Universitat Oberta de Catalunya UOC

Ingeniería técnica en informática de sistemas

Proyecto fin de carrera - Desarrollo de aplicaciones para dispositivos móviles (Android)

Xavier Figuera Alberich

31 de Diciembre de 2013

Universitat Oberta de Catalunya UOC

Ingeniería técnica en informática de sistemas

Proyecto fin de carrera - Desarrollo de aplicaciones para dispositivos móviles (Android)

Desarrollo de videojuegos para Android con libGDX
(Casual games)
Fling the sheep

http://projects.3comet.com/

Consultores: Marc Domingo Prieto
Jordi Almirall López

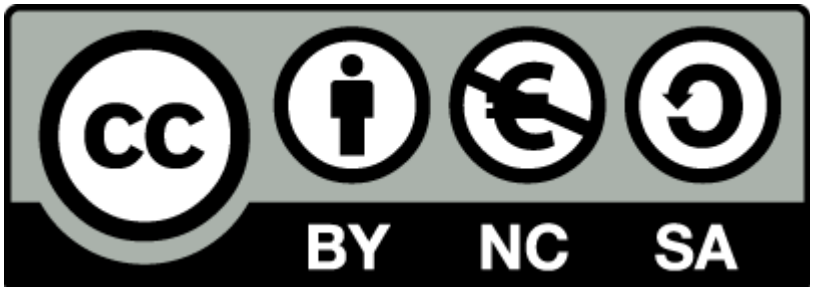
Autor: Xavier Figuera Alberich
Correo: xfiguera@uoc.edu

Sabadell, 31 de Diciembre de 2013

A mi pareja Remei Solé, por su paciencia y apoyo durante estos últimos meses y siempre, sin ella no hubiera sido posible llegar hoy a escribir estas lineas, te quiero.

A la vida por haberme brindado la posibilidad de poder hacer lo que realmente siento dentro de mi.

Licencia



Resumen

Android es un sistema operativo con núcleo monolítico basado en el kernel de Linux version 2.6 que fue lanzado al mercado el 23 de septiembre de 2008. Diseñado principalmente para dispositivos móviles con pantalla táctil, como pueden ser Smartphones o tabletas. Esta disponible gratuitamente para un uso comercial o no comercial. El núcleo esta escrito en C, algunas de las librerías de terceros están escritas en C++ y su interfaz de usuario esta escrita en Java. Soporta distintas estructuras de procesador, entre ellas ARM, x86, MIPS y IBM POWER, no obstante la arquitectura utilizada mayoritariamente son los procesadores ARM. Esta publicado bajo la licencia Apache 2.0 y GNU GPL2.

En el momento de escribir este documento su versión actual es la Android 4.3 Jelly Bean (Gominola) y se encuentra en permanente desarrollo. El primer móvil con Android fue el HTC Dream que se comercializo por primera vez en Octubre de 2008.

Android fue desarrollado inicialmente por la compañía Android Inc, esta estuvo financiada por Google durante un tiempo hasta que la compañía fue adquirida por la multinacional estadounidense en julio de 2005. Google quería tener presencia en el mercado de los dispositivos móviles. Android fue presentado en 2007 conjuntamente con la fundación Open Handset Alliance. Esta entidad, actualmente es un consorcio de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles, liderada por Google. La multinacional estadounidense es el principal desarrollador de Android y aporta el ecosistema necesario para el crecimiento de este. No obstante todos los miembros del consorcio contribuyen de distintas maneras al desarrollo.

En el año 2008 salio a la luz la versión 1.0 de Android. En el momento de su aparición tubo que enfrentarse a las distintas plataformas ya existentes en aquel momento, iPhone OS que actualmente es conocido como iOS, BlackBerry OS y Windows Phone 7. El crecimiento de Android a sido espectacular en estos últimos años, a nivel mundial alcanzó una cuota de mercado del 50,9% durante el último trimestre de 2011, esto represento más del doble que el sistema operativo iOS de Apple. Actualmente, su cuota de mercado se sitúa en el 74,4% según un estudio realizado por la empresa Gartner. Ver la siguiente tabla:

