UOC

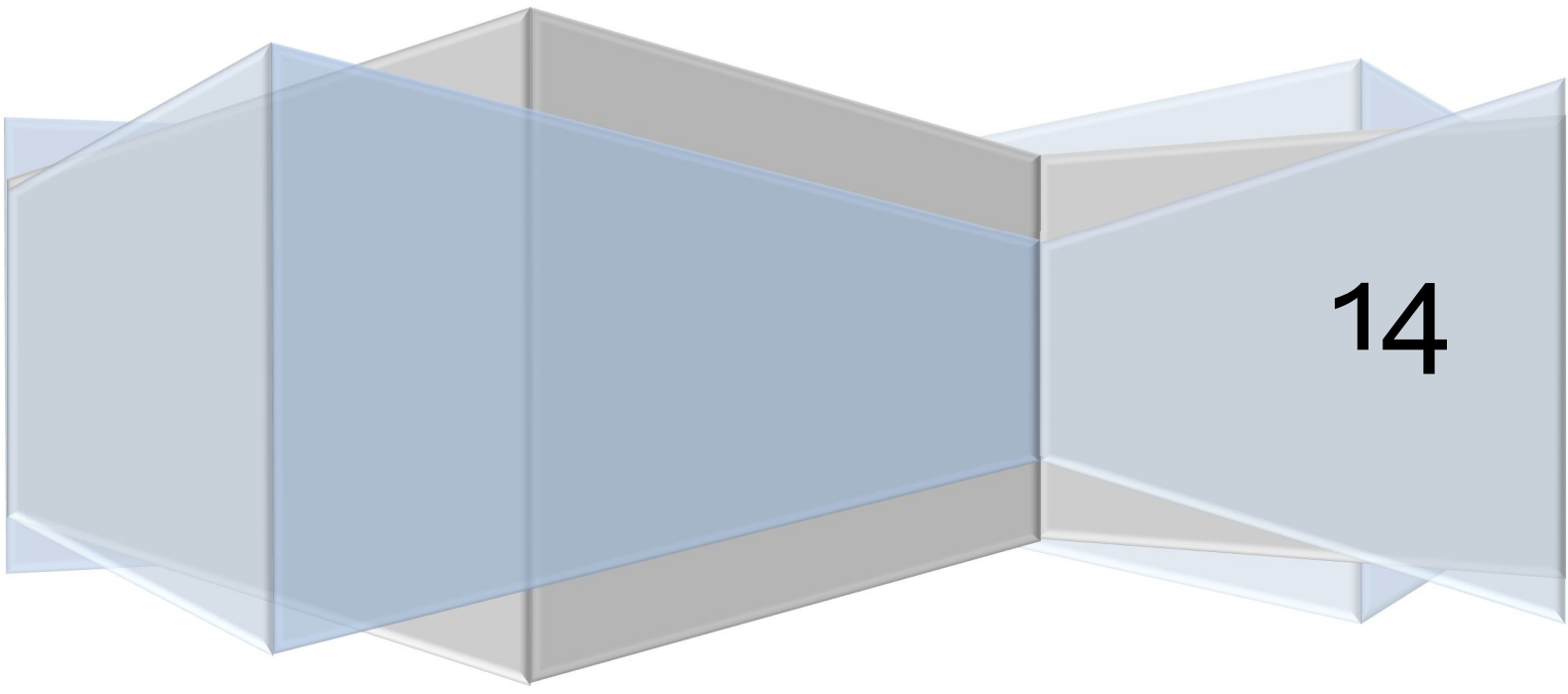
Trabajo de final de grado

Proyecto Snake ILM

David Mirón López

Consultor:

David Gañan Jiménez



14

Agradecimientos

A mi madre y mi padre, por su paciencia y por facilitarme todos los medios para mi desarrollo académico.

A mi novia Cristina por escucharme cuando lo he necesitado.

TRABAJO DE FINAL DE GRADO
[MEMORIA]

Título del trabajo:	*Snake ILM*
Nombre del autor:	*David Mirón López*
Nombre del consultor:	*David Gañan Jiménez*
Data de entrega (mm/aaaa):	*06/2014*
Área del Trabajo Final:	*Desarrollo de software*
Titulación:	*Grado en Ingeniería Informática*

Resumen del Trabajo:

Trabajo de final de carrera del grado de Ingeniería Informática centrado en la creación de un juego para plataformas móviles Windows Phone y bajo el framework llamado XNA.

El proyecto se encuentra divido en diferentes fases siendo la primera de ellas la planificación de los tiempos de las diferentes fases y tareas en las que se divide el proyecto.

En la segunda fase se analizan los diferentes requerimientos, extrayendo de ellos posteriormente los casos de uso en los que se va a dividir el desarrollo. Esta segunda fase incluye también las explicaciones pertinentes a la infraestructura en que esta basado XNA y un prototipo de la interfaz gráfica del proyecto.

La tercera fase del proyecto está basada plenamente en la implementación y documentación de las pruebas realizadas para asegurar una calidad óptima del producto final. Se incluyen también el manual de instrucciones de la aplicación y el manual de instalación de la misma.

La cuarta y última fase se centra en el redactado de los apartados finales de la memoria y en la elaboración de la presentación final del proyecto.

Abstract (in English, 250 words or less):

Working limit the degree of Computer Engineering focused on creating a game for Windows Phone mobile platforms and under the framework XNA.

The project is divided into different phases. The first of them is based on planning the times of the different phases and tasks in which the project is divided .

In the second phase we analyze the different requirements, extracting from them subsequently use cases in which the development will be divided . This second phase also includes infrastructure explanations that XNA is based on and a prototype project GUI.

The third phase of the project is fully based on the implementation and documentation of tests to ensure optimum product quality. The instruction manual of the application and the installation manual of it are also included.

The fourth and final phase focuses on the edition of the final sections of the report and the preparation of the final project presentation.

Palabras clave (entre 4 i 8):

XNA, Snake, Desarrollo, TFG, Juego

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

1.1.1. Situación actual

Hoy en día no hace falta nombrar que la competencia de las diferentes compañías encargadas de proveer sistemas operativos en telefonía móvil, las propias empresas fabricantes y proveedoras de terminales así como la incursión de los Smarthpones y tabletas dentro del mercado está facilitando el aumento de las descargas y uso de las aplicaciones disponibles para móviles y también el incremento de la cantidad de estas. Gracias a este hecho entre otros muchos desarrolladores están empezando ahora a programar aplicaciones ya sea para Android, Windows Phone o IOS.

Según estadísticas de las empresas Flurry Analytics y Statista, en el año 2013 el uso de las aplicaciones móviles aumento un 115 por ciento y no se parará sino que según diferentes estudios la tasa de descargas aumentará en un 30% anualmente hasta el año 2017. Entre otras cosas, esto es debido a la gran accesibilidad de los juegos para móvil, a través de plataformas como iTunes, GooglePlay o el mismo Windows Market, a su precio asequible incluso gratis y también a nuevas formas de negocio aparecidas en los últimos tiempos cómo son los *micropagos*.

Concretamente, nos encontramos ante el éxito que están teniendo muchas compañías hasta hace nada desconocidas como *Rovio,* creador de la saga Angry Birds o King, creador de Farm Heroes Saga, Candy Crush entre otros. Esto está alentando a muchos desarrolladores con o sin experiencia a emprender el camino hacia el desarrollo móvil e incluso montar empresas dedicadas a este negocio de futuro.

Profundizando en el desarrollo de juegos para móvil y obviando el detalle del sistema operativo móvil o plataforma elegida, es importante saber y entender que este tipo de juegos tienen la particularidad de estar destinados para cualquier tipo de público, debido a que cualquiera puede tener un terminal móvil, desde el típico estudiante de la ESO, hasta personal de dirección de empresas.

Teniendo en cuenta lo anterior, no solo es importante el público al que va dirigido un juego para teléfono móvil, sino que también debe tenerse en cuenta cuando un usuario va a jugar. El jugador de este tipo de juegos móviles puede jugar en cualquier momento y lugar, por lo tanto, necesita una aplicación ágil, con una interfaz directa y muy accesible debido a las reducidas dimensiones de pantallas de móviles o tabletas.

1.1.2. ¿Porque Windows Phone?

Llegados a este punto, nos encontramos que existen varias plataformas de desarrollo para telefonía móvil en la actualidad. Las más importantes, como la mayoría de desarrolladores de software saben, son iOS, Android y Windows Phone. Realizando una rápida comparativa podemos ver que Windows Phone supera a los otros dos en varios aspectos destacables como la potencia que ofrecen sus herramientas para desarrollar la interfaz gráfica ya sea con Silverlight mediante Expression Blend o con XNA Game Studio. Aparte de la potencia de sus herramientas, siendo Windows Phone obra de Microsoft, existe una enorme comunidad de desarrolladores y divulgadores tecnológicos que ahorran mucho trabajo a principiantes.

Estas ventajas se pueden ver reflejadas en el aumento de un 40% de las aplicaciones enviadas al Windows Market desde la salida de Windows Phone 8. En último lugar, el hecho de que se pueda desarrollar en C#, siendo mi caso una gran ventaja puesto que tengo muchos años de experiencia con este lenguaje de programación también es un factor a favor.

1.1.3. ¿XNA o Silvelight?

XNA Game Studio ofrece al desarrollador una serie de herramientas para facilitar el desarrollo de juegos, sobre todo a principiantes. Este *framework* abstrae al programador de tener que crear pantallas, implementar funciones de pausado de eventos entre otras cosas. A parte de las ayudas propias a la hora de implementar código, Microsoft también provee a los desarrolladores de una serie de Kits de Inicio como base para comenzar a implementar código inmediatamente. Uno de los

propósitos de Microsoft con XNA es ayudar al desarrollador y liberarle de ciertas tareas para que desde un primer momento este pueda dedicarse a programar código para el juego en sí. Partiendo de estas características, podemos concluir que XNA es la plataforma o framework adecuado para realizar el juego de Snake.

A partir de estas premisas, se considera que el juego de Snake que se va a desarrollar para la plataforma Microsoft Windows Phone en su versión 7 utilizando el framework XNA que provee Microsoft.

1.2. Objetivos del Trabajo

El objetivo principal del Trabajo de Final de Grado es desarrollar la aplicación Snake para plataformas móviles Windows Phone. Por otro lado, existen una serie de objetivos secundarios donde se pueden diferenciar claramente entre dos tipos.

Primeramente definiremos los objetivos a nivel de producto los cuales ya van predefinidos en lo que debería ser por definición el juego de Snake y que se listan a continuación:

- Planteamiento y justificación de la plataforma elegida.
- Planificación, análisis y diseño del producto.
- Diseño y prototipado de la interfaz gráfica.
- Implementación de las diferentes pantallas y lógica del juego.
- Testeo del producto obtenido.
- Generación de la documentación relativa al juego.

En un segundo lugar, existen ciertos objetivos que se quieren alcanzar y que no tienen que ver con el análisis, desarrollo o experiencia de juego final propiamente dichos. Estos son los siguientes:

- Conocer la plataforma para desarrollo de juegos Microsoft XNA.
- Aprender cómo funciona el ciclo de vida de un juego para Windows Phone.
- Desarrollo de código mantenible.

- Explorar las posibilidades ofrecidas por las aplicaciones/juegos desarrollados en Windows Phone.

1.3. Descripción del proyecto

En este proyecto de final de grado se llevarán a cabo las diferentes tareas hasta la consecución del producto final, el cual será la aplicación Snake para la plataforma móvil Windows Phone de Microsoft a través de la plataforma de desarrollo Microsoft XNA. En adelante, nos referiremos al juego con nombre Snake ILM, el cual también será su nombre definitivo.

En nuestra aplicación el usuario podrá crear nuevas partidas, consultar máximas puntuaciones y también configurar el juego para aumentar o disminuir la dificultad del mismo. Comentando el juego concretamente, se basa en el control de la cabeza de una serpiente que puede moverse en cuatro direcciones diferentes (arriba, abajo, derecha e izquierda) y debe ir recogiendo comida a lo largo del terreno de juego de forma rectangular el cual está delimitado por bordes. Según va recogiendo piezas de comida el tamaño de la serpiente aumenta y el juego se vuelve más rápido y por lo tanto más complicado. Si el jugador choca contra los bordes o contra alguna parte de la serpiente, el juego se da por finalizado. El objetivo principal es intentar conseguir la máxima puntuación, calculada en el número de piezas de comida que la serpiente ha conseguido comer.

Aparte del propio desarrollo del juego, existen muchas otras fases que deben realizarse antes.

Primeramente se llevará a cabo una descripción general del proyecto y la definición de los diferentes objetivos que se quieren conseguir al finalizar dicho proyecto. Junto con esto, se realizará una planificación de las diferentes tareas que se deberán llevar a

cabo y los hitos que se pretenden alcanzar, en este caso, marcados por las entregas de las diferentes PACs en las que está dividida la asignatura.

En una segunda fase se va a proceder a definir y analizar las diferentes funcionalidades a través de UML (casos de uso), también se realizará un diseño de la arquitectura a seguir durante la implementación del juego y un posterior prototipo que servirá de representación de las diferentes pantallas que podrá visualizar el usuario final tales como la pantalla de juego, configuración etc.

Una vez finalizada la implementación se documentarán las diferentes pruebas llevadas a cabo para evaluar la calidad del software. A parte de esto también se realizará un resumen del código fuente donde se intentarán resaltar los diferentes patrones, librerías, etc. que se han utilizado en el desarrollo.

Por último, para finalizar y cerrar la documentación, se añadirán apartados donde se evaluará el coste total en horas y dinero de todo el diseño, análisis, implementación y documentación del proyecto junto con un apartado donde se especificarán las posibles mejoras que se podrían llevar a cabo para mejorar y evolucionar el juego.

1.4. Planificación del Trabajo

A continuación se presenta la planificación de tareas inicial del proyecto de final de grado. Esta planificación previa se irá modificando con el tiempo una vez se tengan los análisis, funcionales y diseños de arquitectura necesarios entre otras cosas.

1.4.1. Planificación temporal

En la imagen inferior se puede ver la primera planificación del proyecto de final de carrera que servirá como punto de partida a las posteriores modificaciones que se vayan realizando. Esta planificación consta en un primer nivel de las tareas principales, y una serie de sub tareas que contiene cada una de estas cuatro tareas.

Actualmente se ha realizado una división de tareas genéricas de un tipo proyecto de desarrollo de software y el cual se verá modificado durante la segunda y tercera

entrega por un refinamiento de estas tareas para adaptarlo a un plan de proyecto más acorde con el desarrollo de un videojuego para plataforma móvil.

