VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 1

Introducción

Las tres primeras prácticas de la asignatura consistirán en la implementación de una aventura conversacional sin texto.

Una aventura conversacional[1], también conocida como juego de aventuras de texto o juego de texto interactivo, es una forma de entretenimiento interactivo en la que los jugadores participan en una narrativa digital a través de la conversación escrita. A diferencia de los videojuegos tradicionales, las aventuras conversacionales se basan principalmente en texto en lugar de gráficos avanzados. Los jugadores interactúan con el juego a través de comandos de texto y reciben respuestas en forma de texto. Las aventuras conversacionales se centran en la narrativa. Los jugadores asumen roles en historias complejas y a menudo tienen que tomar decisiones que afectan el curso de la trama. La calidad de la escritura y la inmersión en la historia son aspectos fundamentales.

La primera sesión se centra en los fundamentos necesarios para programar la NDS. En concreto en mostrar texto en las dos pantallas y en controlar la pulsación de los botones de la botonera.

Documentación

Se recomienda leer los capítulos 1, 2, 3 y 4 sobre programación con la NDS, que se pueden encontrar en el aula virtual, y realizar los ejercicios que allí se propopen.

[1] Definición obtenida usando ChatGPT

Trabajo a realizar

El objetivo de esta sesión es aprender los fundamentos básicos para desarrollar videojuegos para la NDS.

- Ejercicio 1.1: Implementa y ejecuta el siguiente programa “Hola Mundo” (ver el capítulo 2):

```
#include <nds.h>
#include <stdio.h>
int main(void)
{
  consoleDemoInit();
  iprintf("Hola Mundo");
  while(1)
  { }
}
```

- Ejercicio 1.2: Implementa un programa que escriba un texto en la pantalla superior de forma centrada, tanto verticalmente como horizontalmente.

- Ejercicio 1.3: Implementa un programa que escriba un texto en la pantalla inferior de forma centrada, tanto verticalmente como horizontalmente.

- Ejercicio 1.4: Implementa un programa que escriba un texto tanto en la pantalla inferior como en la superior.

- Ejercicio 1.5: Implementa un programa que al pulsar una flecha de la botonera, escriba en la pantalla superior un mensaje para informar de cual es la tecla que se ha pulsado.

- Ejercicio 1.6: Implementa un programa que muestre de inicio el número 0 en el centro de la pantalla superior. Si se pulsa la flecha de la derecha se sumará un valor (al inicio 1) al número mostrado en la pantalla superior. Si se pulsa la flecha de la izquierda, se restará el valor a dicho número. Además, en la pantalla inferior se mostrará el valor a ser sumado o restado. Si se pulsa la flecha hacia arriba, el valor se multiplicará por dos y si se pulsa la tecla hacia abajo, entonces se dividirá el valor por dos, siendo el mínimo 1.

Evaluación

En cada sesión, según los ejercicios realizados se obtendrá una nota de **4, 6, 8 o 10 puntos**. Consultar el documento de presentación de la asignatura, incluido en el Aula Virtual, para saber más detalles de la metodología que se usará para evaluar.

- 4 puntos: Ejercicio 1.1.
- 6 puntos: Los anteriores y los ejercicios 1.2, 1.3 y 1.4.
- 8 puntos: Los anteriores y el ejercicio 1.5.
- 10 puntos: Todos los ejercicios..

VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 2

Introducción

La segunda sesión se centra en explicar como desarrollar correctamente videojuegos introduciendo las estructuras de datos más adecuadas.

Para ello, se implementará un sencillo juego en el que el usuario se mueve por la pantalla (usando las cuatro flechas de la botonera) y tiene que recoger una moneda que se muestra en la pantalla. La posición del usuario se mostrará con una “X” y la moneda con el número cero “0”.

GameState

El GameState (o estado del juego) es la estructura de datos que almacena todos los datos necesarios para, como su nombre indica, almacenar el estado del juego. Si estuvieramos en el Ajedrez, el GameState almacenaría la posición de todas las piezas y a quien le toca jugar en el siguiente turno.

En nuestro caso, almacenará:

- La posición del jugador (fila y columna).
- La posición de la moneda (fila y columna).
- Si la moneda ha sido recogida o no.

El GameState tendrá una función **IsTerminal** que devuelve *true* si se ha llegado al final del juego (el jugador ha recogido la moneda) y false en caso contrario.

También tendrá una función **Reset** que inicializa el estado del juego a su estado inicial.

Action

Implementa la acción a realizar por el jugador. En nuestro caso, será posible realizar cuatro acciones: Arriba, Abajo, Derecha e Izquierda.

ForwardModel

El Forward model es la estructura de datos donde se implementan las reglas del juego. Tendrá, como función más importante, una función **Step** que dado un GameState y la acción a realizar por el jugador, modificará el GameState “jugando” la acción. En el caso del Ajedrez, esta función realizaría el movimiento del jugador y modificaría el estado del juego usando las reglas correspondientes a este juego. En nuestro caso, la acción será el movimiento a realizar. También se comprobará si el jugador ha recogido la moneda. Ten en cuenta que, si el jugador esta en el límite superior de la pantalla y selecciona la acción de subir, no debe ocurrir nada. Lo mismo se aplica a cuando está en los otros límites.

