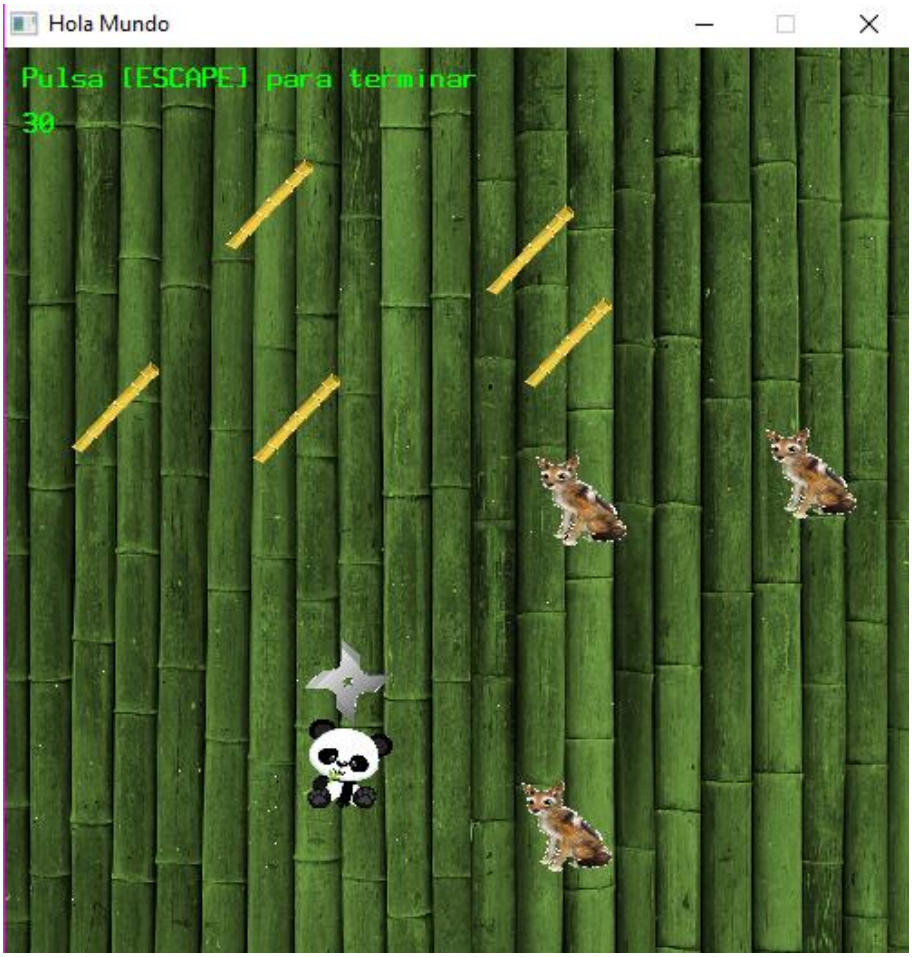
PROYECTO DE PROGRAMACIÓN

INFORME



Apellidos: Pérez-Ródenas Martínez

Nombre: Elena

Asignatura: Tecnología de la Programación

Grupo: 3.1 (PCEO)

Convocatoria: Junio

Año académico: 2017-2018

Profesor: Fernando Jiménez Barrionuevo

ÍNDICE

1. Descripción de la aplicación

1.1. Objetivos

El objetivo del juego es conseguir el mayor número de puntos cogiendo bambús y aguantando sin chocar con los chacales ayudándose de los shurikens que se pueden lanzar a los enemigos.

Por otro lado, en la opción 1 vs 1 el objetivo es aguantar más tiempo que el otro jugador sin chocarse con los chacales.

1.2. Descripción del funcionamiento general de la aplicación

La aplicación consta de un menú con 4 opciones a las cuales se puede ir accediendo con las distintas teclas según se va indicando en el propio juego.

Hay una opción de ayuda con las instrucciones y dos modos de juego (individual y multijugador).

Cuando se acaba la partida te da la opción de volver al menú principal o de salir.

1.3. Descripción del desarrollo de la aplicación

El desarrollo de la aplicación ha sido parte durante las prácticas y parte en casa y está basado en la aplicación de ejemplo mostrada por el profesor pero con bastantes mejoras y cambios. Está programado con CodeBlocks y documentado con Doxygen y las imágenes editadas con Photoshop.