En la planificación de tareas siguiente podremos ver tanto el comienzo estimado de cada una de las tareas así como la duración de estas.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Planificación**	**8 días**	**jue 27/02/14**	**lun 10/03/14**	
2	Selección del proyecto	3 días	jue 27/02/14	sáb 01/03/14	
3	Instalación de programario	1 día	lun 03/03/14	lun 03/03/14	2
4	Definición de objetivos	2 días	mar 04/03/14	mié 05/03/14	3
5	Análisis de riesgos	2 días	jue 06/03/14	vie 07/03/14	4
6	Generación de hitos	1 día	sáb 08/03/14	sáb 08/03/14	5
7	Planificación del proyecto	2 días	dom 09/03/14	lun 10/03/14	6
8	**Análisis y diseño**	**20 días**	**mar 11/03/14**	**lun 07/04/14**	
9	Especificación de requerimientos	2 días	mar 11/03/14	mié 12/03/14	7
10	Casos de uso	4 días	jue 13/03/14	mar 18/03/14	9
11	Análisis funcional	5 días	mié 19/03/14	mar 25/03/14	10
12	Diseño de arquitectura	5 días	mié 26/03/14	mar 01/04/14	11
13	Creació del prototipo	4 días	mié 02/04/14	lun 07/04/14	12
14	**Implementación y test**	**35 días**	**mar 08/04/14**	**lun 26/05/14**	
15	Estudio SDK Windows Phon	3 días	mar 08/04/14	jue 10/04/14	13
16	Implementació de interfase gráfica	6 días	vie 11/04/14	vie 18/04/14	15
17	Implementació de components code behind	19 días	lun 21/04/14	jue 15/05/14	16
18	Pruebas de integración	3 días	vie 16/05/14	mar 20/05/14	17
19	Pruebas funcionales	4 días	mié 21/05/14	lun 26/05/14	18
20	**Entrega final**	**10 días**	**mar 27/05/14**	**sáb 07/06/14**	
21	Redactar memoria	6 días	mar 27/05/14	mar 03/06/14	19
22	Filmación de la presentació	4 días	mié 04/06/14	sáb 07/06/14	21

Figura 1

1.4.2. Diagrama de GANTT

En el diagrama de Gantt presentado en la siguiente figura podemos ver las diferentes tareas en una escala de tiempo.

Este diagrama se irá actualizando regularmente para mantener una visión actualizada sobre el progreso actual del proyecto y también para poder avanzar los posibles retrasos y posibles re planificaciones que el proyecto vaya a necesitar.

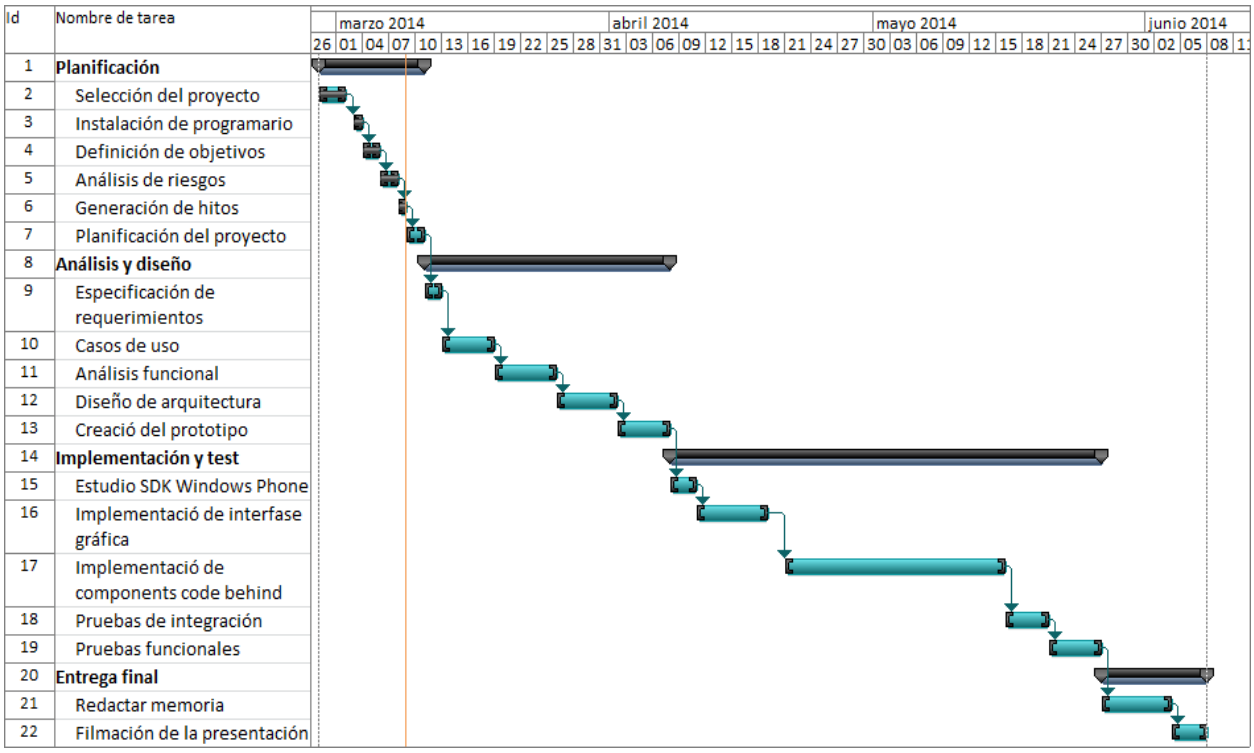


Figura 2

1.4.3. Hitos

Los hitos del proyecto de final de carrera vienen marcados por las diferentes entregas que se deben realizar a lo largo del semestre y que vienen definidas por la propia asignatura y propuesta de trabajo entregada por el profesorado.

Nombre de Hito	Fecha límite
Entrega de la Planificación del proyecto (PAC 1)	10/03/2014
Entrega del análisis y diseño (PAC 2)	07/04/2014
Entrega de la Implementación (PAC 3)	26/05/2014
Entrega de la memoria y presentación (PAC 4)	07/06/2014

1.5. Productos obtenidos

A partir del desarrollo de todo el proyecto se han obtenido una serie de productos bien diferenciados y que en este apartado se listan y describen:

- **Juego XNA**: Aplicación XNA y objetivo principal del proyecto de final de carrera. Se obtiene un archivo con extensión *ccgame* instalable en dispositivos móviles con sistema operativo Windows Phone.
- **Memoria**: Documentación del trabajo de final de carrera donde se realiza un compendio de toda la documentación obtenida durante el desarrollo. Esta documentación contiene desde la planificación del proyecto inicial, objetivos y requerimientos hasta la documentación final de la parte de pruebas de la aplicación.
- **Manual de instalación**: Se encuentran añadidas como anexo dentro de la memoria y explican al usuario final como debe instalar el juego y cuáles son los requerimientos necesarios para poder hacerlo tales como instalación de programas auxiliares.
- **Manual de usuario**: Manual explicativo donde se trata de mostrar al usuario las diferentes posibilidades que tiene el juego, para que sirven y desde donde se accede a todas ellas.

1.6. Capítulos de la memoria

En este apartado se va a realizar un pequeño resumen de los siguientes capítulos existentes dentro de este documento y la relación que estos mantienen con el proyecto global.

Análisis de riesgos

Se exponen los diferentes riesgos que pueden ocurrir en un proyecto de este categorizados según el ámbito de incidencia como la planificación, implementación o testeo de la solución final.

Análisis

En el análisis del proyecto se definen todos los requerimientos asociados a la solución y también todos los posibles casos de uso y se especifican estos

mediante fichas de casos de uso teniendo en cuenta los flujos principales de ejecución y los flujos secundarios o alternativos.

Diseño

En el diseño de la solución se exponen los diagramas de clases principales tales como el diagrama de clases en el que se basará la propia mecánica del juego y también el diagrama de clases basado en el kit de Inicio *Game State Management*. Junto con esto, se explican y se describen las principales funcionalidades de las clases expuestas dentro de los diagramas de clases.

Prototipo

Se presenta la interfaz de usuario desprovista todavía de funcionalidades pero donde se puede ver una aproximación realista de cómo será la aplicación que se instalará dentro de los dispositivos móviles

Implementación

En el apartado de implementación se explica de manera general como funciona el ciclo de vida de las aplicaciones XNA y también se comentan y explican los principales problemas y mediante que algoritmos se les ha dado solución.

Pruebas

Este apartado parte de la base expuesta dentro de los requerimientos y se explican las diferentes pruebas realizadas y que requerimientos se cumplen con cada una de ellas.

Valoración económica

El apartado de valoración económica presenta los roles necesarios que se hubiesen necesitado en el desarrollo del proyecto mediante un equipo de desarrollo y que valor económico tendría la solución en ese caso.

Futuras mejoras

Las futuras mejoras son las ideas obtenidas durante el desarrollo del producto que aportarían cierto valor añadido al juego obtenido.

Conclusiones

Último apartado del proyecto sin tener en cuenta la bibliografía, donde se explica si se han conseguido los objetivos y se ha seguido la planificación

propuesta en el apartado correspondiente y también que lecciones se han aprendido de este tipo de desarrollos para plataformas móviles.

2. Análisis de riesgos

Para realizar el análisis de riesgos, tendremos en cuenta que un riesgo en el proceso de desarrollo de software es cualquier evento que pueda ocurrir y que pueda tener efecto en cualquiera de las fases del proyecto y que impida o ponga en peligro alcanzar cualquiera de los objetivos marcados al principio de este. Para realizar un análisis de riesgos lo más exhaustivo introduciremos todos los riesgos asociados tanto a la planificación, análisis, desarrollo y testeo en una matriz de riesgos.

Dentro de esta matriz valoraremos los riesgos respecto a la probabilidad con la que podría ocurrir y respecto al impacto que tendría en el proceso del trabajo. Para categorizar los diferentes riesgos utilizaremos las categorías *Baja, Media y Alta*.

Riesgo	**Probabilidad**	**Impacto**
Planificación		
Omisión de tareas aparentemente triviales pueden ocasionar problemas y bloqueos de otras tareas que si son importantes.	Media	Alto
Mal dimensionamiento de tareas pueden ocasionar desviamientos en posteriores fases del Trabajo.	Alta	Medio
Hitos inalcanzables por mala planificación.	Baja	Alto
Alineamiento de las diferentes tareas del Trabajo a los objetivos funcionales marcados al principio.	Media	Medio
Orden incorrecto y dependencia incorrecta de las tareas. Se tiene que volver a planificar y en algunos casos incluso a volver a analizar y rediseñar la solución.	Baja	Bajo
Análisis y diseño		
Diseño complejo que impida cumplir los hitos marcados.	Baja	Alto
Diseño demasiado sencillo que probablemente nos conduzca a posteriores rediseños y retrasos en las entregas.	Alta	Medio
Demasiados sub casos de uso o abuso de cláusulas de inclusión y requerido entre ellos.	Media	Alto
Intentos de innovar en el aspecto de la arquitectura nos conducen a periodo extra de formación.	Baja	Medio
Prototipo pretencioso	Baja	Medio
Implementación		
Utilización de tecnologías desconocidas o librerías de terceros pueden derivar en más tiempo extra de formación.	Media	Medio

Código fuente enrevesado. Utilización de *antipatrones* y problemas con la posibilidad de reutilización de código.	Alta	Medio
Código fuente de mala calidad, acoplado o sin cohesión. Las modificaciones por cambios de requisitos pueden aumentar los tiempos de forma dramática.	Media	Alto
Testeo		
Dar por supuesto el buen funcionamiento de algún caso de uso.	Baja	Medio
Realización de pruebas condicionadas y no realistas.	Media	Alto
Problemas de recursos de hardware.	Media	Medio
No realizar las pruebas de manera ordenada y por componentes o funcionalidades.	Alta	Bajo
Factores externos y personales		
Catástrofes ambientales como incendios, inundaciones, etc. que produzcan perdida de datos importantes.	Baja	Alto
Enfermedades que impidan continuar con el trabajo.	Baja	Bajo
Falta de experiencia en tecnologías .NET o desarrollo para plataformas móviles	Alta	Bajo

Para poder abordar los diferentes riesgos de una manera ágil y poder anticiparse a ellos en la medida de lo posible se llevarán cabo las siguientes acciones:

- Planificar exactamente lo que se analizará, diseñará y desarrollará y a la inversa, es decir, no desarrollar más de lo que se ha planificado, analizado y diseñado.
- Hacer testeos de código constante y de funcionalidades separadas. No es una buena práctica testear toda la aplicación de una sola vez.
- Desacoplar e implementar código reutilizable en la medida de lo posible.
- Descentralizar el código fuente en un servidor en la nube. De esta manera se podrán evitar accidentes derivados con la perdida de datos e información.
- Contemplar un segundo entorno de desarrollo de reserva en caso de que el primero sufriera algún tipo de accidente.

3. Análisis

3.1. Análisis de requerimientos

3.1.1. Requerimientos funcionales

Primeramente vamos a definir los diferentes requisitos funcionales deseables de la aplicación Snake ILM y se les va a dar una nomenclatura para poder organizarlos de una forma más correcta.

- **REQ1**: Pantalla de inicio con la presentación y las diferentes opciones que tiene el usuario.
    - **REQ1.1**: Nueva partida. Se presentará por pantalla un dialogo para introducir el nombre del jugador junto con una pequeña explicación.
    - **REQ1.2:** Configuración. Se permite al jugador modificar la dificultad del juego Máximas puntuaciones.
    - **REQ1.3**: El jugador puede visualizar los datos guardados de las partidas con el nombre del jugador, número de puntos y tiempo empleado.
    - **REQ1.4**: Acerca de Snake ILM. Pantalla con la información básica sobre el autor del juego.
    - **REQ1.5**: Salir del juego. Se mostrará al usuario un diálogo de confirmación para poder salir del juego.
- **REQ2**: Visualización de la Pantalla de configuración.
    - **REQ2.1**: La configuración de la dificultad del juego se guardará en el sistema de archivos de Windows Phone.
    - **REQ2.2**: A través de un botón se podrán borrar las máximas puntuaciones.
- **REQ3**: Diálogo de introducción de usuario.
    - **REQ3.1**: Se mostrará una caja de texto y un botón para poder introducir el nombre.

  - **REQ3.2:** Se desplegará el teclado de Windows Phone para que el usuario pueda introducir su nombre cuando el usuario toque la caja de texto.
  - **REQ3.3**: El botón verificará que el nombre no está vacío.
- **REQ4**: Visualización de la pantalla de juego.
  - **REQ4.1**: El jugador tocará sobre la pantalla para cambiar la dirección de la serpiente.
  - **REQ4.2**: Visualización por pantalla de los puntos acumulados de la partida actual
  - **REQ4.3**: Visualización por pantalla del tiempo de la partida actual.
  - **REQ4.4**: Opción para pausar la partida.
- **REQ5**: Visualización de la pantalla de pausa.
  - **REQ5.1**: Opción para volver a la pantalla principal.
  - **REQ5.2**: Opción para volver al juego
  - **REQ5.3**: Opción para salir del juego.
- **REQ6**: Visualización de pantalla de final de partida.
  - **REQ6.1**: Opción para volver a la pantalla principal,
  - **REQ6.2**: Opción para empezar nueva partida.
  - **REQ6.3**: Opción para salir del juego.
- **REQ7**: Mecánica del juego. Se implementará toda la mecánica clásica del juego de Snake ILM.
  - **REQ7.1**: La serpiente puede cambiar la dirección hacia arriba o abajo siempre y cuando esté moviéndose lateralmente
  - **REQ7.2**: La serpiente puede cambiar la dirección hacia la izquierda o hacia la derecha siempre y cuando esté moviéndose verticalmente.
  - **REQ7.3**: Cuando la cabeza de la serpiente coincida en el mismo lugar que la comida aumentará el tamaño de esta en una unidad.
  - **REQ7.4**: Cuando la cabeza de la serpiente coincida en el mismo lugar que la comida se aumentará la velocidad del juego.
  - **REQ7.5**: Cada vez que la serpiente coma aparecerá una pieza de comida en algún lugar no ocupado del rectángulo de juego.
  - **REQ7.6**: El juego se acaba cuando la serpiente se topa con el límite del rectángulo de juego.