Table 3
Worldwide Smartphone Sales to End Users by Operating System in 1Q13 (Thousands of Units)

Operating System	1Q13 Units	1Q13 Market Share (%)	1Q12 Units	1Q12 Market Share (%)
Android	156,186.0	74.4	83,684.4	56.9
iOS	38,331.8	18.2	33,120.5	22.5
Research In Motion	6,218.6	3.0	9,939.3	6.8
Microsoft	5,989.2	2.9	2,722.5	1.9
Bada	1,370.8	0.7	3,843.7	2.6
Symbian	1,349.4	0.6	12,466.9	8.5
Others	600.3	0.3	1,242.9	0.8
Total	**210,046.1**	**100.0**	**147,020.2**	**100.0**

Source: Gartner (May 2013)

El hecho de que Android sea de código abierto, permite a los fabricantes comercializar dispositivos de distintas gamas y precios adaptando ellos mismos el sistema operativo según las necesidades de cada modelo. De esta forma se consigue un alcance mayor a distintos niveles de publico, esto seria uno de los motivos que ha causado en estos últimos años el crecimiento espectacular de cuota de mercado llegando casi al 75% actualmente a nivel mundial.

Android a sufrido numerosas actualizaciones desde su liberación, cada versión liberada se le ha

dado un nombre de postres en ingles por orden alfabético.

En la siguiente tabla se enumeran las distintas versiones salidas a la luz desde su lanzamiento inicial en el año 2008:

Letra	Nombre del Postre - Versión
A	Apple Pie (v1.0), Tarta de manzana
B	Banana Bread (v1.1), Pan de plátano
C	Cupcake (v1.5), Panque.
D	Donut (v1.6), Rosquilla.
E	Éclair (v2.0/v2.1), Pastel francés.
F	Froyo (v2.2), (Abreviatura de «frozen yogurt») Yogur helado.
G	Gingerbread (v2.3), Pan de jengibre.
H	Honeycomb (v3.0/v3.1/v3.2), Panal de miel.
I	Ice Cream Sandwich (v4.0), Sándwich de helado.
J	Jelly Bean/Gummy Bear (v4.1/v4.2/v4.3), Gominola. **(versión actual)**
K	KitKat (v4.4), Kit Kat. **(versión en desarrollo)**

Índice de contenidos

Índice de ilustraciones

1.Introducción

1.1 Justificación y contexto del proyecto

Este trabajo de fin de carrera en adelante TFC, se centra en el desarrollo de aplicaciones para dispositivos móviles con sistema operativo Android. Actualmente existen multitud de aplicaciones. No obstante, este TFC se quiere centrar en un subconjunto de aplicaciones focalizadas al entretenimiento conocidas por todo el mundo como videojuegos.

Un videojuego es un software creado para el entretenimiento en general y basado en la interacción entre una o varias personas y un aparato electrónico que ejecuta dicho videojuego. Así pues un videojuego es considerado un software cuyo fin directo es el entretenimiento.

En estos últimos años los smartphones y tabletas se han convertido en las nuevas plataformas móviles para juegos que compiten con los sistemas portátiles clásicos que generalmente tienden a ser más cerrados.

Los videojuegos se pueden clasificar en distintos géneros, estos son una forma de clasificación, que designan un conjunto de videojuegos que poseen una serie de elementos comunes. A lo largo de la historia de los videojuegos aquellos elementos que han compartido varios de los mismos han servido para clasificar como un género a aquellos que les han seguido en estilo y forma, de la misma manera que ha pasado con la música o el cine. Como ejemplo se enumeran algunos de los géneros más populares: Lucha, Disparos conocidos como Shooters, Sigilo, Plataformas, Arcade, Deporte etc.

Durante mucho tiempo los videojuegos estuvieron reservados para un tipo de usuario muy concreto con un perfil muy específico: la gran mayoría eran hombres entre 15 y 25 años. Saliendo de este perfil, si se buscaban mujeres o personas mayores de 25 años, el número de usuarios caía en picado y es que los videojuegos eran percibidos como el hobby de los adolescentes masculinos. No obstante, en los últimos años se ha visto cómo todo esto ha ido cambiando con un aumento tanto del público femenino, como la aparición en escena de usuarios con edades más avanzadas.

