *Picture Element*) es cada punto en una imagen digital, es la unidad básica de color homogéneo. En otras palabras, cada punto de la pantalla.

- **Tile**.

Un **tile** (en inglés, baldosa) es cada elemento gráfico que se utiliza para construir o "alicatar" un fondo. Cada tile puede ser diferente y normalmente del mismo tamaño.

Sin embargo, para este proyecto, la gestión del fondo ya está resuelta y no es necesario manejar **tiles** (que son de **8x8 píxeles**). El fondo se generan a partir de una imagen de **256x192 píxeles**, que se extraen de un fichero externo de la plantilla (**./gfx/Fondo.png**) y un programa adjunto al compilador lo transforma en código accesible por la consola.

- **Sprite**.

Un **sprite** (del inglés, duende) es cada elemento móvil o personaje protagonista de un videojuego. Son mapas de bits de dos dimensiones, normalmente pequeños. En nuestro caso de **16x16 píxeles**.

La plantilla que adjuntamos puede manejar **128 sprites**. Las funciones que los manejan tienen que identificarlos con un índice que puede tomar valores entre **0** y **127**. El valor **127** se ha reservado para identificar la raqueta, por lo que para las bolas se pueden utilizar cualquier valor entre **0** y **126**.

Las dos pantallas de una consola **Nindendo DS** tienen el mismo tamaño, **256x192 píxeles**. Sin embargo las vamos a utilizar de dos formas diferentes.

- En la **pantalla táctil** (**pantalla principal** o **pantalla inferior**) se van a mostrar los gráficos. En la Figura 1 se puede ver cómo se organizan las coordenadas de los píxeles. El origen de referencia, **(0,0)**, es la esquina superior izquierda. El eje **x** se describe del **0** al **255** de izquierda a derecha; y el eje **y**, del **0** al **191**, de arriba a abajo.
- La **pantalla superior** (**pantalla secundaria**) se va a utilizar en modo texto. Gracias a que la librería **libnds** nos ofrece el procedimiento **consoleDemoInit()**, que nos permite escribir[1] en la **pantalla superior** mediante **printf()** o **iprintf()**, pero perdiendo la visualización de los fondos.

  Para elegir la posición del texto en la pantalla, se usará la secuencia de escape **\x1b[<linea>;<columna>H** usando como valor para la **<linea>** un entero entre **0** y **23**, y el valor de la **<columna>**, un entero entre **0** y **31**. El origen de referencia, **(0,0)**, es la esquina superior izquierda (ver *Figura 1*).

1- *Cada carácter del texto se representa con un **tile** (**8x8 píxeles**).*

Un ejemplo de su utilización sería: **iprintf("\x1b[%d;%dHtexto...", lin, col);**

Figura 1. Coordenadas de las pantallas.

Todo el código necesario para manejar los gráficos se proporciona ya implementado. Las funciones para trabajar con los sprites se adjuntan en el fichero sprites.c.

5 Desarrollo del proyecto

5.1 Plantilla del proyecto

El proyecto debe realizarse en lenguaje **C**. Para ello, y para no tener que partir desde cero, os proporcionamos una plantilla con las estructuras y funciones básicas para este proyecto. Por tanto, en **eGela** disponéis del fichero comprimido "**Plantilla proyecto.zip**" que contiene un fichero y tres directorios:

- **Makefile**. Fichero para compilar.
- **gfx**. Directorio que contiene las figuras para el fondo.
- **include**: Directorio que contiene los ficheros de cabecera.
- **source**: Directorio que contiene los ficheros con el código fuente.

Si se va a realizar el proyecto tal y como está propuesto, no es necesario modificar nada ni en **Makefile**. El desarrollo del juego sólo debería implicar la modificación de ficheros de los directorios **gfx, include** y **source**.

En el directorio **include** se encuentran los ficheros de cabecera (**.h**), donde se declaran las funciones definidas en los correspondientes ficheros fuente. Las rutinas que se añadan en los ficheros del directorio **source** deben ser añadidas a los correspondientes ficheros del directorio **include.**

- **defines.h**

  En este fichero se añadan las **defines** necesarias para poder utilizar nombres más significativos (estados, teclas, etc). Debéis añadir todos los que consideréis oportunos.

- **rutservs.h**
- **fondos.h**

- **sprites.h**
- **teclado.h**
- **temporizadores.h**

En el directorio **source** se encuentran los ficheros fuente (**.c**):

- **fondos.c**

  Rutinas para el manejo de fondos. No es necesario modificar nada.

- **sprites.c**

  Rutinas para el manejo de **sprites**. No es necesario modificar nada.

- **graficos.c**

  Rutinas para el manejo de gráficos. No es necesario modificar nada.

- **main.c**

  Programa principal.

- **rutserv.c**

  La rutina **interrupciones()** para programar los controladores e inicializa la tabla de interrupciones la codificamos en este fichero.

- **teclado.c**

  Se incluyen aquellas rutinas relacionadas con el control del teclado.

- **temporizadores.c**

  Se incluyen aquellas rutinas relacionadas con el control del tiempo.

5.2 Modo de trabajo

Por lo tanto, deberíais crear un directorio en vuestra máquina virtual donde copiar la plantilla con la misma estructura que se proporciona. Una propuesta es crearla en vuestro directorio por defecto con el nombre que consideréis más adecuado (p.e. **proyecto**), al mismo nivel que el directorio **ejemplo** de sesiones anteriores.