Programa principal

Siguiendo el modo de funcionamiento de Unity3D, crearemos dos funciones principales **SetUp()** donde se realizan todas las acciones necesarias para inicializar el juego y **Update()** que se ejecutará en cada iteración**.**

Trabajo a realizar

El código del juego se encuentra en el aula virtual.

- Ejercicio 2.1: Descargate el código del juego, estudia el código y ejecuta el juego.
- Ejercicio 2.2: Modifica el juego para que ahora aparezca un segundo jugador en la pantalla. El segundo jugador se moverá con los botones A, B, X e Y. El primer jugador que recoja la moneda, será el ganador.
- Ejercicio 2.3: Modifica el programa para mostrar en la pantalla superior quien es el ganador.
- Ejercicio 2.4: Modifica el programa para que cuando finalice la partida, sea posible volver a jugar pulsando el boton START.
- Ejercicio 2.5: Modifica el programa para que la posición inicial de la moneda y de los dos jugadores sea establecida de forma aleatoria. Al final del documento hay un ejemplo de como obtener un número aleatorio entre 0 y 23.
- Ejercicio 2.6: Modifica el programa para que existan varias monedas en vez de una. El ganador será quien recoja más. Al final del documento hay un ejemplo de como usar la clase **Vector** que te servirá de ayuda.

Evaluación

- 4 puntos: Ejercicio 2.1.
- 6 puntos: El anterior y los ejercicios 2.2 y 2.3.
- 8 puntos: Los anteriores y los ejercicios 2.4 y 2.5.
- 10 puntos: Todos los ejercicios..

Programas de utilidad

Programa para genera un número aleatorio entre 0 23

```
#include <cstdlib> // for rand() and srand() functions
#include <ctime>   // for time() function to seed rand()

int main()
{
	// Seed the random number generator with the current time
	std::srand(static_cast<unsigned int>(std::time(nullptr)));

	int N = 23;
```

```
	// Generate and print a random number between 0 and N
	int randomNumber = std::rand() % (N + 1);
	return 0;
}
```

Programa para mostrar como funciona la clase Vector

```
#include <iostream>
#include <vector>

int main()
{
	// Create an empty vector of integers
	std::vector<int> numbers;

	// Add elements to the vector
	numbers.push_back(1);
	numbers.push_back(2);
	numbers.push_back(3);

	// Access elements in the vector
	std::cout << "Vector elements:";
	for (int i = 0; i < numbers.size(); ++i)
	{
	  std::cout << " " << numbers[i];
	}
	std::cout << std::endl;

	// Modify an element
	numbers[1] = 42;

	// Access elements using iterators
	std::cout << "Vector elements (using iterators):";
	for (std::vector<int>::iterator it = numbers.begin();
	     it != numbers.end();
	     ++it)
	{
	      std::cout << " " << *it;
	}
	std::cout << std::endl;

	// Remove an element
	numbers.pop_back();

	// Check if the vector is empty
	if (numbers.empty()) std::cout << "Vector is empty." << std::endl;
	else std::cout << "Vector is not empty." << std::endl;


	// Get the size of the vector
	std::cout << "Vector size: " << numbers.size() << std::endl;

	return 0;
```

```
}
```

VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 3

Introducción

En la tercera sesión se implementará una aventura conversacional. Se partirá de una aventura ya implementada y el trabajo a realizar consistirá en realizar algunas mejoras al juego.

Si lo deseas, puedes implementar una aventura diferente a la que se propone como punto de partida. Si ese es el caso, la nota que obtendrás tendrá relación con la cantidad de trabajo realizado y su calidad en relación al trabajo propuesto en este documento.

Juego inicial

El mundo del juego tiene 11 habitaciones (Ver Figura 1). En cada habitación hay puertas que llevan a otras habitaciones. Algunas veces hay 4 puertas (como en la habitación 3), otras dos (como la hbaitación 10), etc. Algunas habitaciones (5 y 7) tienen un baúl donde hay una llave. Para entrar en la habitación 11 y salir de la mazmorra son necesarias dos llaves. Sin haber obtenido las dos llaves no será posible llegar a la habitación 11 y acabar el juego.

Cuando estás en una habitación, la pantalla te muestra un texto descriptivo. En la versión inicial los textos son genéricos. Por lo tanto, el juego no tiene narrativa.

Las posibles acciones dependen de la habitación donde esté el jugador. En términos generales son las cuatro flechas de la botonera para cambiar de habitación y el botón A para abrir los cofres y la puerta final.

Además de las clases que ya se vieron en el laboratorio anterior: *GameState*, *ForwardModel* y *Action*, se ha implementado la clase *World* con información sobre el mundo donde transcurre el juego.