Este cambio ha venido motivado por diferentes motivos evolutivos dentro de la industria del videojuego. Uno de los motivos que ha provocado esto es la aparición de los teléfonos móviles inteligentes, tabletas y juegos online, junto con gran cantidad de videojuegos de tipo casual. Otro claro motivo es que la mayoría de los que jugábamos hace 25 años, pues hoy seguimos jugando ya que hemos crecido con esta cultura y claro esta que a día de hoy ya no tenemos la edad de entonces.

1.2 Descripción del Proyecto

Este proyecto se quiere centrar en el desarrollo de un videojuego de tipo casual para móviles Android. Estos son un tipo de videojuegos dirigidos a un público de jugadores casuales. Los videojuegos casuales pueden tener cualquier tipo de mecánica de juego, y ser clasificados dentro de cualquier otro género. Son diferenciados por sus reglas simples y de no requerir un excesivo compromiso en contraste con la mayoría de juegos, más complejos. No requieren una dedicación de tiempo a largo plazo o habilidades especiales para jugar.

Durante la realización del proyecto, se realizara el diseño del juego, por diseño se entiende realizar un documento que defina el juego de forma generalizada: mecánicas, personajes, etc. este documento es conocido como el "game design", el diseño de arte conocido como "Concep art" engloba todo lo relacionado con el aspecto gráfico del juego, estilos de personajes, estilos de gráficos etc., Finalmente se realizara el análisis, diseño e implementación de la aplicación(videojuego), con las tecnologías escogidas para este proyecto.

1.3 Objetivos

El objetivo principal de este TFC es adquirir experiencia en el desarrollo de videojuegos para Android. Sin embargo, yendo más allá de la plataforma, también existe un interés por ampliar conocimientos en cuanto a programación y diseño de videojuegos en general.

En cuanto al diseño y desarrollo del videojuego, se pretende realizar un casual game con las siguientes características:

El juego consistirá en conducir unas ovejas descarriadas a su corral, éstas con nuestra ayuda deberán sortear los peligros que irán apareciendo durante el camino hacia el deseado corral. El principal peligro que amenaza las ovejas son unos lobos malvados que quieren comérselas. El videojuego sólo dispondrá de una única modalidad de juego para un único jugador. Al tratarse de un juego casual el publico objetivo es muy amplio. Se pretende conseguir un "look & feel" con estética tipo cartoon en clave de humor. El juego sera desarrollado en 2D con perspectiva cenital.

1.3.1 Definición del videojuego

El nivel se describe como una pantalla estática con vista cenital en posición vertical. Con estática se entiende que no hay ningún tipo de scroll que haga desplazar la pantalla. Siempre veremos lo mismo.

En la parte de arriba del todo aparecerá un corral con una puerta, la puerta se irá abriendo y cerrando de forma aleatoria. Por los extremos de la pantalla habrá arbustos que representaran los limites del bosque. De estos limites, irán saliendo lobos a la escena que se querrán comer a las pobres ovejas. Estas ovejas permanecerán en la parte de abajo de la pantalla e irán saliendo a la escena ellas solas sin saber demasiado donde van(son ovejas descarriadas), nuestra misión consistirá en tirarlas hacia el corral para salvarlas de los malvados lobos, metiendolas por la puerta en los momentos que este abierta, para lanzarlas, tendremos que deslizar el dedo encima de ellas para darles impulso hacia el corral, si no acertamos en meter la oveja en la puerta, esta rebotara como si de un pelota se tratase. Los lobos irán saliendo, para sacarnos los lobos de encima, deberemos deslizar nuestro dedo encima de ellos o tocarlos para ahuyentarlos.

De lobos habrá de diferentes colores , marrones , grises y azulados . Cada lobo lo ahuyentaremos de manera distinta:

- **Marrones**: 1 pulsación o deslizar 1 vez el dedo sobre el lobo.
- **Grises:** 2 pulsaciones o deslizar 2 veces el dedo sobre el lobo.
- **Azulados:** 3 pulsaciones o deslizar 3 veces el dedo sobre el lobo.