- **REQ7.7**: El juego acaba cuando la serpiente topa con alguna parte del rectángulo de juego que ya estaba ocupado.
- **REQ7.8**: Cuando acabe una partida, se guardará la puntuación en un fichero en el sistema de archivos de Windows Phone únicamente si es una de las 8 puntuaciones más altas.

3.1.2. Requerimientos no funcionales

Después de definir los requerimientos funcionales deseables de la aplicación, vamos a definir también los requerimientos no funcionales en los siguientes puntos:

- La aplicación entregada al final del trabajo de final de grado ha de ser una versión estable y testeada para cumplir todos los requisitos especificados anteriormente.
- Debe poseer una interfaz intuitiva y fácil de entender.
- Debido a que es una aplicación sencilla, esta de ser ligera y consumir pocos recursos.
- De la misma manera que en el punto anterior, debe ser una aplicación eficiente y rápida. Se deben intentar minimizar al máximo los tiempos de carga y la transición entre pantallas.
- Ahorro de recursos. debe ser una aplicación ligera.
- Tiempos de carga razonables para un juego que exige cierta agilidad a la hora de las transiciones entre pantallas.

3.2. Diseño de casos de uso

3.2.1. Especificación

Para realizar la especificación de todos los casos de uso existentes dentro del sistema del juego, tal y como aparece en el diagrama principal del apartado anterior, vamos a dividir estos casos de uso en diferentes componentes los cuales van a coincidir con las diferentes pantallas a las que tenemos acceso dentro del propio juego.

Pantalla de inicio

La pantalla de inicio nos presenta las diferentes opciones principales y la pantalla principal que veremos al acceder al juego desde el menú de Windows Phone.

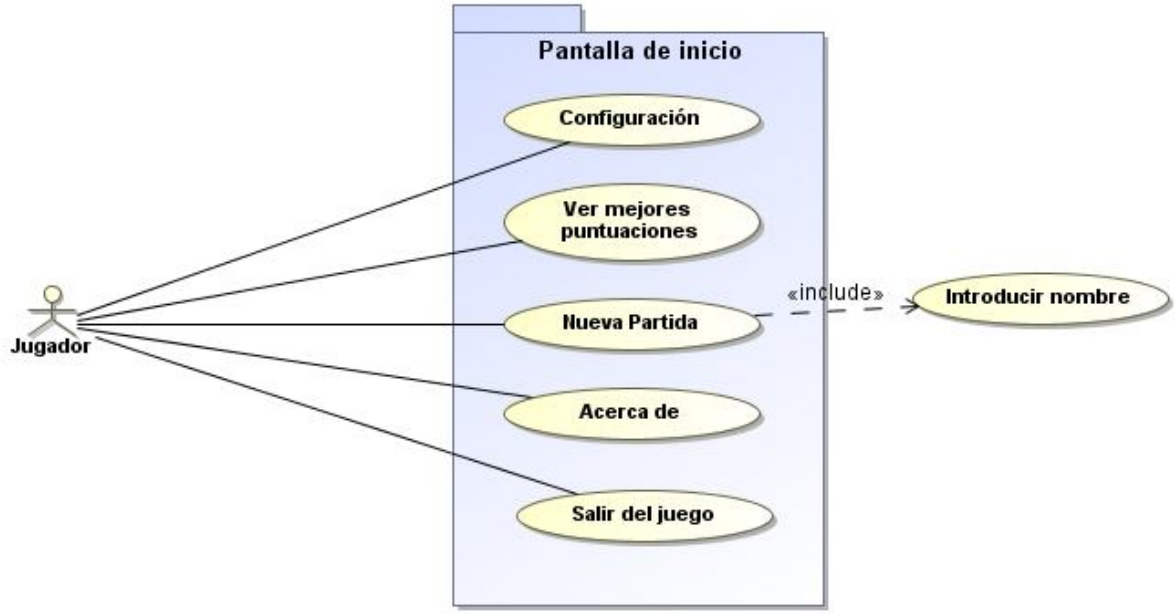


Figura 3

Nombre	**Nueva partida**
Descripción	Opción del menú principal que permite al usuario poder comenzar una partida a Snake ILM
Actores	Jugador
Precondiciones	El sistema verifica el nombre del usuario.
Flujo normal	1. El usuario toca el botón comenzar en el menú principal. 2. El sistema muestra un dialogo que el usuario introduzca su nombre.
Flujo alternativo	
PostCondiciones	Se está mostrando la pantalla de introducir nombre

Nombre	**Introducir nombre**
Descripción	Pantalla para que el jugador pueda introducir su nombre
Actores	Jugador
Precondiciones	El jugador ha tocado la opción de Nueva partida.

Flujo normal	1. El usuario toca la caja de texto para poder introducir su nombre. 2. El sistema le muestra el teclado de Windows Phone. 3. El usuario introduce su nombre. 4. El usuario toca el botón de Comenzar. 5. El sistema verifica el nombre. 6. El sistema muestra la pantalla de juego.
Flujo alternativo	5. El sistema informa al usuario que el nombre es incorrecto o está vacío.
PostCondiciones	Se está mostrando la pantalla de juego

Nombre	**Configuración**
Descripción	Se muestra al usuario una pantalla para modificar la dificultad y borrar las máximas puntuaciones.
Actores	Jugador
Precondiciones	
Flujo normal	1. El usuario toca la opción de configuración en la pantalla principal. 2. El sistema dibuja la pantalla de configuración y muestra un slider para modificar la dificultad y un botón para borrar las máximas puntuaciones.
Flujo alternativo	
PostCondiciones	La pantalla de la configuración del juego se está mostrando.

Nombre	**Ver máximas puntuaciones**
Descripción	El sistema muestra un listado de las máximas puntuaciones conseguidas en el juego.
Actores	Jugador
Precondiciones	
Flujo normal	1. El usuario toca el botón “Ver máximas puntuaciones”. 2. El sistema recupera el fichero guardado en el sistema. 3. El sistema dibuja en pantalla las máximas puntuaciones junto con el tiempo de juego de las partidas y el nombre del usuario.
Flujo alternativo	2. El sistema no encuentra el fichero de máximas puntuaciones. 3. El sistema crea un nuevo fichero. 4. El sistema muestra en pantalla un mensaje de información al jugador de que no existen puntuaciones guardadas.
PostCondiciones	La pantalla de las máximas puntuaciones se está mostrando por pantalla.

Nombre	**Acerca de**
Descripción	Pantalla para visualizar alguna información sobre el creador del

	juego.
Actores	Jugador
Precondiciones	
Flujo normal	1. El usuario toca la opción Acerca de. 2. El sistema dibuja una pantalla con la información general sobre el juego.
Flujo alternativo	
PostCondiciones	La pantalla de acerca de se está mostrando

Nombre	**Salir del juego**
Descripción	Opción para cerrar la aplicación y salir del sistema
Actores	Jugador
Precondiciones	
Flujo normal	1. El usuario toca la opción de Salir del juego. 2. El sistema muestra un diálogo para verificar que el jugador está seguro de salir de la aplicación. 3. El usuario aprieta el botón Start para continuar. 4. El sistema cierra el juego.
Flujo alternativo	3. El usuario aprieta Atrás. 4. El sistema cierra el diálogo y muestra la pantalla de inicio.
PostCondiciones	

Pantalla de configuración

En la pantalla de configuración podremos realizar las opciones que se muestran en el siguiente diagrama de casos de uso.

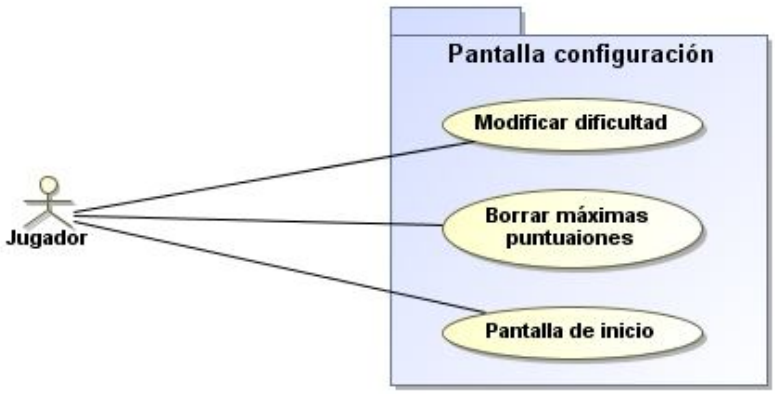


Figura 4

Nombre	**Modificar dificultad**
Descripción	Control tipo slider para modificar la dificultad del juego
Actores	Jugador

Precondiciones	El jugador ha tocado la opción de configuración en la pantalla principal del juego
Flujo normal	1. El usuario el texto del nivel de dificultad mostrado actualmente para modificarlo.
Flujo alternativo	
PostCondiciones	

Nombre	**Borrar máximas puntuaciones**
Descripción	Opción para cerrar la aplicación y salir del sistema
Actores	Jugador
Precondiciones	El jugador ha tocado la opción de configuración en la pantalla principal del juego
Flujo normal	1. El usuario toca el botón de borrar las máximas puntuaciones. 2. El sistema busca el fichero de máximas puntuaciones en el los archivos donde se alojan las máximas puntuaciones de la aplicación dentro del sistema de Windows Phone. 3. El sistema borra el fichero. 4. El sistema muestra al usuario un mensaje de que el borrado del archivo de las máximas puntuaciones.
Flujo alternativo	1. El sistema no encuentra el fichero de máximas puntuaciones. 2. El sistema devuelve un mensaje al usuario de que no existe el fichero de máximas puntuaciones.
PostCondiciones	

Nombre	**Pantalla de inicio**
Descripción	Opción para volver a la pantalla de inicio del juego
Actores	Jugador
Precondiciones	El jugador ha tocado la opción de configuración en la pantalla principal del juego
Flujo normal	1. El usuario toca la opción para volver a la pantalla principal de la aplicación. 2. El sistema guarda el nivel de dificultad en el archivo de configuración de la aplicación.
Flujo alternativo	
PostCondiciones	La pantalla de inicio se está mostrando

Pantalla de juego

La pantalla de juego nos presenta la posibilidad de modificar la dirección de la serpiente y también la de pausar la partida. Las posibles opciones se muestran en el siguiente diagrama de casos de uso.

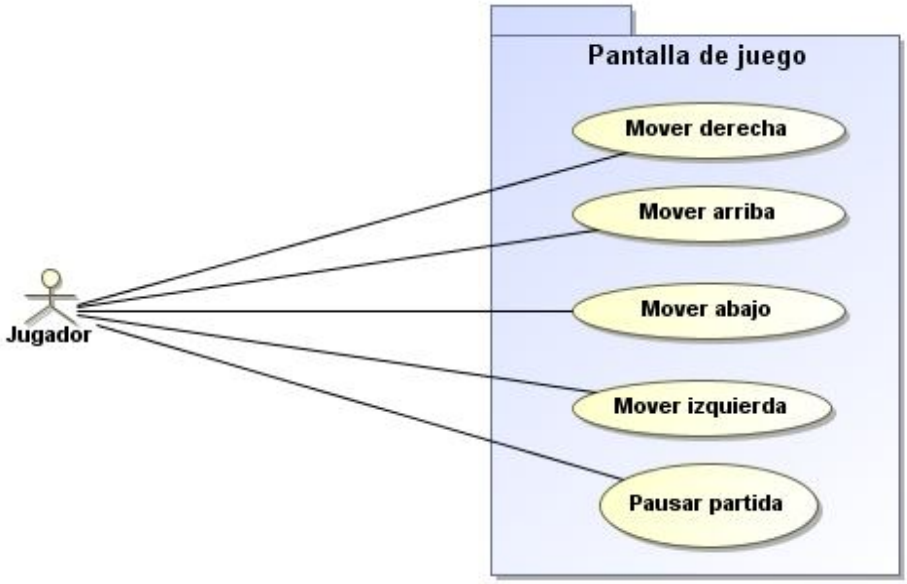


Figura 5

Nombre	**Pausar partida**
Descripción	Cambia la dirección de la serpiente hacia la derecha
Actores	Jugador
Precondiciones	El jugador ha creado una nueva partida
Flujo normal	1. El jugador toca el botón de pausa de la pantalla. 2. El sistema guarda el estado de la partida actual. 3. El sistema muestra un diálogo con las diferentes opciones que puede utilizar el usuario.
Flujo alternativo	
PostCondiciones	La pantalla de pausa se está mostrando.

Nombre	**Mover derecha**
Descripción	Cambia la dirección de la serpiente hacia la derecha
Actores	Jugador
Precondiciones	El jugador ha creado una nueva partida. La serpiente se está moviendo verticalmente (arriba o abajo)
Flujo normal	1. El jugador aprieta a la derecha de la serpiente. 2. El sistema gira 90 grados la cabeza de la serpiente y cambia su dirección hacia la derecha.
Flujo alternativo	
PostCondiciones	El movimiento de la serpiente es ahora hacia la derecha.

Nombre	**Mover izquierda**
Descripción	Cambia la dirección de la serpiente hacia la izquierda

Actores	Jugador
Precondiciones	El jugador ha creado una nueva partida. La serpiente se está moviendo verticalmente (arriba o abajo)
Flujo normal	1. El jugador aprieta a la izquierda de la serpiente. 2. El sistema gira 90 grados la cabeza de la serpiente y cambia su dirección hacia la izquierda.
Flujo alternativo	
PostCondiciones	El movimiento de la serpiente es ahora hacia la izquierda.

Nombre	**Mover arriba**
Descripción	Cambia la dirección de la serpiente hacia arriba
Actores	Jugador
Precondiciones	El jugador ha creado una nueva partida. La serpiente se está moviendo horizontalmente (izquierda o derecha)
Flujo normal	1. El jugador aprieta el terreno de juego por la parte superior de la serpiente. 2. El sistema gira 90 grados la cabeza de la serpiente y cambia su dirección hacia arriba.
Flujo alternativo	
PostCondiciones	El movimiento de la serpiente es ahora hacia arriba.

Nombre	**Mover abajo**
Descripción	Cambia la dirección de la serpiente hacia abajo
Actores	Jugador
Precondiciones	El jugador ha creado una nueva partida. La serpiente se está moviendo horizontalmente (izquierda o derecha)
Flujo normal	1. El jugador aprieta el terreno de juego por la parte inferior de la serpiente. 2. El sistema gira 90 grados la cabeza de la serpiente y cambia su dirección hacia abajo.
Flujo alternativo	
PostCondiciones	El movimiento de la serpiente es ahora hacia abajo.