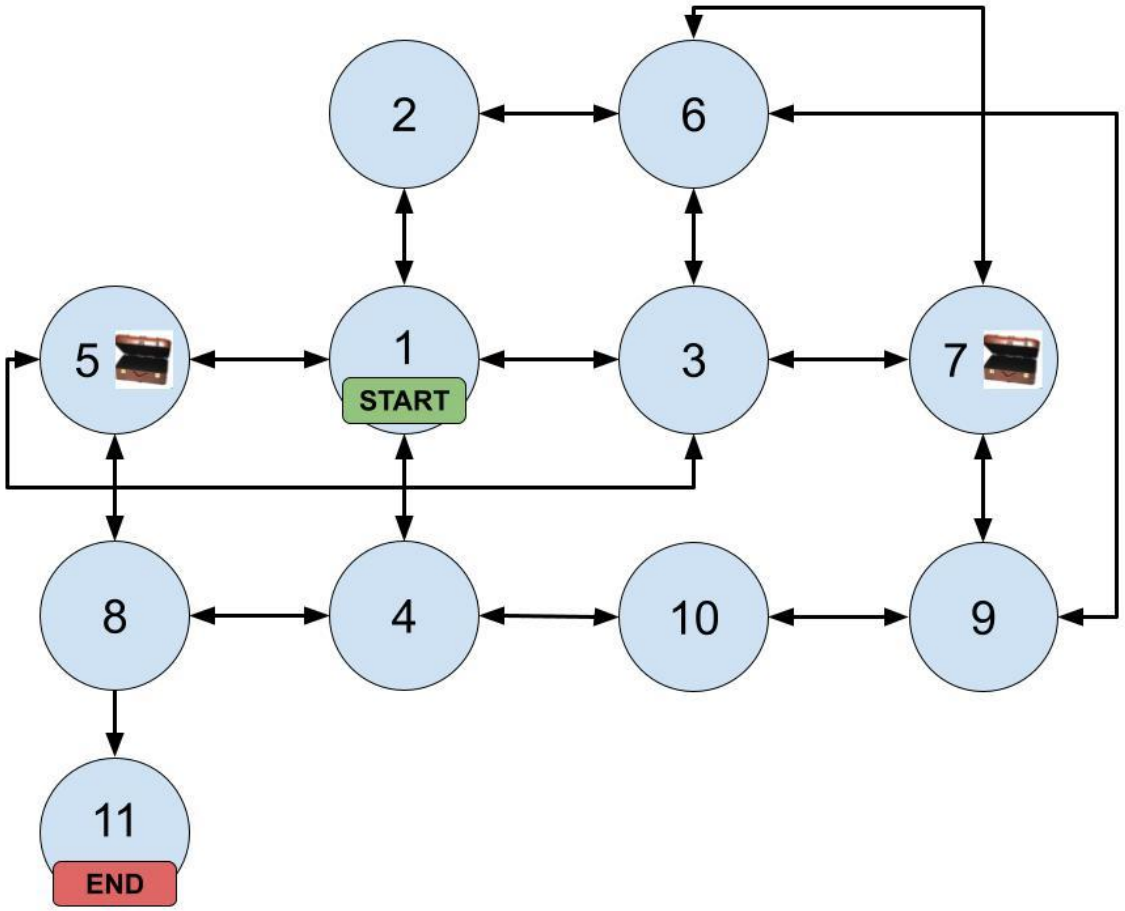

Figura 1: Esquema del mundo programado.

Trabajo a realizar

El código del juego se encuentra en el aula virtual.

- Ejercicio 3.1: Descargate el código del juego, estudia el código y ejecuta el juego. Modifica los textos de cada habitación para proporcionar una historia.

- Ejercicio 3.2: Añade más habitaciones al juego.

- Ejercicio 3.3: Añade un enemigo que, cada N segundos, cambie de habitación. El enemigo atacará al jugador si coinciden en una habitación. El programa 3.8 que se encuentra en el capítulo 3 del libro, proporciona un ejemplo que te servirá de ayuda para aprender a controlar el tiempo en la NDS.

- Ejercicio 3.4: Añade también un arma que se tendrá que encontrar para poder ser usada para combatir al enemigo y todas las modificaciones que consideres oportunas para tener un juego más completo y entretenido.

Evaluación

- 4 puntos: Ejercicio 3.1.
- 6 puntos: El anterior y el ejercicio 3.2.
- 8 puntos: Los anteriores y el ejercicio 3.3.
- 10 puntos: Todos los ejercicios.

VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 4

Introducción

La cuarta sesión tiene como objetivo introducir el modo gráfico Framebuffer. Se recomienda la consulta de los capítulos 5 y 6 del libro sobre la programación con la NDS.

Evaluación

Los ejercicios a realizar son los que se encuentran en el capítulo 6 del libro.

- 4 puntos: Ejercicio 6.1.
- 6 puntos: El anterior y los ejercicios 6.2 y 6.3
- 8 puntos: Los anteriores y los ejercicios 6.4 y 6.5
- 10 puntos: Todos los ejercicios (los anteriores y el 6.6).

VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 5

Introducción

En la quinta sesión se va a transformar el juego realizado en la tercera sesión para que sea un juego del tipo Point & Click. El juego mostrará una imagen, en la pantalla inferior, ilustrativa de la habitación/escenario donde nos encontremos. El usuario pulsará en la pantalla táctil (con el ratón en el simulador) para realizar las acciones correspondientes a esa pantalla. En la pantalla superior se mostrará el texto correspondiente a cada localización.

Se partirá del trabajo realizado en la sesión 3. Si no pudiste acabar la sesión 3, puedes partir del código original proporcionado por los profesores de esa sesión.

Los pasos recomendados a seguir son los siguientes:

1. Modifica el código para que muestre una imagen usando el modo framebuffer en la pantalla inferior. El texto se mostrará en la pantalla superior. La imagen será siempre la misma independientemente de la localización donde esté el jugador. El objetivo es ser capaz de mostrar a la vez, texto en la pantalla superior y una imagen mediante el modo framebuffer en la inferior.

2. Modifica el código para que la interacción con el mundo se realice usando la pantalla táctil (ver el Capítulo 3) en vez de con los botones de la botonera. Por ejemplo, si pulsas en la derecha de la pantalla tendrá el mismo efecto que si pulsas el botón derecho de la botonera y lo mismo para el resto de direcciones. No te olvides de las acciones de abrir cofres o similares.

3. Usa una imagen diferente para cada localización. De momento, puedes usar una imagen simple, editada para poner un número que indique la localización donde estás. Tendrás que tener tantas imágenes como localizaciones. En este punto, no importa la calidad de las imágenes. Lo importante es que, al cambiar de localización, cambie la imagen de forma correcta. La memoria de la NDS es limitada, se recomienda, a pesar de no ser lo más óptimo, usar siempre el mismo espacio de memoria y usar dmaCopy para modificar la imagen a mostrar.

4. Crear imágenes que tengan relación con cada localización. Habrá que hacer versiones diferentes en aquellas localizaciones que tengan, por ejemplo, cofres que al principio estén cerrados y luego se abran. También tendrás que tener en cuenta al enemigo. Para atacar al enemigo puedes hacer que, cuando sale, exista un tiempo límite para pulsar sobre él para atacar. Si pasado ese tiempo no has reaccionado, el enemigo te mata y pierdes la partida. Por lo tanto, tendrás que hacer versiones con y sin el enemigo.

Notas importantes

1. Para poder poner una imagen en la NDS tiene que tener el mismo tamaño y formato que las que viste en la sesión 4. Se recomienda ver, con un programa de edición de fotografías, las características de las imágenes vistas en esa sesión con el fin de realizar las imágenes necesarias en esta sesión.

2. Para llegar al 10 es imprescindible tener las imágenes realizadas con anterioridad a la sesión.

Evaluación

- 4 puntos: Completar el primer paso.
- 6 puntos: Completar los dos primeros pasos.
- 8 puntos: Completar los tres primeros pasos.
- 10 puntos: Completar todos los pasos.

VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 6

Introducción

En la sexta sesión se va a introducir el modo teselado. Para ello será imprescindible consultar el capítulo 7.

Evaluación

- 4 puntos: Realizar los ejercicios del capítulo 7: 7.1 y 7.2.
- 6 puntos: Realizar, además, el ejercicio 7.3.
- 8 puntos: Realizar, además, el ejercicio 7.4.
- 10 puntos: Realizar todos los ejercicios del capítulo 7.

VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 7

Introducción

En la séptima sesión se va a introducir la entrada y salida. Para ello será imprescindible consultar los capítulos 8 y 9.

Evaluación

- 4 puntos: Realizar los ejercicios del capítulo 9: 9.1 y 9.2.
- 6 puntos: Realizar, además, los ejercicios 9.3, 9.4 y 9.5.
- 8 puntos: Realizar, además, los ejercicios 9.6, 9.7 y 9.8.
- 10 puntos: Realizar todos los ejercicios del capítulo 9 menos el 9.9.

VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 8

Introducción

En esta sesión, se implementará el popular *Pacman*. La Figura 1 muestra una versión de este juego. Se proporciona para empezar el diseño de las teselas necesarias y el mapa de teselas inicial. El resto del código tendrá que ser realizado por los/las estudiantes. En esta sesión no tendremos en cuenta el movimiento de los fantasmas.

Se recomienda **leer por completo** los ejercicios antes de empezar la implementación del juego.

Los ejercicios son una guía de cómo se puede realizar la implementación paso a paso. Si se llega a los mismos resultados, no hay problema si se siguen otros pasos.