A parte de impedir que se coman las ovejas que están en camino hacia el corral, también tendremos que evitar que los lobos lleguen a la parte de abajo donde está el grupo de ovejas, ya que esto nos hará perder vidas, el juego inicialmente empezara con 3 vidas, pudiendo alcanzar un máximo de 6 vidas en total, cada 10000 puntos obtendremos un vida extra.

Conforme el juego se va desarrollando, éste irá acumulando:

- Puntos.
- Tiempo jugado.
- Numero de lobos ahuyentados.
- Ovejas colocadas en el corral.

Se tratara pues de ir mejorando nuestro propio récord partida tras partida. El juego irá aumentando de dificultad durante el transcurso de la partida.

Como apunte final el juego recibe el nombre de “Fling the sheep” que traducido al español vendría a ser algo como “lanzar las ovejas”.

1.4 Planificación

1.4.1 Ciclo de vida

Un videojuego no deja de ser un proyecto de software, por tanto para la realización de este proyecto se ha tomado como base el ciclo de vida clásico de cualquier proyecto de software, modificado ligeramente a las necesidades del proyecto. Este ciclo de vida, que también se denomina ciclo de vida en cascada se compone de las siguientes etapas:

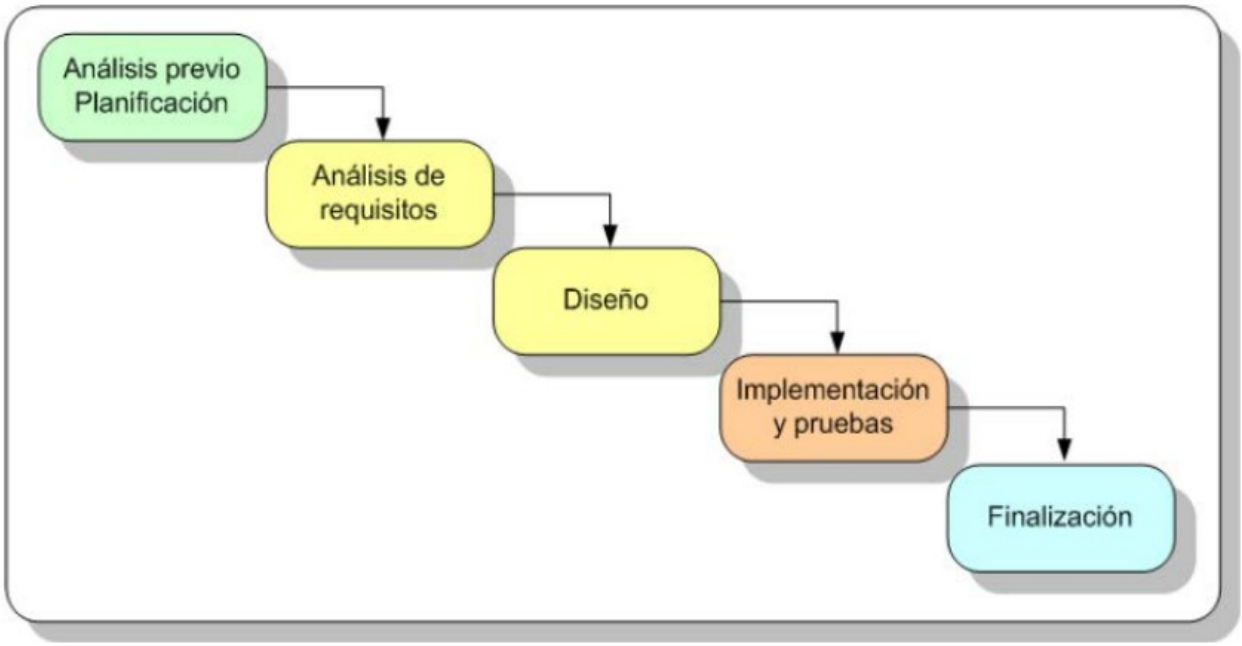


ilustración 1. Ciclo de vida en cascada

1.4.2 Planificación detallada del proyecto

En los siguientes apartados se recogen, por cada etapa que compone el ciclo de vida, las principales actividades realizadas con el objetivo de alcanzar el producto deseado dentro de los plazos definidos.