Pantalla de pausa

Esta pantalla nos permite ejecutar una serie de acciones que ya habíamos definido en casos de uso anteriores, en este caso el diagrama de casos de uso es el siguiente:

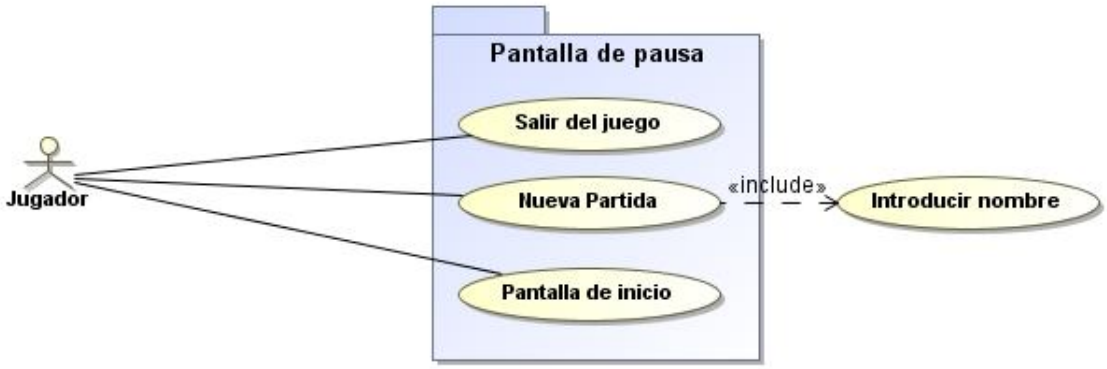

Pantalla de pausa
Salir del juego
Nueva Partida
Pantalla de inicio
«include»
Introducir nombre
Jugador

Figura 6

4. Diseño

4.1. Modelo de objetos

En el apartado de modelo de datos vamos a presentar las diferentes clases de las que se va a componer la aplicación mediante un diagrama de clases y un apartado posterior para explicar cada una de estas entidades y la función que va a tener dentro del juego.

4.1.1. Diagramas de clases

Para definir el modelo de datos y clases que se va a seguir en este proyecto, se van a presentar dos diagramas de clases diferentes y un diagrama de componentes principal que nos dará una visión más general de todo el código fuente del proyecto. Uno de ellos centrado en toda la lógica de transiciones entre pantallas, cambios de estado, etc. y el otro dedicado en exclusiva a la lógica propia del juego.

4.1.1.1. Controlador de estados

Antes de entrar en la propia mecánica del juego y todo lo que conlleva, vamos a necesitar un gestor de estados que pueda gestionar la transición de vistas y dibujar por pantalla los componentes correspondientes según el estado en el que nos encontremos. Este gestor o controlador de estados actuará a modo de capa de objetos que estará situada entre la capa de la lógica de nuestro juego y el framework de XNA. Se considera la opción de abstraer esta parte del juego (gestión de los estados/pantallas) de la parte más funcional para facilitar el trabajo y las posteriores modificaciones que pudiesen surgir a modo de correcciones o evolutivos.

Para llevar a cabo esto, se va a partir de una base propuesta por Microsoft llamada *Game State Management*, la cual se ha de desarrollar para conseguir el resultado deseado.

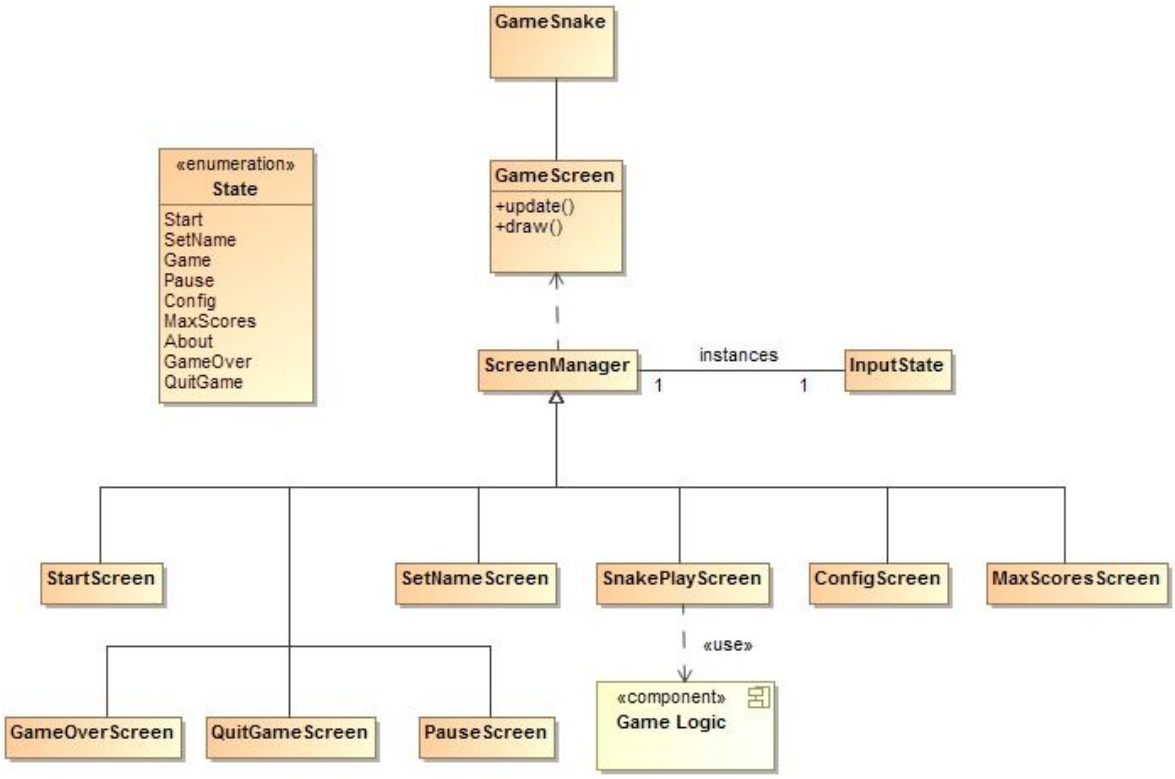


Figura 7

A continuación se realizará una descripción de las diferentes clases que aparecen en el diagrama anterior:

Game

Es la clase más importante de la aplicación. Esta clase hereda de la clase *Game* de las librerías de XNA e implementará los métodos de *Game* necesarios para construir todo el ciclo de vida de la aplicación.

En esta clase se guardarán variables globales tales como la ruta donde se encuentran guardados los ficheros de configuración y máximas puntuaciones, nombre del jugador que está jugando actualmente, etc.

InputState

Como a través de la base propuesta por Microsoft tendremos las diferentes clases por pantalla gestionadas desde un mismo lugar, la clase *InputState* nos

va abstraer de toda la lógica referente a la entrada de datos ya sea desde el teclado o desde las capacidades táctiles de la pantalla.

State

Clase de tipo *Enum* que nos especificará todos los posibles estados en los que puede estar el juego. Definimos los estados como las posibles pantallas en las que puede estar el jugador, es decir, estado de Inicio, estado de Pausa, estado de Configuración, etc.

A partir de esta clase de tipo enumeración, podremos controlar con el manager de estados la pantalla que está activa en ese momento y que contenido es el que tenemos que mostrar por pantalla.

GameScreen y ScreenManager

La clase *GameScreen* será una clase abstracta y *ScreenManager* implementará métodos de la primera. La utilizaremos para presentar y actualizar pantallas aparte de la posibilidad de poder determinar cuál es la pantalla activa.

Cada una de los estados definidos con la clase *State* tendrá una clase asociada dentro del programa. Cada una de estas vistas heredará directamente de la clase base *GameScreen* excepto las que están basadas en diálogos como *PauseScreen, GameOverScreen* y *QuitGameScreen* las cuales lo harán de la clase DialogScreen y esta a su vez de la ya nombrada *GameScreen.* Se dotará a estas vistas de la funcionalidad y/o componentes gráficos correspondiente. A continuación se define una pequeña descripción de cada una de estas vistas:

- **SnakePlayScreen:** Pantalla principal de juego. En el apartado de Mecánica del juego se explicará más detalladamente.
- **ConfigScreen:** Las diferentes funciones de esta vista estarán relacionadas con el tema de carga y borrado de máximas puntuaciones y modificación de dificultad. Entre sus métodos tenemos los siguientes:
    - *saveSettings*: Guarda la configuración dentro de un fichero en el sistema de archivos de Windows Phone.

  - *getHancicapLevel*: carga en el sistema el nivel de dificultad escogido dentro de la pantalla de configuración y guardado dentro del sistema de arhivos de Windows Phone.
  - *deleteScores***:** elimina el archivo de máximas puntuaciones.
- **AboutScreen:** Únicamente nos mostrará información sobre el juego y otras informaciones adicionales.
- **StartScreen:** Será la pantalla principal del juego y se visualizará un menú con las diferentes opciones que se pueden llevar a cabo.
- **SetNameScreen:** Aquí podremos introducir el nombre que posteriormente utilizaremos a la hora de guardar las puntuaciones cuando se finalice una partida. Tendrá los métodos siguientes.
  - *checkName*: Comprueba la validez del texto introducido como nombre de usuario.
  - *setName*: Guarda el nombre del usuario de manera global en la clase Game.
- **MaxScoresScreen:** Vista de máximas puntuaciones que recuperará el archivo guardado en el sistema y mostrará por pantalla las 5 máximas puntuaciones guardadas.
  - *getFileScores***:** recupera el fichero que guarda las máximas puntuaciones y las carga en una lista de datos en memoria.
- **DialogScreen:** Pantalla de dialogo base que cambiará la opacidad de la pantalla y mostrará un diálogo.
- **GameOverScreen:** Diálogo que nos mostrará información sobre la partida y un menú con diferentes acciones. Es muy similar a la vista de Pausa. Extenderá a la clase *DialogScreen*.
- **QuitGameScreen:** Diálogo de confirmación para que el usuario certifique que realmente quiere salir de la aplicación. Extenderá a la clase *DialogScreen*.
- **PauseScreen:** La pantalla de pausa nos mostrará diversas opciones y parará el progreso actual en el juego. Extenderá a la clase *DialogScreen*.

4.1.1.2. Lógica del juego

Este segundo diagrama de clases se centra en la propia mecánica del juego y que clases se van a definir para administrar las puntuaciones de juego, actualización de los posicionamientos en pantalla, configuración de máximas puntuaciones, etc. En definitiva, todo lo referente al propio proceso, almacenamiento de datos y configuraciones del juego Snake ILM.

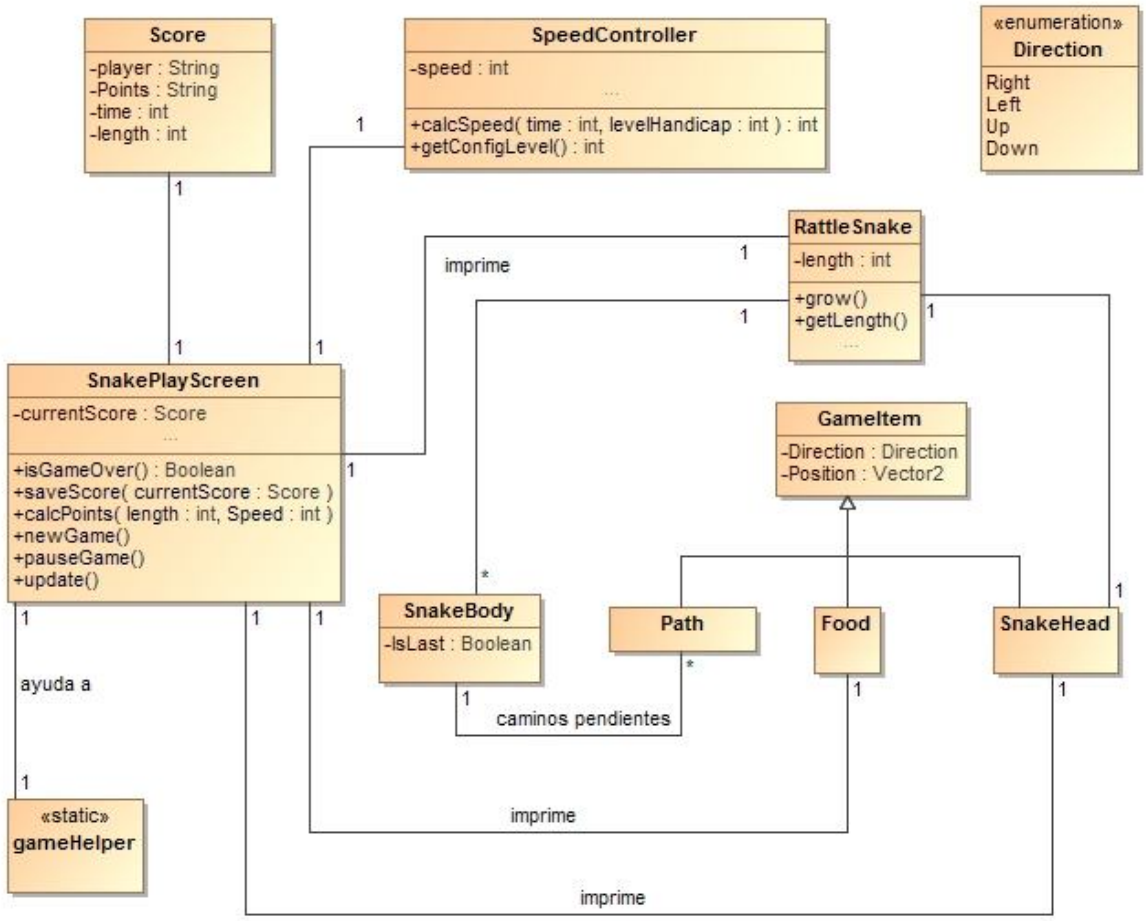


Figura 8

4.1.2. Especificación de entidades

En este apartado explicaremos que son y para que se van a utilizar las diferentes entidades definidas en el diagrama de clases del proyecto. Para cada una de estas entidades, se especificará el uso de los métodos que contiene.

SpeedController

Esta clase almacenará la velocidad actual del juego y la dirección en la que se mueve la serpiente. Aparte de esto, también podrá calcular la velocidad de este a partir de la configuración de dificultad escogida.

- ***calcSpeed***: calcula la velocidad de movimiento de la serpiente a partir del tiempo actual de juego y el hándicap de dificultad especificado en la configuración del juego.

Texture2D

Clase base de la que partirán cada uno de los elementos que se muestren por pantalla. Esta clase pertenece a las librerías del *framework* de XNA. En este caso particular será del tipo Texture2D.

RattleSnake

Esta clase representará el conjunto de elementos que contiene la serpiente, sin contar la cabeza de esta. Según la serpiente vaya recogiendo comida, se irán añadiendo a este listado. La clase RattleSnake tendrá algunos métodos interesantes de comentar:

- ***grow***: llamado desde la función update de la misma clase y añadirá un nuevo ítem de tipo SnakeBody al listado.
- ***getlength***: método encargado de recuperar el tamaño actual de la serpiente.

Food

Elemento del juego que representará los elementos que debe alcanzar la serpiente para poder crecer. Será una instancia de la clase *Texture2D* y tendrá una representación por pantalla exclusiva.

SnakeBody

Clase que referenciará a cada parte del cuerpo de la serpiente la cual va a almacenar las rutas pendientes que le faltan por recorrer a dicha parte, y también la dirección y posición que tiene actualmente.

SnakeHead

Será una instancia de *Texture2D* y representará por pantalla a la cabeza de la serpiente, existente desde el principio y hasta el final del juego. La instancia se creará dentro del elemento RattleSnake.

Position

Las librerías de XNA ya ofrecen un elemento de este tipo, pero para una mayor comprensión del esquema de datos del juego, se especifica como una clase separada dentro del diagrama de clases. Entrando en detalle, esta clase nos especifica un punto en pantalla definido por dos coordenadas.

