# MEMORIA
# Desarrollo de una aplicación de E/S para Nintendo DS

G33

May 20, 2019

# Contents

# 1 Introducción

El tema de este proyecto es el desarrollo de una aplicación para la Nintendo DS. Para realizar este proyecto usaremos la herramienta de desarrollo "devkitPro", y programaremos la aplicación en el lenguaje C.

Antes de empezar con el desarrollo realizaremos un autómata, que tendrá como objetivo diferenciar los distintos estados por el que pasa la máquina y qué ha de hacerse para cambiar de un estado a otro.

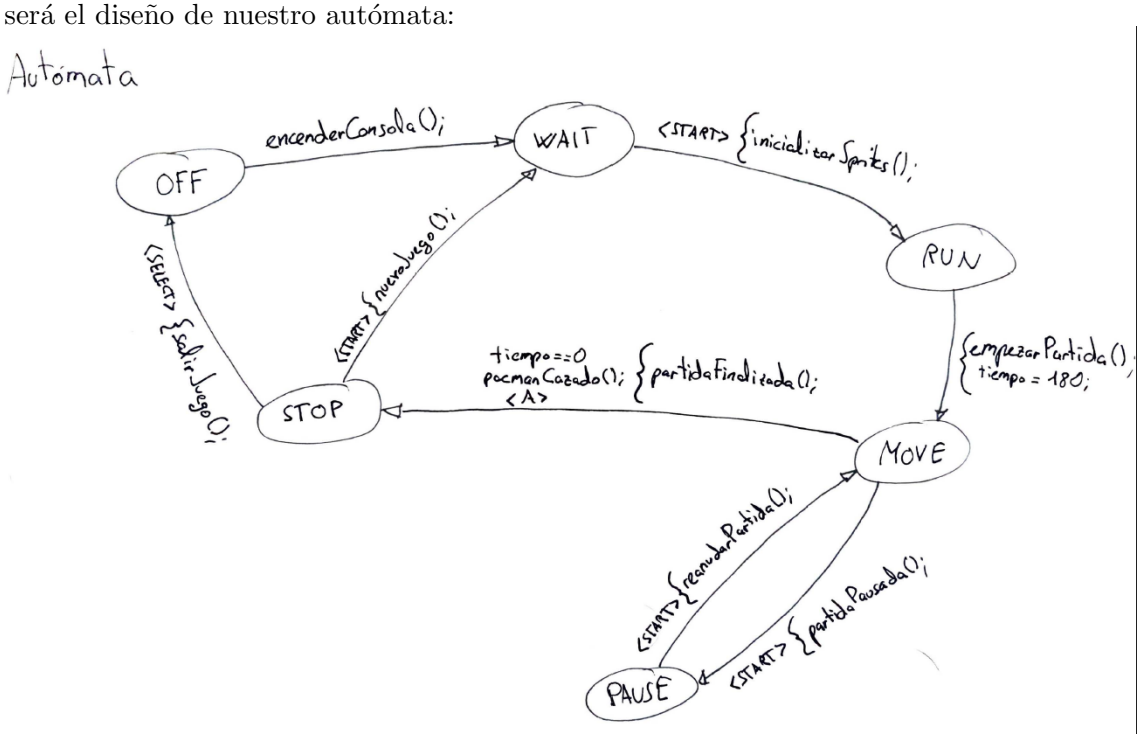
El juego consistirá en una versión sencilla del popular juego "Pacman".

Los objetivos de este proyecto son aprender a desarrollar una aplicación con el diseño de un autómata, manejar la sincronización entre el procesador y los periféricos y llevar a cabo un control de los temporizadores.

Para empezar diseñaremos un autómata. Idearemos los estados por los que pasará la máquina. Después explicaremos la funciones del código de nuestro programa, el porqué de cada función y como hemos estructurado el código. También explicaremos el tiempo que hemos dedicado a cada parte del proyecto. Y finalmente, las conclusiones.

## 2 Autómata

Este será el diseño de nuestro autómata:




El primer estado, que llamaremos ST-OFF, será un estado que no hará nada, simplemente lo utilizaremos cuando el programa finalice entre en este estado. Lo usaremos en la clase main, durante el bucle "while". El programa recorrerá el bucle mientras el estado no sea ST-OFF. De esta forma cuando el programa entre en este estado, el bucle finalizará y el juego terminará. Entraremos en este estado cuando estando en el estado ST-STOP, pulsemos la tecla SELECT.