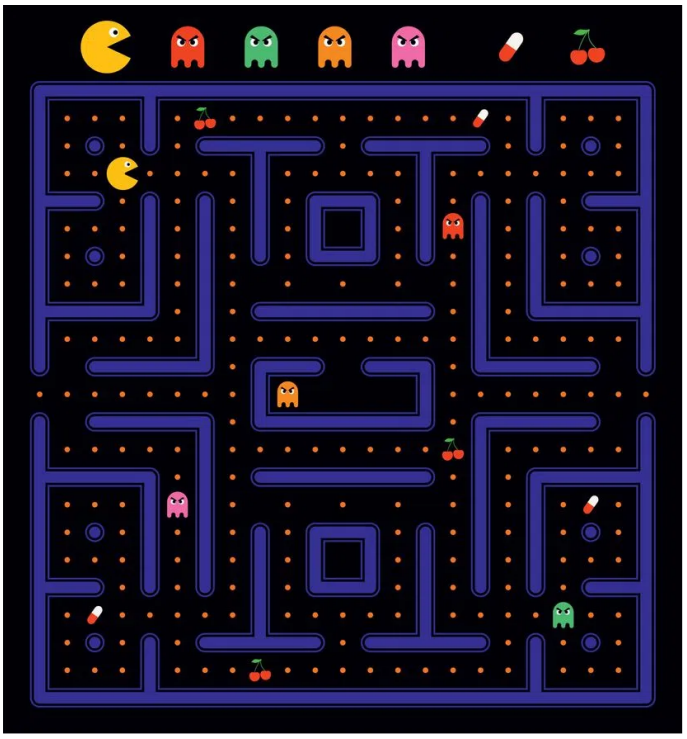

Figura 1: Versión del *Pacman* implementada[1].

[1] Imagen obtenida en https://www.muralsyourway.com/pacman-game-61794625/p

Trabajo a realizar

Ejercicio 1

El juego original del Pacman es de 23x23 casillas. Modifica el diseño original del juego para que ocupe toda la pantalla de la NDS. Es decir deberá ser de 24x32 casillas.

Ejercicio 2

Modifica las teselas y colores de forma que el resultado sea más fiel al original. Por ejemplo, todos los muros de la versión proporcionada por los profesores son rectangulares mientras que en el juego original, hay muros con un lado redondeado.

Ejercicio 3

Crea el *GameState* del juego. El estado del juego deberá almacenar una matriz de la dimensión de la pantalla (i.e. 24x32) con los posibles estados que puede tener cada casilla. El estado del juego también deberá almacenar la posición de *Pacman*.

Ejercicio 4

Crea la clase *Action*. Las acciones posibles son los cuatro posibles movimientos: arriba, abajo, derecha, izquierda.

Ejercicio 5

Crea el *ForwardModel* del juego. Tendrás que tener en cuenta que si *Pacman* se mueve a una casilla con moneda, el estado del juego deberá cambiar para reflejar que, en esa casilla, ya no habrá moneda.

Ejercicio 6

Crea el bucle principal del juego. Tendrás que realizar una función que dado el estado del juego, muestre unas teselas u otras en la pantalla. En la pantalla superior, se deberá mostrar el número de monedas que ha capturado *Pacman* y el tiempo de la partida. El tiempo se debe controlar mediante interrupciones.

Ejercicio 7

Realiza los siguiente pasos:

1. Pantalla de Inicio (Framebuffer): Crea una pantalla de inicio utilizando el modo framebuffer. Configura el programa para esperar a que se presione la tecla "Start" antes de avanzar al juego.

2. Modo Teselado (Juego): Una vez que se presione la tecla "Start", cambia al modo teselado para mostrar la pantalla de juego.

3. Condición de Victoria: Implementa una verificación que se ejecute continuamente para verificar si *Pacman* ha recogido todas las monedas. Cuando *Pacman* haya recogido todas las monedas, cambia al modo framebuffer nuevamente y muestra una pantalla de fin de juego.

4. Pantalla de Fin de Juego (Framebuffer): Diseña una pantalla de fin de juego que muestre un mensaje de victoria al jugador. Puedes incluir estadísticas, en la pantalla de texto, como la puntuación, el tiempo y cualquier otro detalle relevante. Proporciona instrucciones sobre cómo comenzar una nueva partida, como "Presiona Start para jugar de nuevo".

5. Iniciar una Partida Nueva: Implementa la lógica para reiniciar el juego cuando el jugador presione la tecla "Start" en la pantalla de fin de juego. Esto debería llevar al jugador de regreso a la pantalla de inicio, donde pueden comenzar una nueva partida. Asegúrate de que todo el flujo del juego esté bien sincronizado y que las transiciones entre las pantallas se realicen sin problemas.

Evaluación

- 4 puntos: Realizar los ejercicios 1 y 2.
- 6 puntos: Realizar, además, los ejercicios 3 y 4
- 8 puntos: Realizar, además, los ejercicios 5 y 6.
- 10 puntos: Realizar todos los ejercicios.

VJ1214: Consolas y dispositivos de videojuegos
Laboratorio: Sesión 9

Introducción

En esta sesión, se finalizará la implementación del juego *Pacman* añadiendo la parte de los fantasmas.

Si no acabaste la sesión anterior, puedes partir del código que se proporciona.

Trabajo a realizar

Ejercicio 1

Crea teselas para las cerezas y las pastillas y ponlas en la pantalla de juego. Crea también las teselas para los diferentes fantasmas. Debes tener 4 fantasmas de diferente color.