SnakePlayScreen

Al igual que las demás vistas del juego, heredará de la clase base llamada *GameScreen*. Esta clase es la que manejará gran parte de la lógica del juego y el ciclo de vida de la aplicación mediante funciones como *Draw* o *Update* que ya se explicarán más en detalle en el apartado de Implementación. Las funciones que se van a implementar en esta clase serán:

- ***isGameOver***: a cada ciclo de actualización se comprobará que la serpiente no haya tocado ninguno de los bordes o no se haya tocado a sí misma.
- ***update***: esta función actualiza el reloj de la aplicación y actualiza las diferentes posiciones de los elementos en pantalla.
- ***isEating***: será llamada por la función update de la clase y confirmará si la serpiente ha coincidido en la misma posición que la comida. En caso afirmativo actualizará aleatoriamente la posición del elemento *Food* y también a la función de RattleSnake llamada *grow*.
- ***MoveBody***: Se encargará de mover cada una de las partes del cuerpo de la serpiente en la dirección correspondiente según lo marcado por las diferentes rutas a tomar definidas anteriormente por las direcciones tomadas por la cabeza de la serpiente.
- ***saveScore***: a través de esta función y una vez se ha acabado la partida llamará a la clase de configuración y le enviará los datos.
- ***calcPoints***: calcula los puntos a través del tamaño de la serpiente y el tiempo transcurrido de juego.

- ***newGame***: limpia las diferentes instancias, actualiza el estado, borra puntuaciones, etc. y demás operaciones necesarias para generar un nuevo juego desde 0.
- ***pauseGame***: cambia la pantalla que se muestra en el juego y cambia el estado del mismo al estado de Pausa correspondiente.
- ***saveMax:*** guarda en el archivo correspondiente las máximas puntuaciones del listado de puntuaciones.

Todas las clases que sean una herencia de la clase Game de XNA tendrán ciertas funciones de base que se implementarán para poder cumplir el ciclo de vida de los juegos desarrollados. Estas funciones son *LoadContent, Draw, Update, Initialize* y *UnloadContent.* En el apartado de Implementación se dará una descripción más a fondo de para que se utilizan estas funciones.

Direction

Clase de tipo enumeración que nos especifica las posibles direcciones en las que la serpiente puede moverse.

Route

Esta clase almacenará las diferentes rutas tomadas por la cabeza de la serpiente y las cuales deberán seguir las demás partes del cuerpo. La ruta será simplemente la dirección y la posición donde la cabeza cambió de sentido.

Score

La clase Score será la clase encargada de guardar las diferentes puntuaciones que tiene una partida, es decir, se guardará tanto el tiempo de partida, puntos, tamaño de la serpiente y el nombre del jugador.

4.2. Arquitectura

4.2.1. Arquitectura XNA

Para poder explicar la arquitectura empleada en el juego del trabajo de final de grado que se está llevando a cabo, se necesita primero explicar la arquitectura propia del framework de XNA de Microsoft.

En primer lugar, expondremos un gráfico y en segundo lugar explicaremos de manera general cada una de estas capas y lo que proveen al programador.

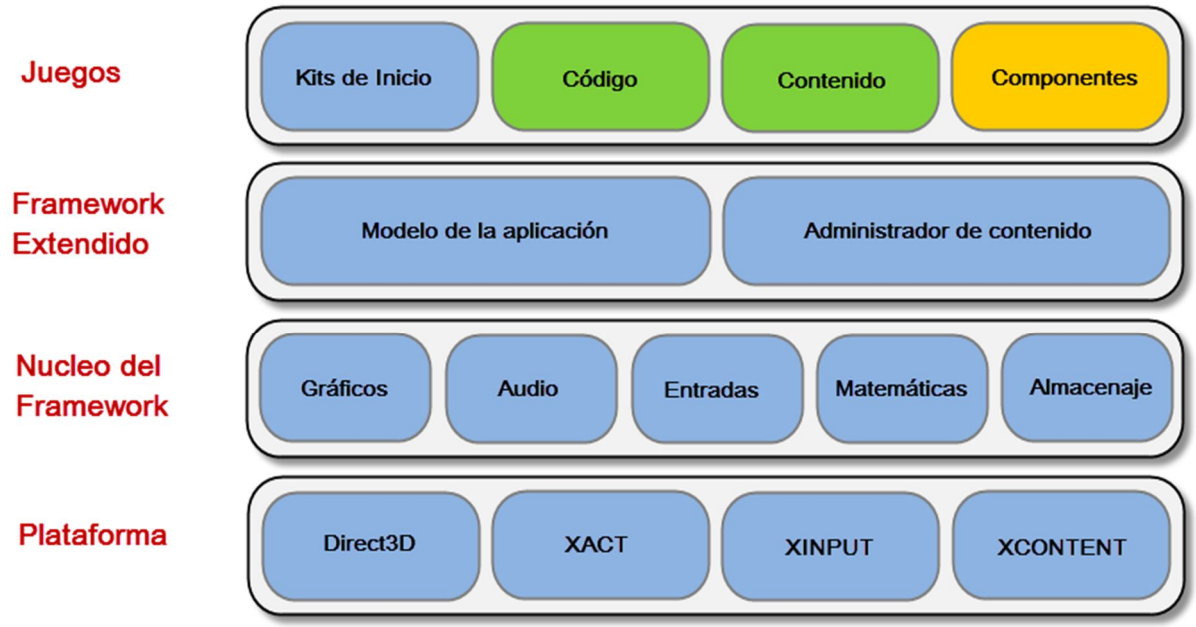


Figura 9

En el gráfico podemos distinguir que la arquitectura de XNA se compone de 4 capas diferentes muy bien diferenciadas. Empezando desde la capa más baja de ellas:

Plataforma

Se compone de las API nativas y administradas de bajo nivel del framework XNA. Algunas de las API incluidas en esta capa son Direct3D 9 XACT, XInput y XContent.

Núcleo

Esta capa proporciona la funcionalidad básica de XNA, en concreto, nos ofrece funcionalidad de audio, gráficos, entrada de datos, operaciones matemáticas y también almacenamiento. Se entrará más en detalle en el apartado de Implementación.

Extensiones del framework

Como su nombre indica, extiende las funcionalidades provistas por el núcleo. Concretamente, estas extensiones pretenden ayudar al desarrollador y serle más fácil la tarea de crear un juego a través de un par de componentes como el Content Pipeline o el modelo de Aplicación el cual nos provee de algunas funciones que controlan aspectos como la actualización del reloj del sistema o la carga de contenido. Como ya se ha comentado anteriormente se entrará más en detalle en el punto de Implementación dentro de la memoria.

Juegos

La capa más alta del framework, en la cual se introducirá el código fuente, componentes de terceros, diferentes recursos en forma de audios, videos, imágenes, etc. y también los kits de inicio.

4.2.2. Arquitectura Snake en XNA

Partiendo del gráfico del apartado anterior, procederemos a explicar dónde va a encajar el desarrollo de la aplicación Snake ILM dentro de este framework. No es muy difícil intuir que nuestra aplicación encajará dentro de los dos recuadros de color verde de la figura del apartado 4.2.1. En este caso, todo lo referente al código fuente y los diferentes recursos que introduzcamos ya sean iconos de inicio de la aplicación o también recursos como imágenes de fondo estarán localizados dentro de la parte de contenido.

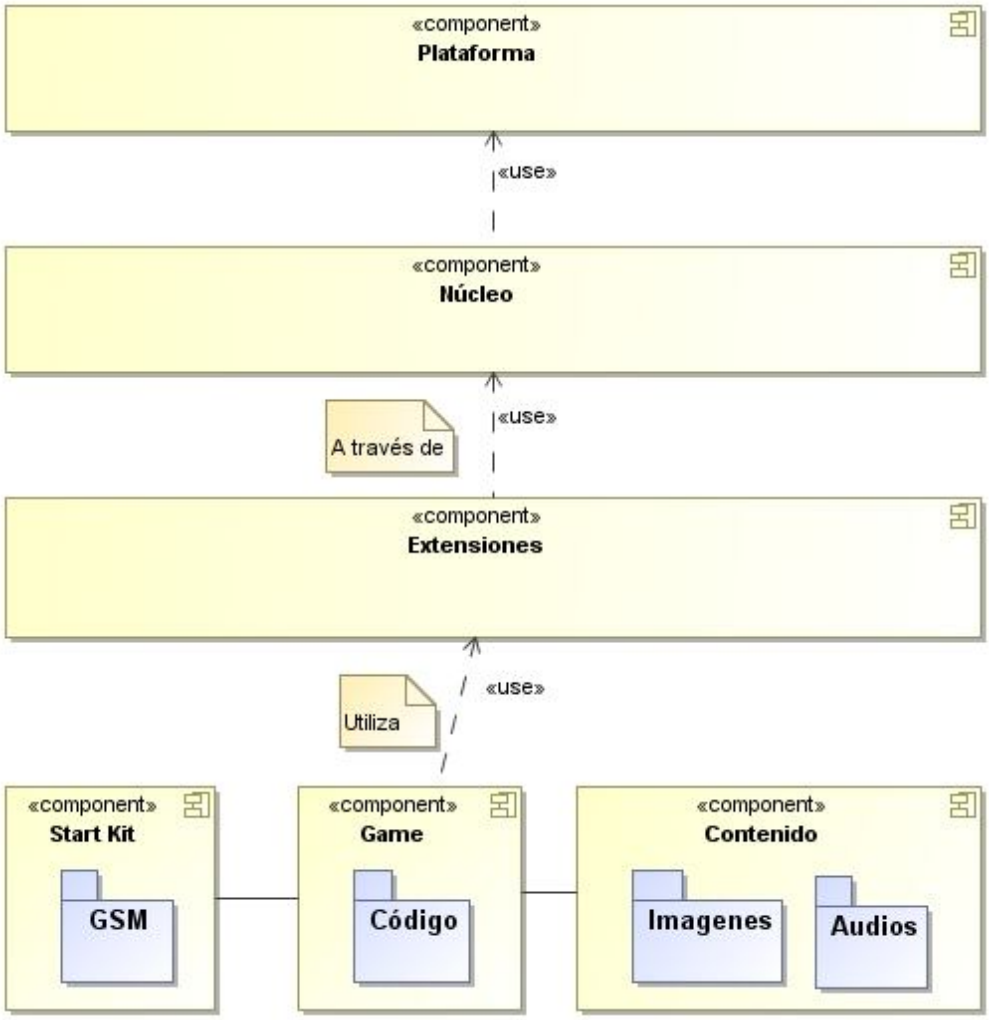


Figura 10

Entrando más en detalle, explicaremos en cada una de las capas relacionándolas con la aplicación.

Plataforma

En el caso de Snake ILM, nos vamos a basar en una plataforma en 2 dimensiones. Por defecto, utilizaremos todas las capacidades y funcionalidades que el motor gráfico de XNA nos va a proveer.

Núcleo

De la capa del núcleo heredaremos las clases necesarias para poder implementar por ejemplo el cuerpo y cabeza de la serpiente, ya que

utilizaremos la clase Círculo para dibujarlos, al igual que el rectángulo para dibujar la zona de juego. A parte de la parte gráfica, también utilizaremos las clases referentes a la entrada de datos por pantalla en la introducción del nombre o la entrada táctil de las pantallas de Windows Phone. La parte matemática será utilizada para calcular los diferentes cambios de posición que ocurrirán dentro del juego.

En definitiva, para desarrollar nuestra aplicación nos ayudaremos de todo el framework que nos provee XNA para desarrollar toda la funcionalidad básica de Snake ILM ya sea la parte gráfica, las transiciones de pantalla, entradas de datos, cálculos de posicionamiento dentro de pantalla o incluso audio.

Extensiones del framework

Con esta capa y sus componentes podremos acceder más fácilmente a crear todo lo referente al juego de manera más rápida.

Juegos

En esta parte es donde entrará en juego todo el código que desarrollemos para implementar la funcionalidad y mecánica del propio juego. A parte de esto, todos los recursos de imágenes, audios, o cualquier cosa que nosotros vayamos a añadir serán considerados parte del contenido de nuestro juego. Aparte de lo comentado anteriormente, se utilizará un kit de inicio llamado *Game State Management* (GSM) que a su vez servirá de base para realizar toda la administración de vistas y de cambios de pantalla de la aplicación.

En esta capa es principalmente donde introduciremos toda nuestra funcionalidad y todos los componentes.

4.3. Interfaz gráfica

Definidos los diferentes casos de uso que se implementarán durante el desarrollo y el diseño tanto del modelo de datos como de la arquitectura de la aplicación en este apartado se especifica la interfaz gráfica de cada una de las pantallas de las que está

compuesto el juego y un gráfico donde se pueden contemplar las diferentes transiciones entre pantallas a modo de diagrama de estados.

4.3.1. Transición de vistas

En la siguiente figura podemos ver un diagrama de estados para tener una visión más general de cómo estarán organizadas las diferentes pantallas del juego y cuáles son las transiciones entre ellas.

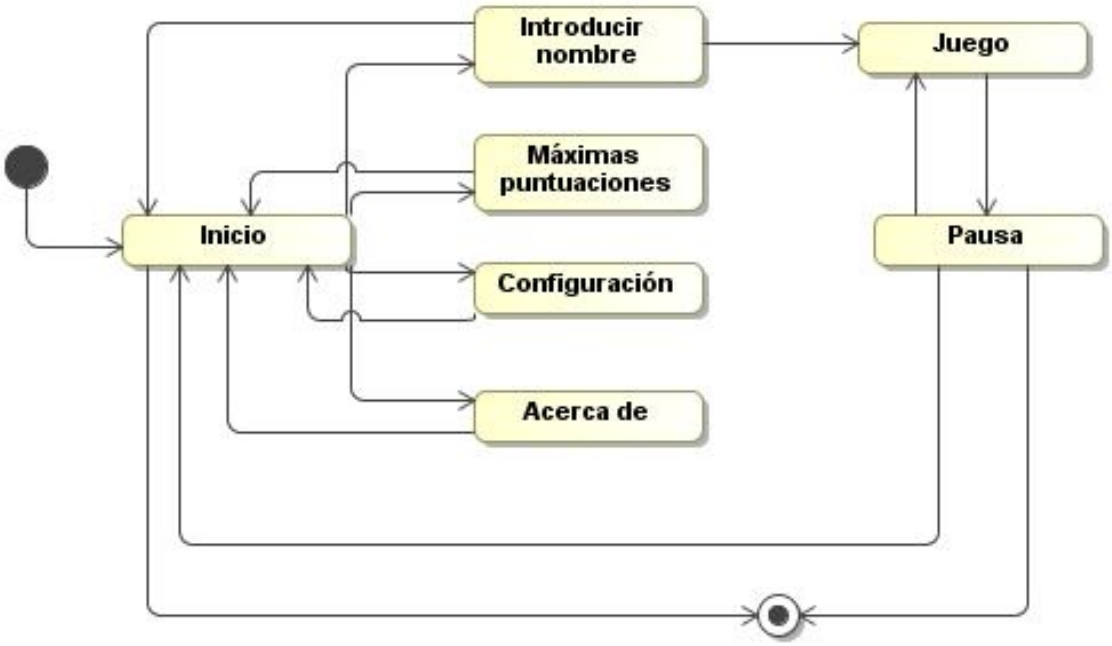


Figura 11

4.3.2. Vista principal

Esta pantalla es la primera que se muestra al usuario una vez entra en la aplicación y presenta las diferentes acciones básicas que se pueden realizar en el juego tales como la creación de una nueva partida, la configuración del juego, visualización de máximas puntuaciones, una pantalla de información de la aplicación y también un enlace para salir del juego.