1.4.1.1 Análisis previo planificación

En este punto se ha abordado la visión global del proyecto, comprendida en establecer los puntos iniciales necesarios para crear una planificación de tareas en función de las actividades a realizar.

Actividad	Descripción
Diseño conceptual del videojuego	Mecánicas del videojuego
Definición de las tecnologías	Tecnologías que se utilizaran en el desarrollo del videojuego
Análisis de riesgos	Posibles riesgos del proyecto
Planificación del proyecto	Crear una planificación adaptada a los plazos de entrega de este TFC

Redacción del documento	Generación del plan de trabajo.
Entrega de la PEC1	Entrega del documento.

1.4.1.2 Análisis de requisitos

En esta fase se ha detallado que necesidades debe cubrir el videojuego, tanto desde un punto de vista funcional, como desde un punto de vista arquitectónico de la aplicación a diseñar, se ha diseñado un modelo troncal de clases para posteriormente desarrollar en base a estas, todo el videojuego. No obstante, en ningún caso se ha entrado en detalle en las tecnologías a utilizar ya definidas previamente.

Actividad	Descripción
Especificación de requisitos funcionales	Definición de casos de usos.
Diseño básico estructura aplicación	Diagrama básico de clases.
Especificación de requisitos funcionales	Definición de máquinas de estado

1.4.1.3 Diseño

En este apartado, teniendo en cuenta que en este proyecto se va a desarrollar un videojuego, se ha realizado la recopilación conceptual de arte, conjuntamente con el diseño de todas las pantallas que intervienen en el. Conjuntamente con la selección de los tipos de fuente que se utilizaran para el videojuego, selección de personajes y enemigos.

Actividad	Descripción
Definición de usuarios y contexto de uso del videojuego	Conocer las características de los usuarios, sus necesidades y objetivos, así como los contextos de uso.
Diseño conceptual	Un escenario de uso describe desde el punto de vista del usuario como será utilizado un producto determinado (es este caso un videojuego) en un escenario determinado.
Selección de arte	Selección de arte ya hecho con licencias libres ya que no se disponen de conocimientos de arte para abordar el diseño.
Selección de fuentes	Selección de los tipos de fuente adecuados para el diseño del videojuego.
Selección de personajes y enemigos	Selección de personajes y enemigos disponibles en la red con licencias libres ya que no se disponen de conocimientos de arte para abordar el diseño.
Diseño de pantallas que intervienen(prototipo)	Con todo el arte, fuentes y personajes, se realizara el diseño de todas las pantallas que intervienen en el juego. Realización de un boceto a mano alzada, Sketches y prototipo de alta fidelidad.
Redacción del documento	Generación del documento de diseño

Entrega de la PEC2	Entrega del documento.

1.4.1.4 Implementación y pruebas

En este punto se ha desarrollado la implementación del videojuego partiendo de todas las definiciones de los apartados anteriores y las verificaciones oportunas para asegurar el correcto funcionamiento del videojuego.

Actividad	Descripción
Implementación de navegables y contenidos	Implementación de la estructura de todas la pantallas del video juego y la navegabilidad entre ellas y las funciones principales de cada una de ellas.
Implementación del gameplay	Implementación de la logica de juego.
Implementación de sonido y efectos	Implementación de los efectos de sonido y la musica.
Version alpha del videojuego	En este punto se dispone de un videojuego funcional, listo para ser revisado y refinado.
Pruebas y refinamiento de la implementación	Pruebas y correciónes de posibles errores.
Preparación del paquete entregable y la redacción del documento	Testeo final y preparación del apk a entregar, mas la redacción del documento estrutural de l'aplicación.
Entrega de la PEC3	Entrega del documento.

1.4.1.5 Finalización

En un proyecto de software esta parte quedaría focalizada en la fase de implantación y puesta en marcha. Sin embargo, al tratarse de un videojuego, esta fase quedaría focalizada en la fase de lanzamiento del producto. No obstante en este caso se ha utilizado para documentar todo el trabajo realizado a lo largo de este TFC.