Ejercicio 2

Usando temporizadores, haz que, pasado un tiempo, un fantasma salga del centro y se empiece a mover. Puedes usar dos temporizadores, uno para controlar el tiempo que ha de pasar hasta que salga el fantasma y otro para controlar el movimiento del mismo.

Para mover el fantasma, tendrás que almacenar su posición en el GameState. El patrón de movimiento puede ser algo tan sencillo como seguir en la dirección que va hasta que se encuentre con una pared, entonces cambiará la dirección de forma aleatoria a alguna de las posibles.

Ejercicio 3

Crea la lógica para que si el fantasma alcanza a Pacman, se acabe la partida.

Ejercicio 4

Crea la lógica para que si *Pacman* se come una pastilla o una cereza, entonces ahora *Pacman* podrá alcanzar a los fantasmas y hacerlos volver a la parte central, obteniendo un plus de puntos. Los fantasmas se moverán más deprisa y cambiarán de aspecto. Este efecto durará un determinado número de segundos. Usa temporizadores para implementar la lógica necesaria.

Ejercicio 5

Modifica el código para que exista un máximo de cuatro fantasmas al mismo tiempo.

Evaluación

- 4 puntos: Realizar el ejercicio 1.
- 6 puntos: Realizar, además, los ejercicios 2 y 3.
- 8 puntos: Realizar, además, el ejercicio 4.
- 10 puntos: Realizar todos los ejercicios.

Grado en Diseño y desarrollo de videojuegos

VJ1214 Consolas y dispositivos de videojuegos

Laboratorio Sesiones 10 y 11

Introducción

En estas dos sesiones, se proponen una serie de ejercicios para realizar animaciones. Se trata del mismo proyecto, pero en cada sesión se realizan cuatro ejercicios.

La pantalla estará dividida en dos partes (ver *Figura 1*). La parte superior serán las primeras 10 filas y representará un cielo donde se mueven unas nubes. En la parte inferior (las 14 filas restantes) mostrará el mundo del juego y al personaje principal.

En total se tendrán que realizarán las siguientes animaciones:

1. Animar dos tipos de nubes en la parte superior de la pantalla.
2. Animar el movimiento de un personaje.
3. Crear una animación “idle” para cuando el personaje lleva un tiempo sin moverse.
4. Realizar un scroll horizontal de la parte inferior de la pantalla.

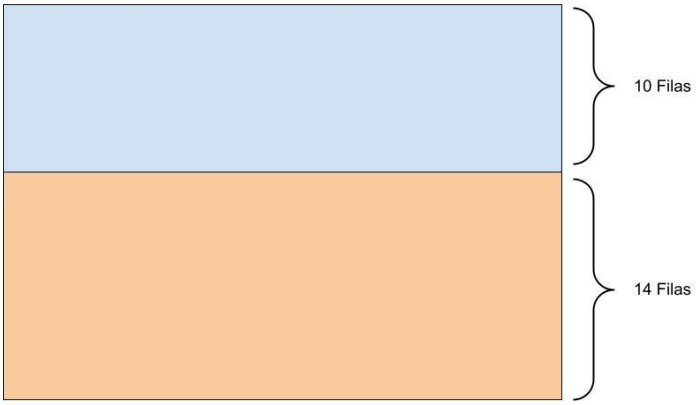

Figura 1: La pantalla está dividida en dos partes.

Ejercicio 1

Crea un objeto nube grande con 2x2 teselas. Las nubes grandes se moverán de izquierda a derecha por la parte superior de la pantalla (las primeras 10 filas). Deberá existir varias nubes grandes moviéndose por la pantalla. Usa un temporizador T1 para animar las nubes grandes.

Ejercicio 2

Crea un objeto nube pequeña con 1x2 teselas. Las nubes pequeñas se moverán de derecha a izquierda por la parte superior de la pantalla (las primeras 10 filas). Deberá existir varias nubes pequeñas moviéndose por la pantalla. Usa un temporizador T2 para animar las nubes pequeñas. La velocidad de animación de las nubes pequeñas tendrá que ser diferente a la de las nubes grandes.

Ejercicio 3

Crea las teselas necesarias para crear el mundo que se situará en la parte inferior de la pantalla (últimas 14 filas). No es necesario hacer un mundo que sea una referencia en el mundo del pixel art, pero si que tenga algunos elementos reconocibles.

Crea también un personaje de 2x2 teselas. Tendrás que crear dos versiones en cada dirección. Lee el ejercicio 4 antes de empezar a diseñar el personaje.

Ejercicio 4

Anima el movimiento del personaje. Para ello, simplemente tienes que cambiar la versión del personaje cuando pulsas una tecla. Por ejemplo, si el personaje está orientado mirando hacia la derecha y está mostrando una versión, al pulsar la tecla derecha pasará a mostrar la otra versión del personaje (mirando hacia la derecha).