A parte de estas acciones también se visualiza el título de la aplicación y una imagen a modo de logotipo.

Figura 12

4.3.3. Vista Configuración

Como su nombre indica, se proveerán las diferentes acciones respecto a la configuración del juego, siendo en esta primera versión la modificación de la dificultad y el borrado de las máximas puntuaciones. Existirán 5 niveles de dificultad diferentes (Muy fácil, Fácil, Normal, Difícil y Muy difícil) y se modificará tocando encima del texto que especifica la dificultad. La acción de borrar las puntuaciones también será ejecutada cuando se toque encima del texto correspondiente.

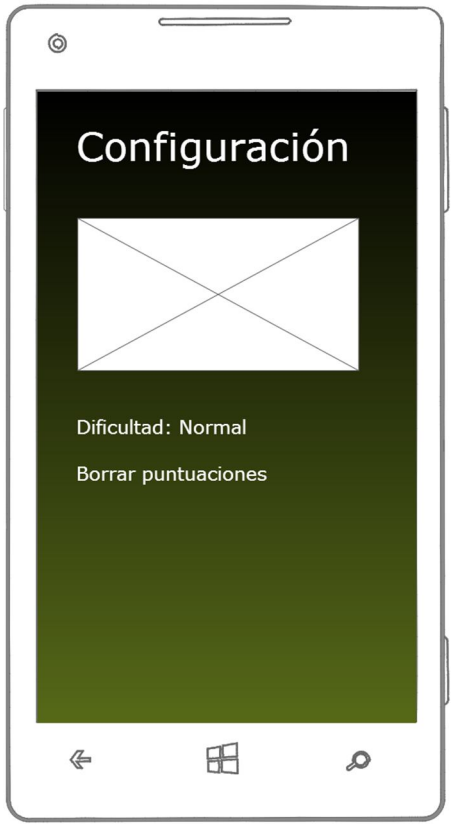


Figura 13

4.3.4. Vista Máximas puntuaciones

La pantalla de máximas puntuaciones muestra al usuario las 5 máximas puntuaciones guardadas en el archivo de disco correspondiente. Los datos que se mostrarán serán el nombre del jugador que consiguió la puntuación, el tamaño que corresponde al número de ítems o comida conseguidos por la serpiente y la puntuación conseguida que es un valor calculado a través del tiempo de partida y los ítems conseguidos.

En el caso de que no existan máximas puntuaciones se informará al usuario que no existen puntuaciones guardadas en el juego. Por último, en la parte inferior derecha se mostrará un enlace para volver a la pantalla de inicio.

Figura 114

4.3.5. Vista Acerca de

La pantalla Acerca de simplemente mostrará información relevante sobre:

- Nombre y versión de la aplicación.
- El desarrollador del juego como el email, empresa, etc.
- Licencia con la que se distribuye la aplicación.
- Información adicional.

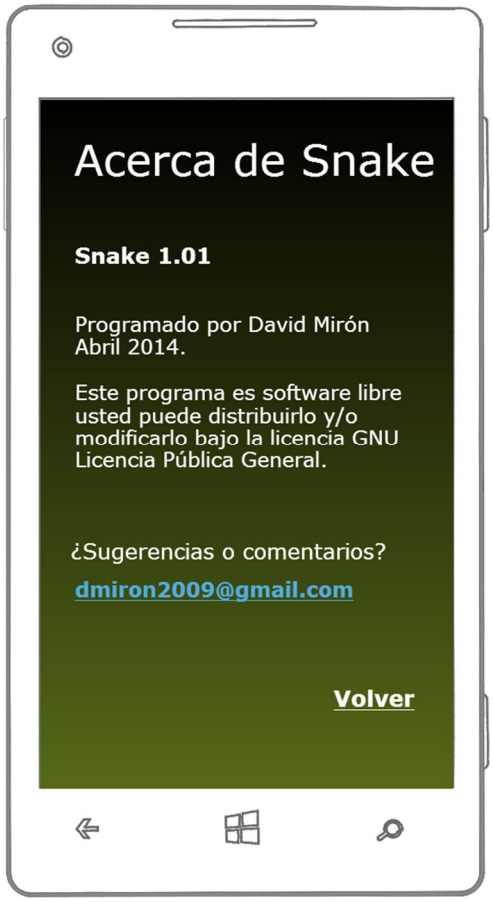


Figura 115

4.3.6. Vista Introducir nombre

Esta es la pantalla que ofrece al jugador introducir su nombre después de tocar el botón Nueva partida. En esta pantalla, aparte de poder introducir el nombre se ofrece una información adicional al jugador a modo de tutorial y recordatorio de como jugar al juego de Snake ILM. El botón Comenzar realiza una verificación del nombre introducido por el usuario y da paso a la pantalla de juego de la aplicación. Esta verificación únicamente comprobará si el usuario ha introducido caracteres extraños y si esta caja de texto tiene algún contenido dentro.

Figura 16

4.3.7. Vista de Juego

La pantalla de juego mostrará dos zonas básicamente, la parte superior donde se alojan las diferentes puntuaciones y el resto de la pantalla donde se aloja la zona de juego Snake ILM.

La parte superior muestra las puntuaciones en tiempo real respecto al tamaño de la serpiente (número de ítems comidos por la serpiente), tiempo total transcurrido desde el inicio de la partida, y el botón para pausar la partida y entrar en el menú de pausa.

La zona de juego como podemos ver tiene un estilo minimalista sin demasiados alardes gráficos mostrando una serpiente con una cabeza fácilmente identificable y las diferentes partes del cuerpo. El corazón que aparece en la pantalla muestra el ítem que debe alcanzar la serpiente para ir creciendo de tamaño.

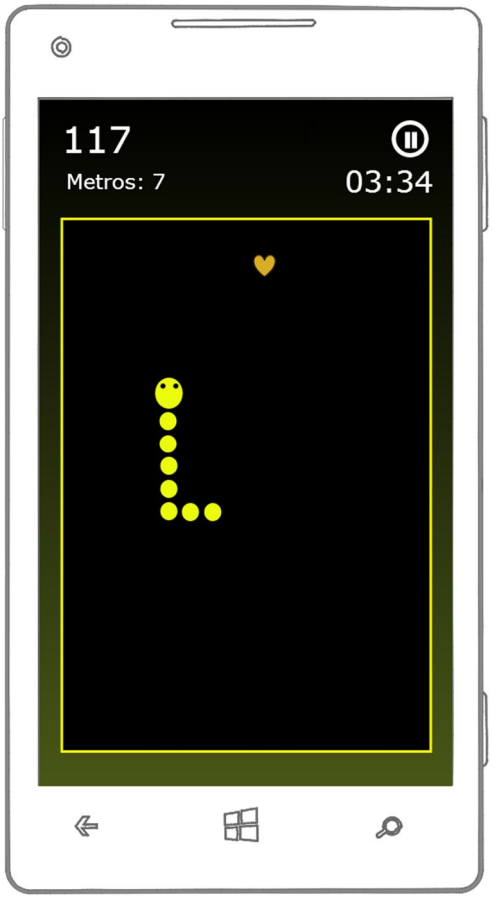


Figura 17

4.3.8. Vista de Pausa

A esta vista podremos acceder a través del botón superior derecho con el icono de pausar de la vista de juego. Esta vista se nos mostrará como un diálogo con varias opciones que pueden ser elegidas. Estas opciones son Volver al juego, volver al menú o salir del juego.

Figura 18

4.3.9. Vista de *Game Over*

La pantalla de *Game Over* no es una pantalla a la que pueda acceder un usuario directamente, es decir, no forma parte de las acciones o de los casos de uso del propio usuario y tampoco se incluye como enlace en ningún sitio dentro del propio prototipo. Independientemente de lo anterior, se presenta la pantalla de *Game Over* para tener una imagen global del juego.

Figura 19

4.3.10. Mensaje para Salir del juego

En este caso, más que una pantalla en sí, se tratará de un diálogo que aparecerá por pantalla pidiéndole al jugador una confirmación para salir del juego. Este diálogo aparecerá pulsando las opciones Salir del juego que aparecen tanto en la pantalla principal como en el menú de pausa del juego. Para confirmar o declinar la salida del juego, se debe tocar la superficie del recuadro azul para seguir con la acción o fuera de esta para cancelar.

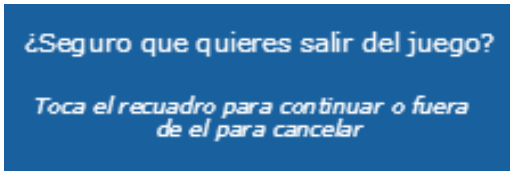


Figura 20

5. Prototipo

5.1. NinjaMock

Para implementar y diseñar el prototipo se ha utilizado la herramienta Web *NinjaMock* totalmente gratuita para uso no comercial y con algunas características muy interesantes que la convierten en la escogida frente a otras opciones para el desarrollo del diseño dela interfaz gráfica y un prototipo totalmente funcional. Entre otras características *NinjaMock*:

- Es gratuito para uso no comercial, permite guardar hasta un total de 3 prototipos en su cuenta gratuita.
- Es colaborativo. Se pueden crear prototipos y posteriormente crear enlaces públicos para que cualquier usuario sin cuenta en la página pueda visualizarlos y también dejar sus comentarios los cuales llegaran al creador del prototipo en forma de email.
- Tiene plantillas prediseñadas como las diferentes interfaces que imitan dispositivos móviles o los propios controles como los cuadros de textos, botones, etc.
- Es tremendamente fácil de utilizar e intuitivo. No necesita demasiados conocimientos previos sobre diseño web y la simpleza de su diseñador gráfico es de gran ayuda.

La URL del proyecto Snake ILM de NinjaMock para acceso público es la siguiente:

http://ninjamock.com/s/cwdhmq

5.2. Interfaz de NinjaMock y prototipo

En la imagen inferior podemos ver la interfaz de NinjaMock y su modo de presentar el prototipo.

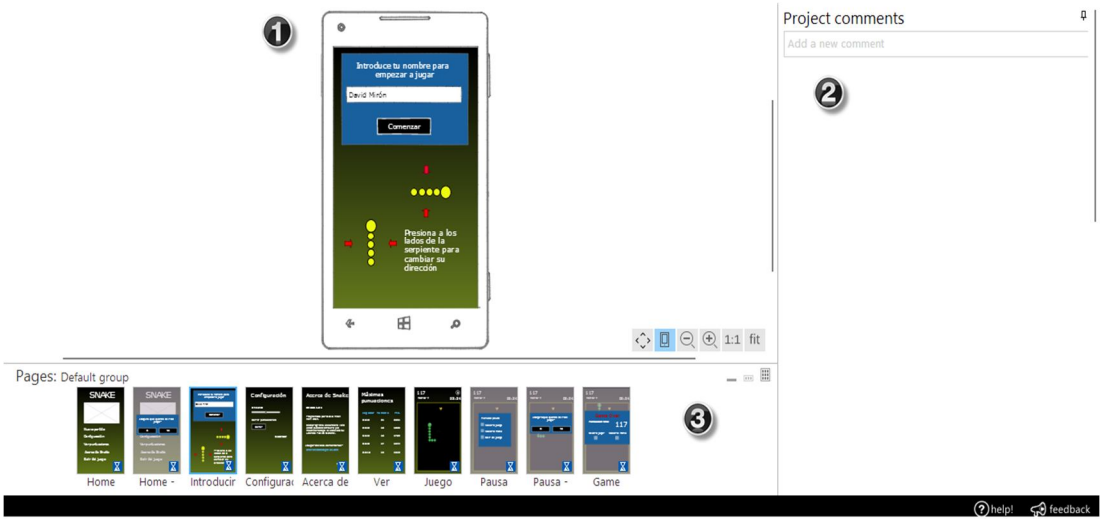


Figura 21

Dentro del navegador podemos diferenciar tres áreas:

1. El área marcada con el número nos sirve como visualizador de la pantalla actual, donde podemos pulsar las diferentes y utilizar las diferentes acciones que nos cambiarán de pantalla.
2. El área 2 es muy interesante en lo que se refiere a las características de colaboración de la herramienta. La caja de comentarios nos ofrece la posibilidad de insertar comentarios a cada una de las diferentes pantallas y que estos comentarios lleguen como emails directamente al buzón de correo del creador o desarrollador del prototipo.
3. Esta última área nos ofrece una visión general de todas las pantallas de la aplicación y la posibilidad de cambiar de pantalla según queramos sin tener que ir pulsando los enlaces del propio prototipo.

Para probar las funcionalidades y validación de interfaz gráfica del prototipo debe hacerse desde Internet Explorer en su versión 8 como mínimo.

5. Implementación

5.1. Aplicación

5.1.1. Ciclo de vida XNA

En este apartado se explica de manera general el ciclo de vida de una aplicación XNA desde que se inicializa hasta que se cierra y se libera.

Básicamente, una aplicación desarrollada en XNA consta de cuatro partes principales muy bien diferenciados:

- **Inicialización y carga de datos**. Como su nombre indica, aquí se cargan los contenidos del juego, es decir, las imágenes, fuentes, sonidos, etc. También se ejecuta una primera función que permite inicializar variables antes de la carga de contenidos.
- **Bucle del juego**. Se ejecutan de manera reiterativa en forma de bucle y es donde reside toda la lógica del juego. Se utiliza para actualizar y realizar cálculos de la lógica del juego con la función *Update* y se dibuja en pantalla a través de la función *Draw*.
- **Descarga de contenidos y cierre.** Guarda el estado, descarga los contenidos cargados en la función Load, etc.
- **Entrada de datos.** XNA permite capturar las entradas (arrastre, toque de pantalla, etc.) de la pantalla táctil en caso de tratarse de Windows Phone.

Estas cuatro partes o fases del ciclo de vida de las aplicaciones XNA se encuentran dentro de la clase principal de tipo Game.

En la siguiente figura se puede ver un esquema del ciclo de vida que siguen las aplicaciones XNA.

