

Snake



Sergi villalobos gascón

Albert boguñá pozo

Clase A, grupo 2

Tabla de contenido

[1.Juego 3](#_Toc473582966)

[1.1. Selección del juego 3](#_Toc473582967)

[1.2. Adaptaciones, cambios o mejoras con respecto al juego original 3](#_Toc473582968)

[1.3. Pantallas 3](#_Toc473582969)

[2.Modelo: Game Objects & Scenes 3](#_Toc473582970)

[2.1 Game Objects y escenas 3](#_Toc473582971)

[➔ Diagrama de clases 3](#_Toc473582972)

[➔ Explicación breve de las estructuras de datos usadas y el porqué 3](#_Toc473582973)

[2.2 Diseño del fichero XML de configuración del juego 3](#_Toc473582974)

[2.3 Uso y/o modificación de la clase SceneManager 3](#_Toc473582975)

[3. Controlador: Game 4](#_Toc473582976)

[3.1 Game. Atributos y métodos 4](#_Toc473582977)

[3.2 Eventos 4](#_Toc473582978)

[3.3 Diagrama de clases 4](#_Toc473582980)

[3.4 Uso y/o modificación de la clase InputManager 5](#_Toc473582981)

[4. Vista: Renderer 5](#_Toc473582982)

[➔ Uso y/o modificación de la clase Renderer 5](#_Toc473582983)

[5. Diagrama de clases de todo el proyecto 5](#_Toc473582984)

[6. Deployment 5](#_Toc473582985)

[7. Estimación de tiempo​ 6](#_Toc473582986)

[8. Conclusiones 6](#_Toc473582987)

[➔ Lecciones aprendidas 6](#_Toc473582988)

[➔ Satisfacción 6](#_Toc473582989)

[➔ Qué fue lo más difícil de entender y/o de implementar 6](#_Toc473582990)

[➔ Fortalezas/debilidades de su software 6](#_Toc473582991)

[9. Referencias 7](#_Toc473582992)

# 1.Juego

## 1.1. Selección del juego

Hemos escogido el Snake ya que fue el primer juego que tuvimos en nuestros móviles y es con el que más familiarizados estamos, además, todo el trabajo de este semestre de todas las asignaturas ha sido muy concentrado y consideramos que éste era el más fácil de programar y el que nos ocuparía menos tiempo, ya que teníamos muchísimo trabajo.

## 1.2. Adaptaciones, cambios o mejoras con respecto al juego original

Ya que no somos artistas, en el apartado gráfico no hay mejora alguna a no ser que nos remontemos a los Snakes clásicos de píxeles negros o blancos, lo de implementar tres dificultades con sus respectivas variaciones respecto a la velocidad, los alimentos y el tamaño del tablero es una mejora respecto a otros. También hay que mencionar el ranking de puntuaciones como un punto a nuestro favor respecto al resto.

## 1.3. Pantallas

Disponemos de la pantalla de título, la cual permite acceder al juego o salir de la aplicación. En caso de elegir la opción de jugar, tendremos disponibles las tres dificultades (fácil, medio y difícil) y nos llevará directamente a la pantalla del juego tras elegir una. Al morir, tendremos la pantalla de ranking en el caso de haber superado una alta puntuación.

# 2.Modelo: Game Objects & Scenes

## 2.1 Game Objects y escenas

### ➔ Diagrama de clases

De Game Objects no hemos aprovechado clases del Candy Crush, las hemos creado nosotros: Manzana, Serpiente, Grid, Score, a lo que a Game Objects se refiere.

### ➔ Explicación breve de las estructuras de datos usadas y el porqué

Struct es una estructura de datos a la que hemos recurrido bastante para almacenar las variables. El resto de clases están formadas por variables del tipo int y char, con un vector de sprites para crear la serpiente.

## 2.2 Diseño del fichero XML de configuración del juego

El fichero XML contiene la configuración de las tres dificultades del juego, dentro de la etiqueta “variables”. Como ejemplo pondremos la configuración “fácil”:

<easy>

<rows>25</rows> //número de filas del tablero.

<columns>25</columns> //número de columnas del tablero.

<time>120</time> //Tiempo para completar el nivel en segundos.

<speed>3</speed> //Velocidad del movimiento de la serpiente.

<initial>5</initial> //Número de alimentos iniciales por nivel.

<incremental>2</incremental> //Número de alimentos incrementales.

</easy>

## 2.3 Uso y/o modificación de la clase SceneManager

El Scene Manager consta de un método que se ejecuta una única vez al principio, en el que se crea el tablero, se coloca la serpiente y el alimento.

Otro método que también se ejecuta una única vez es el de la salida de la escena, donde se destruye todo lo de la escena para liberar la memoria que ocupaba.

El siguiente método, es el que hace el loop que comprueba si la serpiente se come un alimento o choca contra algo y llama a los métodos pertinentes para actualizar correctamente el juego, como perder una vida o hacer crecer la serpiente.

Por último también está el método que se encarga de printar los elementos en pantalla, como el tablero, la serpiente, la puntuación, etc. Tmbién funciona en loop.