El segundo estado, ST-WAIT, es el primer estado que el programa entra una vez se ejecuta la aplicación. En este estado seleccionaremos la dificultad con las teclas L y R. Una vez seleccionemos la dificultad si pulsamos la tecla START pasaremos al estado ST-RUN.

El estado ST-RUN, lo usaremos de transición. En este estado inicializaremos los objetos necesarios para que el juego funcione. Estos se ejecutarán según el nivel de dificultad que hallamos seleccionado. Una vez se ejecuten pasaremos al estado ST-MOVE.

El estado ST-MOVE, será en el que controlaremos el movimiento de todos los sprites. Con el teclado direccional, controlaremos la dirección a la que señala Pacman, y con el temporizador controlaremos el movimiento tanto de Pacman como de los fantasmas. En este estado tenemos 2 opciones, pasar al estado ST-PAUSE y al estado ST-STOP.

Si pulsamos la tecla START, pausaremos el juego, pasando al estado ST-PAUSE, hasta que volvamos a pulsar START y volvamos al estado ST-MOVE.

Para pasar al estado ST-STOP hay 3 formas diferentes. La primera es que la variable que hemos inicializado en 180 llegue a 0. La segunda que un fantasma haya capturado a Pacman y la tercera que pulsemos la tecla A.

Una vez en el estado ST-STOP, podemos finalizar el programa pulsando SELECT y pasando al estado ST-OFF, o podemos pulsar la tecla START y empezar una nueva partida, por lo que pasaremos al estado ST-WAIT.

# 3 Funciones

Para estructurar mejor el código de nuestro programa, hemos creado 2 clases. La primera clase la llamaremos "obsprites" y será la encargada de controlar los sprites y la interacción entre ellos.

Para identificar los sprites necesitaremos 2 datos, la coordenada y la direccion del sprite. Con la primera sabremos donde está el sprite en pantalla y la segunda a que dirección se tiene que mover. Para esto crearemos un "struct" con 3 variables, 2 para las coordenadas y 1 para la dirección. Crearemos una variable de esta estructura por cada sprite que vamos a usar.

Dentro de esta clase, definiremos la funciones que modifican estos objetos:

- ***inicializarSprite()***: Pasando un objeto como parámetro, genera 2 número aleatorios para las 2 variables de las coordenadas. La usaremos al iniciar la partida.
- ***moverSprite()***: Pasando un objeto como parámetro, modifica las variables de las coordenadas según la variable de dirección. Con esta función moveremos los sprites.
- ***cambiarDireccion()***: Modifica la dirección del objeto, pasado como primer parámetro, por la dirección que pasamos como segundo parámetro. Con esta función cambiaremos la dirección de los sprites como deseemos.
- ***perseguirPac()***: Esta función pasa 2 objetos como parámetros. En este caso cambia la dirección del primer objeto, según la posición que tenga respecto al segundo objeto. Esta función la usaremos para cambiar la dirección de los sprites de fantasma. Que tendrán que cambiar según la posición de Pacman.
- ***capturado()***: Esta función booleana pasa 2 objetos como parámetros. Devuelve true si la distancia de los 2 objetos es menor que 5 en las 2 coordenadas. Esta función la usaremos para comprobar tanto la captura de Pacman por los fantasmas, como la captura de puntos por Pacman.

La segunda clase, llamada "funciones", definiremos la gran parte de las funciones que usaremos tanto en la clase "main", como en la clase "teclado" y "temporizador". Estas funciones la hemos diferenciado de 3 formas. Por un lado las funciones que manejarán la consola. Por otra parte la funciones que usaremos al resetear partida y por último funciones que usaremos para el funcionamiento de los sprites en partida.

Funciones de la consola:

- ***msgTiempoScore()***: Muestra en consola el tiempo de juego, la puntuación y los saltos disponibles. Esta función se ejecuta en el temporizador pues se actualiza constantemente.

- ***msgPartidaTerminada()***: Mensaje que se muestra al finalizar la partida, donde sale la puntuación, el tiempo y el motivo de finalización.