Para compilar el trabajo tendréis que trabajar de forma similar a las sesiones anteriores, es decir, desde el nuevo directorio (como antes lo hacías desde **ejemplo**) de la forma:

```
user@ubuntu:~/proyecto$ make
```

Se crearán tres fichero con el mismo nombre que el directorio de trabajo (en este caso serían **proyecto.arm9**, **proyecto.elf** y **proyecto.nds**).

Para testear el proyecto (**proyecto.nds**) podréis utilizar el programa gratuito **NO$GBA** (*No Cash GBA -Game Boy Advance-*) que es un emulador de **Nintendo DS** (y de la **Game Boy Advance**) para **Microsoft**. En este caso **Desmume** está desaconsejado porque no funciona bien con algunas interrupciones.

La versión **Linux** es **nocashgba** que se lanzará de la forma:

```
user@ubuntu:~/proyecto$ nocashgba proyecto.nds
```

El emulador **NO$GBA** mostrará las dos pantallas de una consola **Nintendo DS** y las teclas las tomará del teclado del ordenador de la forma[2]:

NDS	*teclado*	*NDS*	*teclado*	*NDS*	*teclado*
Up	***Q***	***A***	***Space***	***L***	***Alt(L)***
Down	***A***	***B***	***Tab***	***R***	***Alt R***
Left	***O***	***Select***	***Control(L)***	***X***	***X***
Right	***P***	***Start***	***Return***	***Y***	***Z***

6 Material a entregar

Cada grupo deberá realizar dos entregas:

6.1 Primera entrega

En día miércoles **1 de abril de 2020**, cada grupo deberá entregar el **autómata** y la descripción del proyecto que propone a través de **eGela**..

6.2 Entrega final

En la semana del **15 al 19 de mayo de 2017**, cada grupo deberá entregar el proyecto final a través de **eGela**. Cada grupo debe subir un fichero comprimido que debe llamarse **Gxx-proyecto.zip**, (u otro formato habitual de compresión de archivos como **.7z**, **.tar**,…), siendo **xx** el número del grupo.

El documento comprimido deberá contener:

1. Una **memoria** que contenga la siguiente información:
   a) Descripción del trabajo que sirva para entender el diseño del programa.
   b) El autómata correspondiente al diseño utilizado.
   c) Las funciones programadas acompañadas de una explicación.
   d) Una descripción de los problemas encontrados a lo largo del desarrollo del proyecto y cómo se les ha dado solución
2. El directorio de trabajo completo (**Makefile**, **buid**, **gfx**, **include**, **source**,…) con los **códigos fuente** y el **ejecutable**, que debe llamarse **Gxx.nds**, siendo **xx** el número del grupo.

7 Evaluación

Es imprescindible realizar este proyecto para superar la asignatura. Esta tarea se evalúa con el **20%** sobre el total de la asignatura.

Lo señalado en los apartados anteriores son los requerimientos mínimos para aprobar el proyecto. Pero se pueden hacer otras consideraciones que creáis oportunas y que añadan valor al proyecto. A continuación se sugieren algunas (que se pueden combinar):

2 - ***NO$GBA*** *permite configurar vuestro propio juego de teclas.*

Criterios para evaluar la memoria (40%)			
	Bien	***Suficiente***	***Mal***
Descripción	*En el documento se distingue claramente la parte análisis y la de síntesis. Presenta el contexto del trabajo y termina con unas conclusiones.*	*Aunque el documento tiene una estructura más o menos definida, algunas veces mezcla contenidos.*	*El documento carece de estructura.*
Legibilidad	*El documento no tiene faltas de ortografía y usa un lenguaje que se entiende perfectamente.*	*Contiene alguna errata. A veces hay que esforzarse para entender lo que quiere expresar.*	*Contiene faltas de ortografía, las frases están mal construidas y no se entiende qué pretende expresar.*
Presentación	*Los detalles de presentación están bien cuidados.*	*Descuida algunos detalles de la presentación, aunque en general resulta agradable.*	*El formato del documento descuida los detalles de presentación.*
Autómata	*Describe perfectamente el comportamiento del juego*	*Describe el comportamiento del juego en general, aunque no contempla algunos casos que resultan ambiguos.*	*No se contemplan los diferentes estados y/o faltan transiciones importantes.*
Explicación de las funciones	*Se explica cada función de forma clara y precisa. Se indica que hace cada una y para qué es necesaria.*	*Se explica cada función aunque a veces hay que esforzarse para entenderlo. Algunas funciones no se sabe para que se van a utilizar*	*No se explican las funciones o se incluye el código en vez de una explicación.*
Estimación del tiempo	*Se realiza una estimación razonable para cada una de las partes del trabajo.*	*Se realiza una estimación para cada una de las partes del trabajo, aunque la proporción entre ellas no resulta razonable*	*No se realiza una estimación del tiempo.*
Criterios para evaluar el código y ejecutable (60%)			
	Bien	***Suficiente***	***Mal***
Código	*El código desarrollado está expresado de forma clara y estructurada.*	*El código desarrollado no está expresado de forma estructurada, aunque funciona.*	*El código desarrollado es imposible de seguir (leer)*
Funcionamiento	*El funcionamiento del programa satisface todos los casos de prueba.*	*El funcionamiento del programa satisface la mayoría de los casos de prueba, y todos los importantes.*	*El funcionamiento del programa no satisface muchos de los casos de prueba.*