# 3. Controlador: Game

## 3.1 Game. Atributos y métodos

class GameScene : public Scene {

public:

Grid drawGrid;

Snake snake;

Manzana manzana;

Score score;

Printar printar;

int columnas;

int filas;

bool start;

explicit GameScene();

~GameScene();

int GiveScore(void);

void OnEntry(void) override;

void OnExit(void) override;

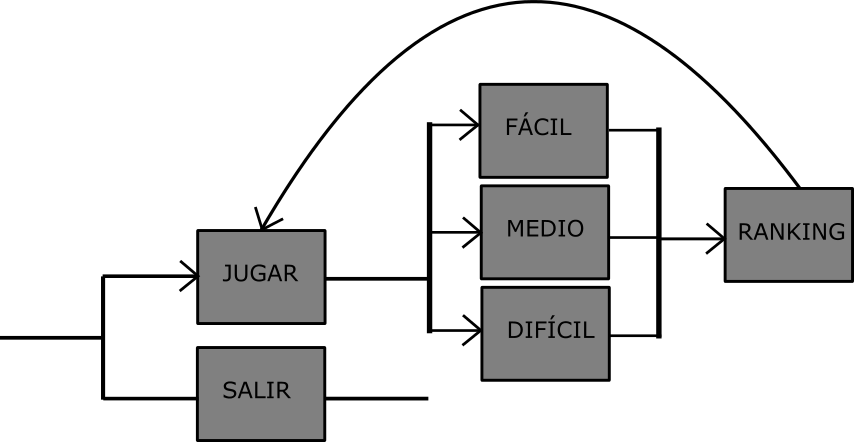
void Update(void) override;

void Draw(void) override;

};

## 3.2 Eventos

## 



## 3.3 Diagrama de clases

Hemos utilizado clases como el renderer el input manager, entre otras del Candy Crush para el proyecto, las que hemos creado nosotros han sido: Niveles, Game Manager, Printar.

## 3.4 Uso y/o modificación de la clase InputManager

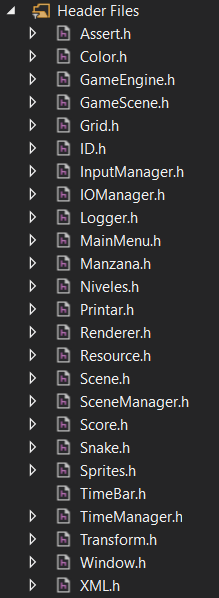
Utilizamos lo que había del Candy Crush, ya que está muy completo y cumple con creces lo que necesitamos.

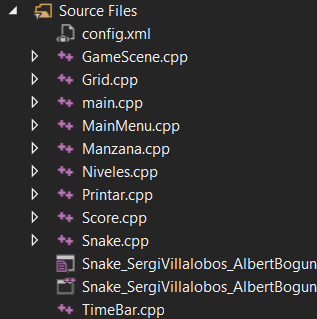
# 4. Vista: Renderer

## ➔ Uso y/o modificación de la clase Renderer

También hemos utilizado lo que ya estaba hecho del Candy Crush, ya que no tenemos experiencia en la programación de estos elementos tan complejos.

# 5. Diagrama de clases de todo el proyecto





# 6. Deployment

Se juega desde el mismo Visual Studio no hemos buscado cómo exportarlo fuera de él.

# 7. Estimación de tiempo​

Nos hemos dividido el trabajo a partes iguales, no hemos llevado el cálculo del tiempo invertido, pero hemos trabajado juntos de manera online y realizar el proyecto nos ha costado aproximadamente 30h.

La configuración XML, el cambio de escenas, el tablero, el movimiento de la serpiente, mostrar la puntuación y vidas por pantalla, la documentación del proyecto, recoger los valores del fichero XML, la colisión con los límites del tablero y las imágenes corren de parte de Sergi Villalobos Gascón.

Los niveles, la intercepción de alimentos, la colisión de la serpiente con su cuerpo, el ranking, los modificadores de velocidad, puntuación e incrementación de alimentos y los “checkpoints” al superar niveles corren de parte de Albert Boguñá Pozo.

# 8. Conclusiones

## ➔ Lecciones aprendidas

Hemos aprendido a programar en más profundidad en C++ y aprendido el modelo vista controlador y descubierto diversas funciones de la librería SDL, lo cual creemos que era el objetivo del proyecto.

## ➔ Satisfacción

**Sergi:** Muy escasa, de estar programando en Unity a C++ se nota mucho el salto, es mucho más primitivo y costoso, ha sido muy estresante y agobiante, entiendo que para formarse como programador hay que pasar por esto, y todo lo que se aprenda siempre es bienvenido. Hemos tenido tiempo de sobra para realizar el proyecto, pero en mi opinión ha faltado tutelaje, ya que la mayoría de las cosas hemos tenido que buscarlas por internet.

## ➔ Qué fue lo más difícil de entender y/o de implementar

El fichero binario para guardar y mostrar las puntuaciones, modificar la velocidad de la serpiente y mostrar por pantalla la puntuación y las vidas es lo que más nos ha costado a la hora de implementar y entender el funcionamiento, sobre todo lo relacionado con SDL.

## ➔ Fortalezas/debilidades de su software

El código funciona correctamente y el juego es jugable, lo cual es obligatorio pero a su vez es una fortaleza. Tratando los temas de optimización, ni de lejos es el código más óptimo y mejor implementado, por lo que sería la principal debilidad. Otra fortaleza sería que los destructores funcionan correctamente para liberar la memoria y el programa no se colapse. Por último, el ranking no está acabado de implementar ya que trabajar con el fichero binario es algo que no hemos sabido aprender correctamente por nuestra cuenta.

# 9. Referencias

<https://www.libsdl.org>

<http://www.muylinux.com>

<http://www.cplusplus.com>

<http://stackoverflow.com>

<http://codereview.stackexchange.com>

<https://www.gamedev.net>