Actividad	Descripción
Redacción memoria final proyecto	Redacción memoria final
Montaje presentación diapositivas	Montaje presentación diapositivas
Montaje del video final presentación proyecto	Montaje del video final presentación proyecto
Entrega final	Entega final

1.4.3 Temporalización del proyecto

La siguiente ilustración muestra el calendario definido para el proyecto.

Nombre de tarea	Duración	Comienzo	Fin
Análisis previo planificación	**9 días**	**lun 23/09/13**	**mar 01/10/13**
Diseño conceptual del videojuego	1 día	lun 23/09/13	lun 23/09/13
Definición de las tecnologías	2 días	mar 24/09/13	mié 25/09/13
Análisis de riesgos	2 días	jue 26/09/13	vie 27/09/13
Planificación del proyecto	1 día	sáb 28/09/13	sáb 28/09/13
Redacción del plan de trabajo PEC1	2 días	dom 29/09/13	mar 01/10/13
Entrega PEC1 (Hito)	1 día	mar 01/10/13	mar 01/10/13
Analisis de requisitos	**5 días**	**mié 02/10/13**	**dom 06/10/13**
Definición de casos de uso	2 días	mié 02/10/13	jue 03/10/13
Definición de maquinas de estado	2 días	vie 04/10/13	sáb 05/10/13
Diseño básico estructura aplicación	1 día	dom 06/10/13	dom 06/10/13
Diseño	**23 días**	**lun 07/10/13**	**mar 29/10/13**
Definición de usuarios y contextos de uso del videojuego	2 días	lun 07/10/13	mar 08/10/13
Diseño conceptual	2 días	mié 09/10/13	jue 10/10/13
Selección de arte	4 días	vie 11/10/13	lun 14/10/13
Selección de fuentes	2 días	mar 15/10/13	mié 16/10/13
Selección de personajes y enemigos	2 días	jue 17/10/13	vie 18/10/13
Diseño de pantallas que intervienen (prototipo)	8 días	sáb 19/10/13	sáb 26/10/13
Redacción del documento PEC2	3 días	sáb 26/10/13	mar 29/10/13
Entrega PEC2 (Hito)	1 día	mar 29/10/13	mar 29/10/13
Implementación y pruebas	**42 días**	**mié 30/10/13**	**mar 10/12/13**
Implementación de navegables y contenidos	15 días	mié 30/10/13	mié 13/11/13
Implementación gameplay	15 días	jue 14/11/13	jue 28/11/13
Implementación de sonido y efectos	4 días	vie 29/11/13	lun 02/12/13
Pruebas y refinamiento de la implementación	5 días	mar 03/12/13	sáb 07/12/13
Preparación del APK y redacción de la PEC3	3 días	sáb 07/12/13	mar 10/12/13
Entrega PEC3 (Hito)	1 día	mar 10/12/13	mar 10/12/13
Finalización	**29 días**	**mié 11/12/13**	**mié 08/01/14**
Redacción memoria final proyecto	12 días	mié 11/12/13	dom 22/12/13
Montaje presentación diapositivas	2 días	vie 20/12/13	sáb 21/12/13
Montaje del video final presentación proyecto	10 días	dom 22/12/13	mié 01/01/14
Revisión y entrega final (Hito)	7 días	jue 02/01/14	mié 08/01/14
Total temporización proyecto	**108 días**	**lun 23/09/13**	**mié 08/01/14**

ilustración 1.1. Calendario proyecto

En la siguiente pagina se muestra una ilustración del diagrama de gantt correspondiente al calendario mostrado en la ilustración 1.