Ejercicio 5

Realiza una animación “IDLE” que se ejecute cuando pase un tiempo desde que no se ha movido el personaje. La animación se ejecutará de forma interrumpida mientras no se pulse una tecla. En ese momento, se detendrá. Pasado un tiempo (desde la última pulsación de una tecla), la animación se volverá a repetir.

Usa un temporizador (T3) que se pondrá a ON al inicio del programa. Cuando, pasado un tiempo, se dispare T3, se ejecutará una función que pondrá en marcha la animación. Para ello, pondrá en OFF a T3 e iniciará otro temporizador T4. La función del temporizador T4 irá cambiando entre varias versiones del personaje de forma continua. Por lo tanto, tendrás que diseñar las nuevas versiones del personaje para esta animación.

Para realizar la animación, puedes usar una máquina de estados. Por ejemplo, vamos a ver un pseudocódigo que anima el personaje con tres versiones:

```
Programa principal
     Activar T3 (tiempo grande)

Función temporizador T3
```

```
      Desactivar T3
      Activar T4   (tiempo pequeño)
      estado <- 0

Función temporizador T4
      si estado == 0
            Mostrar versión personaje idle0
            estado <- 1
      sino si estado == 1
            Mostrar versión personaje idle1
            estado <- 2
      sino si estado == 2
            Mostrar versión personaje idle2
            estado <- 0
```

Ejercicio 6

Fusiona el código que has realizado para animar el personaje cuando se mueve (ejercicio 4), con el código para la animación IDLE (ejercicio 5). Para ello, cuando se pulse una tecla, T4 se pondrá en OFF (para que deje de animar el personaje con la animación IDLE) se moverá al personaje y se volverá a poner T3 a ON.

El siguiente pseudocódigo muestra cómo hacerlo:

```
Función tecla “->” ó “<-” pulsada
      Desactivar T3
      Desactivar T4
      Mover el personaje, mostrando la versión correspondiente.
      Activar T3
```

Ejercicio 7

En los dos siguientes ejercicios vas a programar un scrolling horizontal.

Crea un mundo de, al menos 64 columnas (tendrá que tener 14 filas), para la parte inferior de la pantalla. De nuevo, no es necesario hacer un mundo que sea una referencia en el mundo del pixel art.

Ejercicio 8

Implementa el scrolling horizontal. El personaje estará siempre en el medio de la pantalla de la NDS. Cuando pulses la tecla derecha o izquierda, tendrás que copiar la parte que corresponda del mundo en la memoria de pantalla para producir el efecto de que se mueve el personaje por el mundo (ver *Figura 2*).

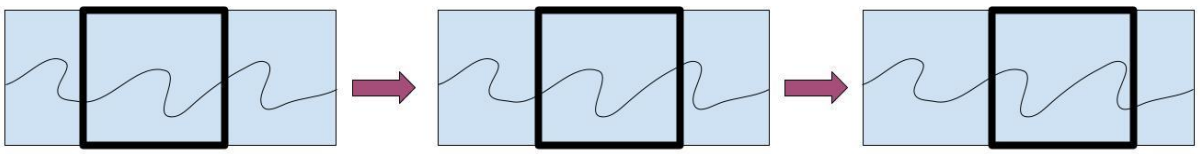

Figura 2: Scrolling horizontal. El rectángulo de línea gruesa representa la pantalla.

Puedes tener una variable que controle la columna de referencia en el mundo creado. Por ejemplo, si esa variable vale 12, entonces se mostrarán en la pantalla las columnas de la 12 a la 12+32 del mundo. Al pulsar la tecla derecha, la variable valdrá 13 y se mostrará de la 13 a la 13+32.

Grado en Diseño y desarrollo de videojuegos

VJ1214 Consolas y dispositivos de videojuegos

Laboratorio Sesión 12: Sonido

Introducción

En esta sesión vamos a ver cómo añadir sonido a los juegos NDS. Para ello, usaremos la librería ***maxmod9*** incluida en el conjunto de herramientas *DevkitPro*.

Hay dos tipos de sonidos principales. El primer tipo son las canciones que se reproducen mientras juegas. En este caso, hay varias posibilidades, pero lo que funciona mejor es que sean archivos ***mod***, que es un formato muy comprimido e ideal para ser usado en la NDS. Puedes encontrar muchas canciones libres en este formato en la web https://modarchive.org/.

El segundo tipo son los efectos de sonido que se reproducen cuando ocurre algún evento. En este caso, lo mejor es que sean archivos ***wav***. Los sonidos que uses deben estar muy comprimidos para poder ser ejecutados en la NDS. En esta web encontrarás un compresor de ***wav*** que te será de mucha utilidad https://www.freeconvert.com/wav-compressor/.

Ejercicio 1

En **devkitpro/examples/nds/audio/maxmod/basic_sound** puedes encontrar un ejemplo muy útil donde se reproduce una canción y dos efectos de sonido al pulsar dos teclas.

Compila el programa y prueba su funcionamiento.

Fíjate que el fichero ***Makefile*** que se usa para crear el ejecutable NDS, no es el mismo que el que has estado usando hasta ahora. Deberás usar el nuevo si quieres incluir sonidos en tu juego.