- ***msgInstrucciones()***: Mensaje que se muestra durante la partida indicando que hace cada tecla.

- ***msgPause()***: Mensaje que se muestra durante el estado ST-PAUSE.

- ***msgFacil(), msgMedio() y msgDificil()***: Mensajes identicos pero en la linea que imprime en pantalla "FACIL INTERMEDIO DIFICIL" resalta en color verde la palabra que le corresponde.

- ***msgFin()***: Mensaje que se muestra al entrar en el estado ST-OFF

- ***msgClean()***: Esta función limpia la consola completamente y deja las lineas que corresponde a cada grupo.

Funciones de reinicio:

- ***reiniciarStats()***: Esta función resetea las variables globales que vamos a utilizar en partida, como el tiempo de juego, la puntuación...

- ***limpiarSubPantalla()***: Esta función limpia la pantalla inferior de todo los sprites que haya después de finalizar la partida.

- ***partidaTerminada()***: Función para cambiar de estado al finalizar una partida. Pasamos la variable del motivo de finalización como parámetro.

Funciones de sprites: (Estas funciones a diferencia de la clase "obsprites", manejan el funcionamiento de sprites específicos)

- ***inicializarSprites()***: Esta función inicilializa todos los sprites necesarios para empezar la partida según el nivel de dificultad.

- ***saltoPacman()***: Esta función realiza 5 movimientos de Pacman en el tick que tendría que hacer uno. Simulando que Pacman ha hecho un salto.

- ***saltoFantasma()***: Misma función que el salto de Pacman solo que con estadísticas y condiciones diferentes.

- ***movimientoPacman()***: Realiza el movimiento completo de Pacman. Primero llamamos a la función moverSprit() que cambiará las coordenadas del Pacman y luego llamamos a la función de MostrarMsPacman().

- ***movimientoFantasma()***: Realiza el movimiento completo del fantasma (diferente según el tipo de fantasma). Es parecido al de Pacman, pero en este también llamaremos a la función perseguirPac(), que cambiará la dirección del fantasma persiguiendo a Pacman.

- ***comerPunto(), comerCereza() y comerMierda()***: Funciones que

comprobarán si Pacman ha capturado alguno de estos items, y si es así creara una nueva coordenada para el sprit y sumará (o restará) en la puntuación.

A parte de estas 2 clases, hemos creado una función dentro de la clase "sprites" llamada GirarPacman(). Esta función afecta al dibujo del sprite, por eso hemos creído que encaja más en esta clase.

- ***GirarPacman()***: Esta función gira el sprite de Pacman, de modo que su cabeza señale a la dirección que apunta el objeto.

Para la dirección del objeto hemos definido que 0 sea a la derecha, 1 arriba, 2 izquierda y 3 abajo. Pero no lo hemos hecho de manera arbitraria. Para la función GirarPacman(), debemos indicar el angulo que tiene que girar el sprite. Para ello hemos programado la función de rotar de manera que pasando como parámetro uno de estos números, gire el sprite a la dirección deseada.

# 4 Estimación de tiempo

El diseño del programa

Para el autómata, estimamos que necesitaríamos una hora para realizar un esquema y pasarlo a limpio. Finalmente si tenemos en cuenta que primero se realizo un borrador y luego hicimos el autómata final una vez terminado el programa, la duración total ha sido de 1:30.

La parte de programación es la de mayor duración. Estimamos que entre 10 y 12 horas serían suficientes. Aunque finalmente, entre que algunas partes han sido más difíciles de lo esperado y que al terminar queríamos perfeccionarlo, han llegado a ser 16 horas.

Las pruebas las hemos realizado mientras programábamos, una vez terminada una función se ejecutaba para ver si funcionaba, así que es más difícil hacer una estimación. En total digamos que han sido entre media hora y una hora de pruebas.

# 5 Conclusión

Como conclusión, hemos trabajado en el diseño y desarrollo de una aplicación de Nintendo DS. La parte más importante creemos que ha sido el transladar la idea del diseño, como el autómata, al código de nuestro programa. De la manera que es más fácil identificar cada parte o estado de un programa y trabajar cada uno de ellos por separado.

También en cuanto al manejo de los controladores y las diferentes formas que hay de que un procesador responda a la interacción de un teclado (encuesta/interrupción).