2. Manual de usuario

2.1. Instrucciones para utilizar la aplicación

La aplicación consta de un menú principal con 4 opciones:

Una para jugar donde eres un panda y tienes que huir de los chacales. Conforme pasa el tiempo aumenta la puntuación y cada bambú que coges incrementa la puntuación 10 puntos. Si pulsas la tecla [ESPACIO] se disparan hacia arriba shurikens que al colisionar con los enemigos los hacen desaparecer haciendo más fácil aguantar más tiempo. Mientras dura la partida existe la posibilidad de salir pulsando la tecla [ESCAPE] que te lleva a otra página. Tanto si el personaje colisiona con el enemigo como si se pulsa esta tecla aparece una pantalla de despedida que muestra la puntuación obtenida y el récord de todas las partidas jugadas hasta ese momento. Entonces te da la posibilidad de pulsar 4 y volver al menú principal o 5 y salir de la aplicación.

En la opción de ayuda del menú hay una imagen con las instrucciones básicas de los dos modos de juego y los controles. En el modo multijugador el jugador 1 (el panda sin lazo) se mueve con las flechas mientras que el jugador 2 (el panda con lazo) se mueve con las letras W (arriba), A (izquierda), S (abajo) y D (derecha).

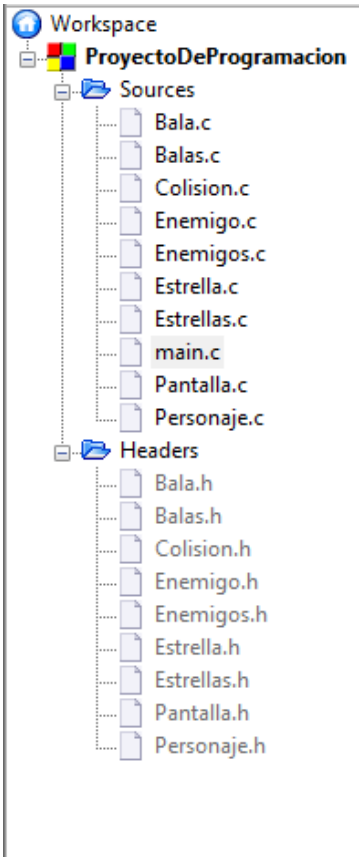
En el modo 1 vs 1 el objetivo es simplemente no morir antes que el otro. En el momento en que uno de los jugadores colisione con un enemigo la partida se acaba y se muestra en pantalla el panda ganador con un mensaje de enhorabuena y la opción de pulsar 4 y volver al menú principal o pulsar 5 y salir de la aplicación. Por el contrario, si durante la partida se pulsa [ESCAPE], aparece la misma pantalla pero sin el mensaje de enhorabuena con el jugador ganador.

Por último, la opción salir simplemente sale de la aplicación.

3. Organización del proyecto

3.1. Descripción de los ficheros que componen la aplicación

Consta de 9 ficheros de módulo .c con sus correspondientes ficheros de cabecera .h además de un main.c que contiene el cuerpo principal del programa.



- Bala (.c y .h): son módulos necesarios para el tratamiento de una bala que constan de un renombrado de tipos; una función para crear una bala que devuelve la bala; un procedimiento que libera una bala; otra función que mueve la bala; un procedimiento que dibuja la bala en pantalla; funciones para obtener la anchura, altura y coordenadas de la bala.
- Balas (.c y .h): son módulos para tratar balas. Consta de un renombrado de tipos; una función para crear balas; un procedimiento para liberarlas, otro para moverlas y otro para dibujarlas; otro para insertar una nueva bala; una función que comprueba si alguna bala ha colisionado; y un procedimiento que libera la bala si colisiona con un enemigo.
- Colisión (.c y .h): son módulos para el tratamiento de colisiones. Consta de una función que comprueba si se ha producido una colisión entre dos objetos.
- Enemigo (.c y .h): son módulos necesarios para el tratamiento de un enemigo y se estructuran de la misma forma que bala.
- Enemigos (.c y .h): son módulos necesarios para el tratamiento de enemigos y se estructura igual que balas pero sin el procedimiento de liberar las balas si colisionan con el enemigo.
- Estrella (.c y .h): son módulos para tratar una estrella con funciones y procedimientos para crearla, dibujarla, liberarla y obtener sus dimensiones y coordenadas.
- Estrellas (.c y .h): son módulos para tratar estrellas que añade un procedimiento para insertar una nueva estrella y una función para comprobar si colisiona.
- Pantalla (.c y .h): módulos proporcionados por el profesor que permiten realizar programas interactivos que usan gráficos sin necesidad de conocer los detalles de la visualización gráfica ni de la programación dirigida por eventos.
- Personaje (.c y .h): módulos para el tratamiento de un personaje que permiten crearlo, liberarlo, moverlo, dibujarlo y obtener sus dimensiones y coordenadas.
- Main.c: contiene el código principal del juego.

3.2. Relaciones y dependencias entre los distintos módulos y otros aspectos relevantes

El main incluye todos los ficheros .h del proyecto ya que llama a funciones de todos ellos.

Además los .c incluyen sus respectivos .h.