Ejercicio 2

Modifica el programa anterior para cambiar la música de fondo y los efectos de sonido.

Los pasos a seguir son:

1. Descargar de Internet una canción en formato MOD y dos sonidos en formato WAV. Recuerda que los WAV tienen que estar muy comprimidos. Llama a la canción, ***musica.mod*** y a los efectos ***efecto1.wav*** y ***efecto2.wav***.

2. Copia los ficheros en la carpeta ***maxmod_data***. No borres los ficheros que hay en esa carpeta.

3. Ejecuta make clean y luego make. Si todo ha ido bien es porque las canciones y sonidos son correctos para la NDS. Si no es el caso, prueba con otros ficheros ***MOD*** y ***WAV*** o comprimir más los ***WAV***. Ten un poco de paciencia en este punto, pues la NDS es muy sensible con respecto a qué sonidos puede ejecutar y que no.

4. Abre en la carpeta ***build*** el fichero ***soundbank.h***. Verás que, al hacer make, se han incluido las definiciones: ***MOD_MUSICA***, ***SFX_EFECTO1*** y ***SFX_EFECTO2***. Esto lo hace la herramienta ***mmutil***.

5. Ahora hay que editar el programa principal ***MadModExample.c***

    a. Usa la función ***mmLoad*** para cargar la canción ***MOD_MUSICA***.

    b. Usa la función ***mmLoadEffect*** para cargar los efectos ***SFX_EFECTO1*** y ***SFX_EFECTO2***.

    c. Cambia la función ***mmStart*** para que ejecute tu música ***MOD_MUSICA***.

    d. Crea una variable del tipo ***mm_sound_effect*** por efecto. Fíjate en los ejemplos.

    e. Cambia el código del bucle principal del programa para que al pulsar las teclas A y B, suenen los efectos ***SFX_EFECTO1*** y ***SFX_EFECTO2***, en vez de los originales.

Ejercicio 3

Partiendo del programa anterior, crea otro programa que muestre un menú de varias canciones y con las teclas UP y DOWN permita seleccionar qué canción queremos reproducir. La tecla A reproducirá la canción.

Ejercicio 4

Añade sonido al juego Pacman que creaste en las prácticas anteriores. Debe sonar una canción durante la partida. Cuando la partida se acabe se cambiará la canción. Cada vez que se pulsa una tecla, deberá sonar un efecto de sonido.

Evaluación

- Ejercicio 1 (4 puntos)
- Ejercicio 2 (6 puntos)
- Ejercicio 3 (8 puntos)
- Ejercicio 4 (10 puntos)

Grado en diseño y desarrollo de videojuegos

VJ1214 - Consolas y dispositivos

Sesión 13 de laboratorio

2024

Nombre	
DNI	
Firma	

Nota obtenida:

Instrucciones

- El objetivo de esta prueba es validar los conocimientos adquiridos durante las sesiones de laboratorio.
- Se podrá obtener una nota de 0 a 10.
- El tiempo máximo para la realización del ejercicio es de **2 horas**.
- El ejercicio es **individual**. Se puede consultar toda la documentación que se quiera exceptuando las respuestas a esta misma prueba realizadas por un compañero/a. Se permitirá el acceso a Internet para consultar documentación, pero no el intercambio de ficheros con las soluciones entre compañeros/as. Tampoco se admite el acceso a generadores de contenido estilo ChatGPT.
- Si se detectan dos o más soluciones con demasiadas coincidencias, todos los implicados obtendrán una nota de 0.
- La solución al ejercicio propuesto se deberá entregar en un zip en la tarea del aula virtual. Se deberá incluir el proyecto fuente y el fichero NDS creado. El nombre del fichero zip deberá ser **Apellido1Apellido2Nombre.zip**.

Enunciado

Partiendo del proyecto del Aula Virtual realiza las siguientes tareas:

1. **(0,5 puntos)** En la pantalla superior debe aparecer en una posición centrada, tanto horizontal como verticalmente, tus apellidos y nombre, tu dni y la fecha de la prueba.

2. **(0,5 puntos)** Modifica el color del avión.

3. **(3 puntos)** Inicialmente el avión se desplaza por la pista de despegue. Una vez pulsado el botón A comienza el despegue.

4. **(1 punto)** Se debe emplear una tesela diferente para el despegue del avión.

5. **(2 puntos)** Transcurridos unos segundos el avión se pondrá en posición de crucero y se parará cuando llegue al límite izquierdo de la pantalla.

6. **(2 puntos)** Una vez en dicha posición pulsando el botón B se inicia de nuevo el proceso de despegue.

7. **(1 punto)** Una vez haya despegado el avión no tendrán efecto ninguno de los botones A y B.

Aclaraciones:

1. El control del tiempo se deberá hacer mediante interrupciones.

2. En la pantalla inferior se deberá usar el modo teselado. Por lo tanto, no se permite el uso del modo framebuffer.

3. En el aula virtual se encuentra el fichero NDS **examen_laboratorio.nds** que muestra el funcionamiento solicitado.