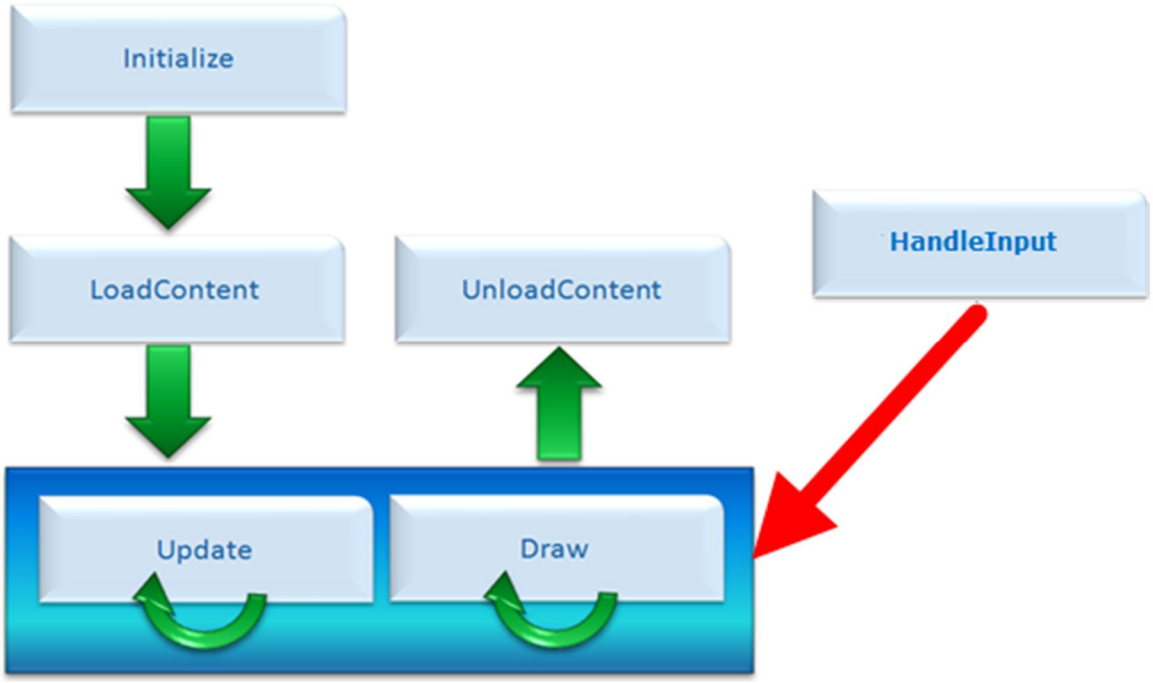


Figura 22

5.1.2. Funciones principales

Para un entendimiento un poco más profundo de cómo funcionan las aplicaciones XNA y para facilitar un mayor entendimiento del código fuente de la aplicación Snake ILM, en este apartado se realiza una pequeña descripción de las funciones que conforman el esqueleto.

- **Initialize**. Primera función que se ejecuta al principio del juego una única vez, se cargan valores preliminares.
- **LoadContent**. Carga de gráficos, sonidos y demás contenidos.
- **HandleInput**. Se ejecuta como un bucle durante todo el ciclo de vida de la aplicación y se utiliza para la detección de las entradas externas, ya sean con teclado o pantalla táctil.
- **Draw**. Dibuja los valores actuales de los gráficos y sus posicionamientos en la pantalla del dispositivo. Se ejecuta de forma continua a cada paso de reloj del sistema.
- **Update**. Aquí entra toda la lógica y física del juego, ya sean cálculos de posicionamientos de gráficos o comprobaciones de intersecciones de

componentes de juego entre otros. De la misma manera que la función Draw, se ejecuta reiterativamente.

- **UnloadContent**. Se ejecuta una única vez al finalizarse la ejecución del juego y se utiliza para liberar los contenidos.

5.1.3. Entorno de desarrollo

En este tercer apartado se describe de forma detallada el entorno, tanto software como hardware, que se ha utilizado para desarrollar la aplicación Snake ILM.

5.1.3.1. Hardware

El software que se ha utilizado en el proceso de desarrollo es básicamente un ordenador portátil para realizar toda la codificación y compilación:

- **Ordenador portátil**. Intel Core Duo i5 a 1.8GHz y 4 GB de memoria RAM. El sistema operativo instalado se trata de un Windows 7 Standard Edition.

5.1.4. Software

A continuación se listan y se describen los recursos de software necesarios para llevar a cabo toda la parte de la codificación de la aplicación:

- **Visual Studio 2010 Ultimate con SP1**: entorno de desarrollo integrado donde se desarrollará y compilará el código fuente asociado a la aplicación Snake ILM.
- **XNA Game Studio 4.0**. Conjunto de herramientas que se integran dentro de Visual Studio 2012 que permiten desarrollar aplicaciones en XNA.
- **Microsoft XNA Framework 4.0 Refresh**. Contiene las librerías necesarias para poder desarrollar aplicaciones XNA.
- **Windows Phone SDK 7.1**. Herramientas y librerías necesarias para desarrollar aplicaciones para Windows Phone en su versión 7.1 como mínimo.
- **Google Drive**. Como repositorio de código fuente.
- **NinjaMock**. Herramienta web que permite realizar prototipos de aplicaciones móviles entre otras. Se describe más detalladamente en el apartado Prototipo.

- **MagicDraw**. Herramienta de modelado UML utilizada para generar los diferentes diagramas de clases, componentes y casos de uso especificados en la memoria.

5.2. Lógica del juego

5.2.1. Algoritmos principales

Para facilitar el proceso de desarrollo de toda la lógica del juego, se ha procedido a estructurar el problema principal del desarrollo de la lógica del juego Snake ILM en problemas más pequeños que permitan conseguir soluciones menos complejas. A continuación se describen algunos de los detalles de los algoritmos o procesos más importantes.

5.2.1.1. Movimiento de la cabeza

Como suele ser habitual, la cabeza de la serpiente en el juego Snake ILM se puede mover en 4 direcciones diferentes y solo se puede cambiar de dirección en un ángulo máximo de 90º.

Para resolver el problema del movimiento de la cabeza se ha optado por utilizar un sistema de movimiento únicamente con toques de pantalla dependiendo del movimiento actual de la serpiente y si se toca la pantalla a un lado u otro de esta.

5.2.1.2. Movimiento del cuerpo

Primeramente se pensó en realizar el movimiento de la cola o cuerpo de la serpiente de manera igual a como se hace con la cabeza pero surgió el problema de que los caminos que tienen que seguir cada una de las partes del cuerpo de la serpiente ya vienen predefinidas por los caminos que ha seguido la cabeza de esta. Por lo tanto, se debía implementar un sistema que guardase todos los cambios de dirección que hacia la cabeza de la serpiente. La clase *Route*, compuesta por una dirección y una posición, realizaría este trabajo.

En la clase *Update*, se llamará a la función *MoveBody* que se encargará de cambiar los posicionamientos de todas las partes del cuerpo de la serpiente y actualizándolos también en el caso de que haya un cambio de dirección.

Se realiza un cambio de dirección, por ejemplo, cuando uno de los cuerpos de la serpiente que está moviéndose hacia abajo supera la posición de las ordenadas marcada por el siguiente camino a tomar.

1. Para cada parte del cuerpo:
    a. Según la dirección que se esté moviendo, se comprueba si ha pasado del valor de una de las coordenadas de la siguiente ruta a tomar.
        i. Se actualiza la nueva posición y dirección.
        ii. Se elimina la ruta.

De esta manera, se actualiza la posición de esa parte del cuerpo, su nueva dirección y se elimina el camino de los caminos pendientes.

5.2.1.3. Acción de comer

Se ejecuta dentro de cada *Update* y comprueba que la cabeza y el ítem de la comida que existe en la pantalla realizan una intersección. Una vez sabemos esto, es necesario almacenar la posición y dirección actual del último ítem de la serpiente (cabeza o parte del cuerpo) y pasárselo por parámetro.

1. Calcular posición y dirección del último elemento de la serpiente.
2. Pasarle por parámetro a la serpiente la dirección y posición calculada actualmente.
    a. Actualizar la posición actual de la nueva parte el cuerpo según su dirección.
3. Actualizar las rutas pendientes del nuevo ítem copiándole las rutas pendientes de la parte del cuerpo predecesora a esta.

5.2.1.4. Controlador de velocidad

Para realizar los cálculos correspondientes al control de la velocidad de desplazamiento de la serpiente en la pantalla del juego se realiza una operación matemática concreta que marcará el número de pixeles que debe moverse la serpiente por ciclo de reloj. La operación o fórmula matemática encargada de realizar este trabajo dependerá del nivel de juego escogido en la pantalla de configuración del

juego Snake ILM y también del tamaño actual que tiene la serpiente. La fórmula utilizada es la siguiente:

*(tamaño + 1.0) / 20 + 0.3 * nivel*

Mediante esta fórmula matemática para calcular el desplazamiento de la serpiente por la pantalla podemos aumentar la dificultad progresivamente.

5.2.1.5. Comprobación de final del juego

Cualquier tipo de colisión de la cabeza de la serpiente contra un ítem de la pantalla que no sea los ítems que esta puede capturar para crecer, se consideran la finalización del juego. En este caso, tenemos dos posibles situaciones donde esto se puede producir:

1. Colisiones de la cabeza con alguna parte del cuerpo que no sea la primera.
2. Colisiones con los bordes de la zona de juego.

La comprobación de las colisiones de la cabeza ya sea con los bordes de la zona de juego, elementos del cuerpo de la serpiente o la propia acción de comer, se implementa a partir de las funcionalidades de intersección que tiene la clase *Rectangle* en XNA.

La función encargada de realizar estas comprobaciones se ejecutará tal y como hacen las demás, en cada iteración de la función *Update*, es decir, a cada ciclo de reloj se debe comprobar síncronamente si la partida ya ha finalizado o no.

6. Pruebas

Dentro de este apartado realizaremos las pruebas necesarias para asegurar una calidad óptima del producto final. Las pruebas que se describirán en el apartado siguiente tratarán de realizarse desde el punto de vista de un usuario final con el objetivo de cubrir todos los requerimientos especificados anteriormente en el análisis.

6.1. Entorno de pruebas

Antes de describir las diferentes pruebas a realizar, se describirá el entorno en el cual se han llevado a cabo dichas pruebas.

- Sistema Operativo: Windows Phone 7.8
- Hardware: Nokia Lumia 800

6.2. Pruebas funcionales

Partiendo de los requerimientos funcionales especificados dentro del apartado 3.1.1 se han diseñado una serie de pruebas funcionales para certificar que todos los requerimientos secundarios (REQX.X) y como principales (REQX) se cumplen y los resultados de las pruebas son positivos.

Nº	Descripción	Requerimiento	Resultado
1	Comprobar los enlaces a las diferentes pantallas del menú principal.	REQ1.1, REQ1.2, REQ1.3, REQ1.4	Positivo
2	Comprobar que la pantalla de “acerca de” muestra los datos correctamente.	REQ1.4	Positivo
3	Comprobar que el botón de Salir del juego funciona y muestra el diálogo de salida y salir pulsando start (botón central).	REQ1.5	Positivo
4	Comprobar que el botón de Salir del juego funciona y muestra el diálogo de salida y pulsar el botón atrás para volver.	REQ1.5	Positivo
5	Se entrará dentro del juego, se modificará la configuración y se saldrá del juego para volver a entrar en la configuración y comprobar que la dificultad se ha guardado.	REQ2.1, REQ1.4	Positivo
6	Comprobar el almacenado de las máximas puntuaciones.	REQ2.1, REQ1.4	Positivo
7	Con algunas máximas puntuaciones almacenadas. Entrar en la configuración y borrar las máximas puntuaciones. Comprobar que se han borrado.	REQ2.2, REQ1.4, REQ1.3	Positivo
8	Comprobar la carga de las máximas puntuaciones	REQ1.3	Positivo

9	Crear una nueva partida y comprobar que se muestra la caja de texto con el nombre y las instrucciones de la partida.	REQ3.1, REQ1.1	Positivo
10	En el menú de introducir el nombre, comprobar que se despliega correctamente el teclado de Windows Phone.	REQ3.2, REQ1.1	Positivo
11	Intentar comenzar una partida con el nombre vacío y comprobar que se muestra un mensaje de error en la pantalla.	REQ3.3, REQ1.1	Positivo
12	Crear una nueva partida. Cuando acabe, comprobar que la nueva máxima puntuación se ha guardado correctamente.	REQ7.8	Positivo
13	Dentro del juego. Comprobar todos los cambios de dirección de la serpiente.	REQ7.1, REQ7.2, REQ4.1	Positivo
14	Dentro del juego. Comprobar la actualización de los datos de la cabecera (tiempo, puntos, tamaño)	REQ4.2, REQ4.3	Positivo
15	Dentro del juego. Comprobar que la serpiente crece cuando come.	REQ7.3, REQ7.5	Positivo
16	Dentro del juego. Comprobar que la serpiente aumenta su velocidad de desplazamiento a medida que se hace más grande.	REQ7.4	Positivo
17	Dentro del juego. Comprobar el funcionamiento del menú de pausa pulsando el botón superior derecho.	REQ5, REQ4.4	Positivo
18	En el menú de pausa. Comprobar que se puede volver al juego y continuar desde donde se pausó.	REQ5, REQ5.2	Positivo
19	En el menú de pausa. Comprobar la opción de menú de volver al menú principal.	REQ5, REQ5.1	Positivo
20	En el menú de pausa. Comprobar la opción de menú de salir del juego.	REQ5, REQ5.3	Positivo
21	Dentro del juego. Comprobar que el choque de la serpiente con los bordes provoca la finalización del juego.	REQ7.6	Positivo
22	Dentro del juego. Comprobar de la cabeza de la serpiente con otras partes del cuerpo provoca la finalización de este.	REQ7.7	Positivo
23	Dentro del juego. Comprobar que la finalización del juego lanza la pantalla de *Game Over* con los datos correctos.	REQ6	Positivo
24	En la pantalla de *Game Over*. Comprobar que se puede iniciar una nueva partida con la opción de menú correspondiente.	REQ6, REQ6.2	Positivo
25	En la pantalla de *Game Over*. Comprobar que se puede volver al menú principal con la opción de menú correspondiente.	REQ6, REQ6.1	Positivo

7. Valoración económica

7.1. Recursos humanos

Antes de entrar de lleno en la valoración económica propiamente dicha, se definirán y se describirán los diferentes perfiles dentro de la planificación, análisis o desarrollo de las diferentes partes de Snake ILM.

En la tabla siguiente aparecen los diferentes perfiles que se necesitarían para llevar a cabo este proyecto y el número de personas necesarias de cada perfil.

Perfil	Número de Personas
Jefe de Proyecto	1
Analista/Programador	1
Programador Junior	2
Total	**4**

Es importante comentar que para la realizar todas las fases necesarias para obtener el producto final Snake ILM solo se ha necesitado una persona que ha llevado a cabo al mismo tiempo los tres perfiles y las diferentes tareas asignadas a cada uno de ellos, ya se la planificación del proyecto y supervisión por parte del Jefe de Proyecto, el análisis y diseño por parte del analista/programador y por último el desarrollo y las pruebas por parte de los dos programadores junior junto con el propio analista/programador.

7.2. Valoración económica

En este apartado se puede visualizar una estimación económica del proyecto de final de carrera de Snake ILM. Es importante comentar que esta estimación está realizada teniendo en cuenta que las jornadas laborales no son de 8 horas como tal, sino que en el caso particular de este trabajo de final de carrero se está partiendo de la base que cada jornada contabiliza 2 horas de trabajo real.

De esta manera, y partiendo de la planificación de proyecto especificada dentro de esta memoria, en la tabla siguiente se puede visualizar el importe según el perfil

asociado y el importe total del proyecto en lo que se refiere a las horas dedicadas a cualquiera de las fases de dicho proyecto. En la siguiente valoración no se tienen en cuenta los costes de las posibles licencias tales como las de Visual Studio y Windows 7 entre otras, necesarias para el terminal de desarrollo necesario.

Perfil	Coste/Hora	Jornadas	Horas	Importe
Jefe de Proyecto	65 €	8	16h	1040 €
Analista/Programador	50 €	25	50h	2500 €
Programador Junior	30 €	14x2	28hx2	1620 €
Total		**60**	**120h**	**5120 €**

8. Futuras mejoras

Una vez finalizado el desarrollo e implementación de la aplicación Snake ILM se han pensado en diversas mejoras propuestas tanto por el propio desarrollador del juego como por los usuarios que ya han probado la aplicación.