Otras dependencias:

Balas.c incluye ‘’Colision.h’’ y ‘’Enemigos.h’’.

Enemigos.c incluye ‘’Colision.h’’.

Estrella.c incluye ‘’Pantalla.h’’.

Estrellas.c incluye ‘’Colision.h’’.

Además, todos incluyen la librería <stdlib.h> y Pantalla.c incluye además <signal.h>, <stdio.h> y <math.h>.

Bala.h incluye ‘’Pantalla.h’’.

Balas.h incluye ‘’Bala.h’’ y ‘’Enemigos.h’’.

Enemigo.h incluye ‘’Pantalla.h’’.

Enemigos.h incluye ‘’Enemigo.h’’.

Estrella.h incluye ‘’Pantalla.h’’.

Estrellas.h incluye ‘’Estrella.h’’.

Personaje.h incluye ‘’Pantalla.h’’

Pantalla.h incluye ‘’SDL2/SDL.h’’.

La razón por la que se incluyen los distintos módulos unos en otros es porque en algunas de sus funciones y procedimientos se llama a otras que pertenecen a módulos distintos.

4. Estructuras de datos

4.1. Descripción de las estructuras de datos y Tipos Abstractos de Datos diseñados e implementados para el proyecto

En el proyecto existen las siguientes estructuras de datos:

- BalaRep: está formado por una imagen i, una anchura w, una altura h, unas coordenadas x e y y una velocidad v.

```
struct BalaRep
{
    Imagen i;
    int w, h, x, y, v;
};
```

- BalasRep: está formado por una Bala b de tipo BalaRep y Balas sig de tipo BalasRep.

```
struct BalasRep
{
    Bala b;
    Balas sig;
};
```

- EnemigoRep: está formado por una imagen i, una anchura w, una altura h, unas coordenadas x e y y coordenadas de la velocidad vx y vy.

```
struct EnemigoRep
{
    Imagen i;
    int w, h, x, y, vx, vy;
};
```

- EnemigosRep: está formado por un enemigo e de tipo Enemigo (apuntador de EnemigoRep) y por enemigos sig de tipo Enemigos (apuntador de EnemigosRep).

```
struct EnemigosRep
{
    Enemigo e;
    Enemigos sig;
};
```

- EstrellaRep: está formado por una imagen i, una anchura w, una altura h y unas coordenadas x e y.

```
struct EstrellaRep
{
    Imagen i;
    int w, h, x, y;
};
```

- EstrellasRep: está formado por un apuntador a Estrella e (apuntador de EstrellaRep), un entero que indica el máximo de estrellas max y otro entero n que indica el número de estrellas que hay.

```
struct EstrellasRep
{
    Estrella * e;
    int max,n;
};
```

- PersonajeRep: está formado por una imagen i, una anchura w, una altura h y unas coordenadas x e y.

```
struct PersonajeRep
{
    Imagen i;
    int w, h, x, y;
};
```

En cuanto a los TDAs, nos permiten utilizar tanto estructuras de datos como funciones y procedimientos entre módulos. Además, se tiene una mayor privacidad del código ya que los usuarios del TDA no conocen la representación de sus valores de memoria proporcionando así una mayor protección.

Para manejar el TDA, el usuario no necesita conocer los detalles de la representación o la forma en que se realizan las operaciones. Sólo es necesario mirar la documentación realizada con Doxygen que muestra para qué sirve cada cosa.

5. Conclusiones

5.1. Dificultades encontradas y forma de superarlas

La primera dificultad encontrada fue la de conseguir que los shurikens se dispararan como yo quería y que desaparecieran al chocar contra los enemigos pero observando un poco las otras funciones tanto de liberación como de colisión no fue muy difícil conseguirlo. Por otro lado, para que el menú funcionara correctamente, tuve problemas con las teclas pulsadas y el paso de unas ventanas a otras. Sin embargo, con el uso de las flags explicadas por el profesor se solucionó. También tuve problemas con algunos whiles pues no contemplé la condición de que la pantalla siguiera activa y, por último, tuve que estructurar mejor el código para que fuera más legible y corregir algún Pantalla_Actualiza que estaba en el lugar equivocado.

5.2. Grado de satisfacción con la asignatura y sugerencias de mejora

Estoy bastante contenta con el desarrollo de esta asignatura, en especial con la parte de prácticas, pues hacer un juego es una idea muy buena para motivar a los alumnos a programar, investigar y conseguir solventar los problemas que van surgiendo durante el proceso de creación. Además, al estar programada para poder trabajar en clase, el tiempo dedicado era más ameno y las dudas podías resolverlas más fácilmente con la ayuda del profesor y los compañeros.

Sugeriría una lista de errores típicos o advertencias para que el tiempo en solucionarlos sea menor. Pero en general no tengo ninguna pega.