De esta manera, en este apartado de la memoria se van a especificar, algunas de las mejoras más interesantes que podrían darle un valor añadido a la aplicación en el futuro.

Pantallas

Para que la aplicación no resulte monótona y demasiado repetitiva para los usuarios la propuesta de disponer diferentes pantallas para jugar podría ser una buena idea, por ejemplo:

- Pantallas con elementos estáticos que conformen un obstáculo para el usuario y la serpiente.
- Pantallas que contengan elementos móviles, que se vayan moviendo siguiendo un patrón concreto.

Opciones de configuración

Sería interesante añadir algunas opciones más para poder configurar el juego.

- Selector de color para la serpiente.
- Tamaño de la serpiente. Es posible que con el Framework de XNA y con las limitaciones de este no sea trivial conseguir configurar la serpiente para que se pueda configurar su tamaño.

Movimiento de la serpiente

Existen varios puntos en los cuales se podrían realizar diversas modificaciones para llevar a cabo.

- El movimiento de la serpiente cuando aumenta progresivamente de velocidad esta conseguido, quizás

- Mejorar sensación de movimiento de la serpiente durante el juego. Actualmente tiene un movimiento demasiado articulado. Sería interesante suavizar los cambios de dirección de la serpiente.

Nuevos ítems

Sería buena idea añadir nuevos ítems para mejorar la experiencia de juego.

- Elemento paralizador.
- Elemento que puntúa el doble.
- Multiplicador de puntuación por tiempo limitado.

Sonido

Actualmente el juego no dispone de ninguna clase de sonido por lo que en una futura actualización ciertos sonidos podrían dar a la aplicación algo de dinamismo.

- Añadir sonido al tocar las opciones de menú de cualquiera de las pantallas.
- Añadir un audio que dinamice la partida dentro del juego.
- Añadir sonidos de acciones concretas como cuando la serpiente come o se termina el juego.

9. Conclusiones

En términos generales se pueden extraer una serie de conclusiones sobre el trabajo de final de grado en sí y también sobre la plataforma para desarrollo de juegos llamada XNA.

Primeramente, comentar que se han comprendido e interiorizado las bases y concepto de programación mediante XNA, ya sea comprender el ciclo de vida de las aplicaciones programadas en esta plataforma o las particularidades del propio SDK como los elementos encargados de proporcionar la funcionalidad base o los propios elementos para dibujar en la pantalla del dispositivo.

En segundo lugar, centrándonos en los objetivos propuestos al principio del trabajo, estos se han conseguido totalmente, tanto de toda la parte de planteamiento y planificación del trabajo como de la parte del propio desarrollo e implementación y el posterior testeo para asegurar la calidad final del producto obtenido. Especial mención para el descubrimiento de la herramienta *NinjaMock* que ha facilitado de manera notable el diseño de la aplicación dentro del propio código fuente del juego. Esto sin olvidar el kit de inicio Game State Management en el que se ha basado la solución y a partir del cual se ha podido estructurar de forma más correcta y menos anárquica el código fuente del juego.

Entrando más en detalle en la planificación del proyecto, está se ha seguido de la manera marcada al inicio pero se ha tenido que realizar un pequeño re-análisis de la solución inicial propuesta como la adición de un nuevo elemento llamado Ruta, el cual era absolutamente necesario para la consecución de la solución final. Aún con este cambio en la arquitectura del juego, no ha sido necesario re-planificar los tiempos de desarrollo en ningún caso.

Finalmente y como comentario personal comentar que un siendo XNA una plataforma de desarrollo descontinuada por parte de Microsoft a principios del año 2013, ha sido una experiencia enriquecedora debido a que realizar desarrollos de este tipo no es lo más normal en el mundo donde nos movemos la mayoría de estudiantes del Grado de Ingeniería Informática.

10. Bibliografía

http://blog.flurry.com/bid/103601/Mobile-Use-Grows-115-in-2013-Propelled-by-Messaging-Apps

http://aprendiendoxna.wordpress.com/articulos/xna/el-framework-de-xna/

http://www.widget-101.com/juegos/xna-juegos/xna-hola-mundo/

http://www.microsoftvirtualacademy.com/training-courses/desarrollo-en-xna

http://www.xatakawindows.com/actualidad-en-redmond/las-aplicaciones-enviadas-a-la-store-suben-un-40-desde-el-lanzamiento-de-windows-phone-8

http://jsaenzr.com/blog/2012/01/04/porqu-deberas-de-desarrollar-para-windows-phone-en-el-2012/

http://www.ninjamock.com

http://msdn.microsoft.com/en-us/gg258447.aspx

http://redcatdev.wordpress.com/tag/xna/

Anexo A: Manual de instalación

Requisitos de instalación

Para instalar la aplicación es necesario tener una máquina con Windows 7 instalado. El Framework de XNA no soporta Windows 8 y versiones superiores. A continuación se listan una serie de pasos que se han de seguir para poder llevar a cabo la instalación de Snake ILM de forma correcta.

Windows Phone 7.x

Se deben seguir los siguientes pasos para poder instalar la aplicación Snake ILM dentro de un terminal móvil con sistema operativo Windows Phone 7.x.

- Instalar ZUNE 4.8. Es necesario instalarlo de lo contrario no reconocerá el terminal móvil conectado vía USB.
  http://www.microsoft.com/en-us/download/details.aspx?id=27163
- Instalar XNA Game Studio 4.0. Es necesario para poder abrir y desempaquetar los archivos de tipo .ccgame (Creator club Game) generados por Visual Studio.
  http://www.microsoft.com/en-us/download/details.aspx?id=23714
- Reiniciar sistema.
- Ejecutar Zune 4.8. Si no se ejecuta, el móvil no será visible.
- Conectar teléfono vía USB. Se instalarán los drivers correspondientes y el teléfono será visible desde Zune.

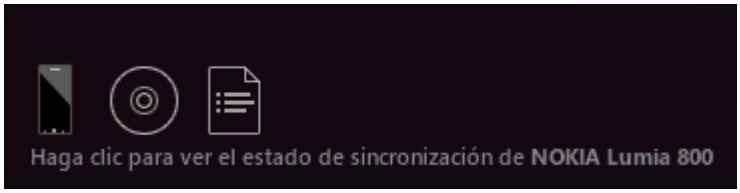


Figura 23

- Desbloquear el terminal móvil. Si no se desbloquea, la aplicación encargada de desempaquetar el juego e instalarlo en el dispositivo no funcionará correctamente.

- Descomprimir el fichero con extensión *.rar "MironLopezDavid_PAC3 (Snake Installator).rar"* en cualquier carpeta del sistema de archivos de Windows. Este archivo comprimido contiene los datos y ficheros necesarios para realizar la instalación de la aplicación Snake ILM en el terminal móvil de forma correcta.
- Ejecutar el fichero *install.bat* de instalación del juego. La consola nos devolverá lo siguiente:

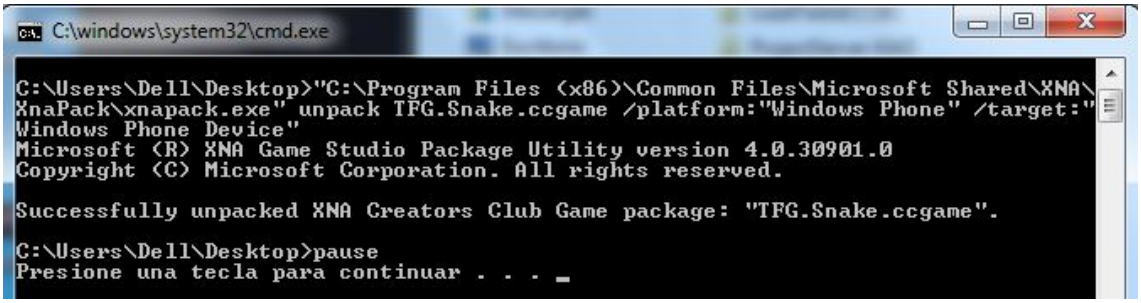


Figura 24

Nos informa de que el juego se ha instalado satisfactoriamente dentro del terminal móvil.

Windows Phone 8

La instalación para el sistema operativo Windows Phone 8 es similar a la de Windows Phone 7 pero en este caso pueden evitarse varios de los pasos ya que en este SO consta de drivers Plug and Play.

1. Instalar XNA Game Studio 4.0.
2. Conectar teléfono vía USB. Se instalarán los drivers correspondientes será visible desde el explorador de Windows.
3. Desbloquear el terminal móvil.
4. Descomprimir el fichero con extensión .rar *"MironLopezDavid_PAC3 (Snake Installator).rar"* en cualquier carpeta del sistema de archivos de Windows. Este archivo comprimido contiene los datos y ficheros necesarios para realizar la instalación de la aplicación Snake ILM correctamente.
5. Ejecutar el fichero *install.bat* de instalación del juego. La consola nos devolverá lo siguiente:
6. La aplicación ya está disponible dentro del terminal móvil para poder utilizarla.

Anexo B: Manual de usuario

Pantalla principal

En la pantalla principal se pueden ver las diferentes opciones que tenemos dentro de Snake ILM.

Figura 25

1 – Nueva partida: Con esta opción accederemos a la pantalla para introducir el nombre de usuario y poder crear una nueva partida.

2 – Configuración: Opción para acceder a la pantalla de configuración del juego.

3 – Ver puntuaciones: Aquí podremos ver las 8 máximas puntuaciones conseguidas.

4 – Acerca de Snake: Se podrá visualizar la información de Snake ILM tal como la versión, licencia, etc.

5 – Salir del juego: Como su nombre indica, con esta opción saldremos del juego no sin antes aceptar el mensaje de confirmación correspondiente.

Configurar el juego

En la pantalla de configuración del juego se podrán modificar opciones para adaptar la dificultad del juego a nuestro gusto y también borrar las máximas puntuaciones almacenadas actualmente.

Figura 26

1 – Configurar dificultad: Tocando encima del texto que especifica el nivel de dificultad, se podrá modificar este nivel. Existen 5 niveles diferentes (Muy fácil, fácil, normal, difícil y muy difícil). El nivel de dificultad implica que la velocidad de movimiento será mayor o menor en desde el comienzo de la partida.

2 – Borrar puntuaciones: Tal y como su nombre indica, permite al jugador borrar las máximas puntuaciones conseguidas. Esta acción no permitirá volver a recuperar las puntuaciones que se habían guardado anteriormente.

Crear una nueva partida

Para crear una nueva partida se debe utilizar la opción indicada para esto en la pantalla principal. Antes de empezar a jugar, se introducirá el nombre de usuario correspondiente tocando encima del recuadro blanco (1), el cual desplegará el teclado en pantalla de Windows Phone. Una vez se introduce el nombre, empezará la partida después de tocar el botón de Comenzar (2).

En último lugar, en la parte inferior aparece un pequeño texto informativo (3) a modo de recordatorio con instrucciones de cómo mover a la serpiente de Snake ILM.

Figura 27

Pantalla de juego

En la imagen de la parte inferior se muestra la interfaz de la pantalla del juego y la diferente información y componentes señalizada y que se listan a continuación:

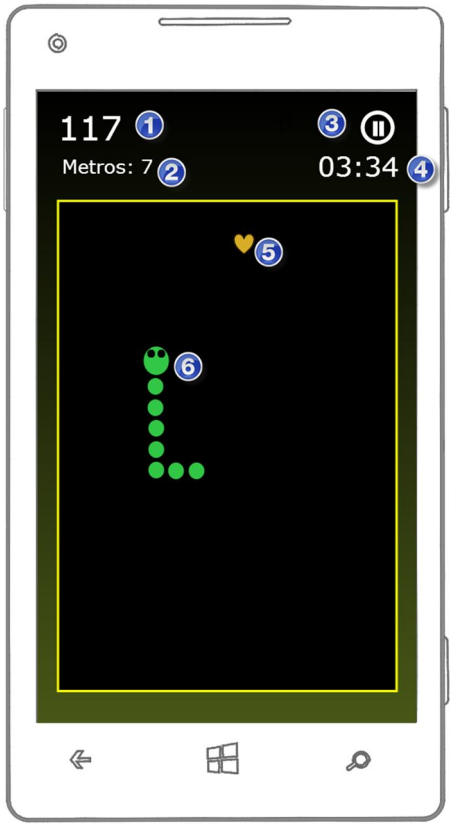


Figura 28

1 – Puntuación.

2 – Tamaño de la serpiente.

3 – Botón de pausa.

4 – Tiempo de juego transcurrido.

5 – Elemento de juego de la comida.

6 – Serpiente del juego que controla el jugador.

Como jugar

En la pantalla de introducción del nombre, se muestra al usuario un pequeño resumen a modo de recordatorio de cómo jugar a Snake ILM.

Una vez dentro de la pantalla de juego el objetivo es recoger con la serpiente los diferentes ítems que irán apareciendo por la pantalla de juego. La serpiente se moverá de la manera siguiente:

1. Si la serpiente se está moviendo de manera vertical (arriba o abajo) se puede cambiar su dirección 90º, es decir, hacia la izquierda o la derecha. Si se toca la pantalla de juego al lado izquierdo o derecho de la serpiente, esta modificará su movimiento y empezará a desplazarse hacia la izquierda o derecha respectivamente.

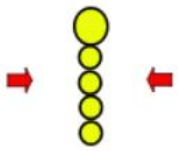


Figura 29

2. Si el desplazamiento de la serpiente es en horizontal (izquierda o derecha) se modificar la dirección hacia arriba o abajo tocando la pantalla por la parte superior o inferior de la serpiente respectivamente.

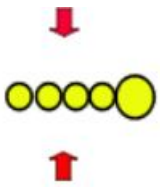


Figura 30

Final del juego

El juego llegará a su fin cuando la serpiente controlada colisione con su cabeza en los límites de la zona de juego o cuando colisione con ella misma, es decir, que la cabeza no coincida en un mismo lugar con alguna parte del cuerpo.

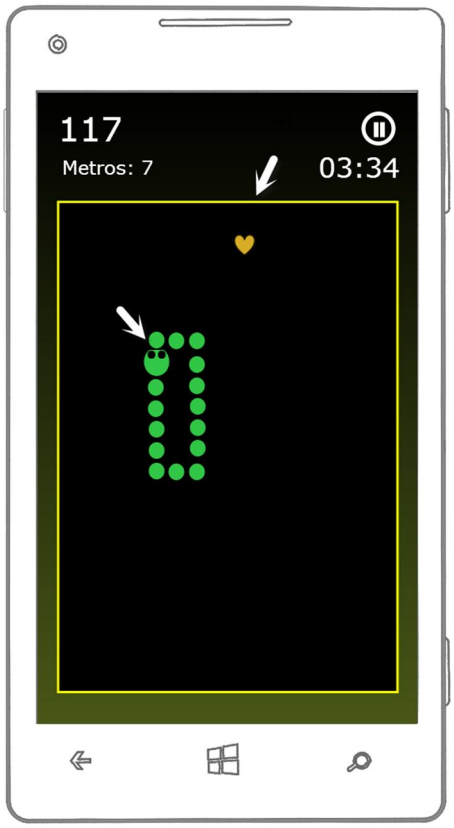


Figura 31

Una vez terminada la partida, se mostrará por pantalla la puntuación final obtenida por el usuario y se ofrecerá la posibilidad de comenzar una nueva partida o volver al menú principal.