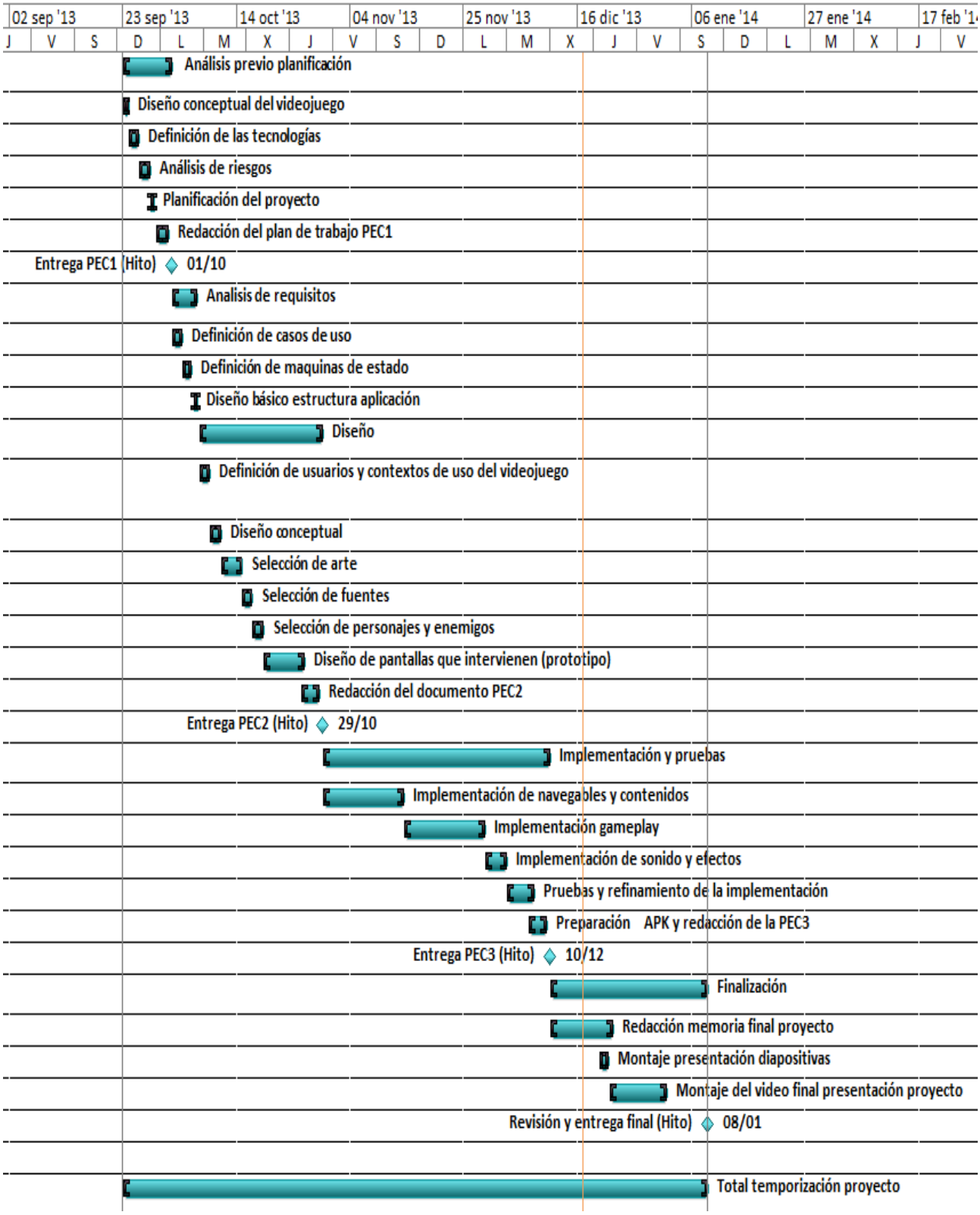


ilustración 2. Diagrama de gantt

1.5 Herramientas y tecnologías utilizadas

Actualmente existen varios frameworks para desarrollar videojuegos sobre plataformas móviles. Se ha hecho un análisis comparativo preliminar entre Cocos2D-android1, Cocos2D-x, y libGDX, aunque existen otros, estos tres son los más nombrados y utilizados.

Cocos2d-android1: Es un framework en Java basado en la versión de cocos2d-android que esta basado en la versión de cocos2d para iphone. Este framework permite el desarrollo de videojuegos en 2D, y se programa en Java en lugar de objective-c. Para la versión Cocos2d de iphone. Actualmente este framework no parece tener mucha continuidad por parte de la comunidad de desarrolladores, aunque no está discontinuado del todo.

Cocos2D-x: Es un framework en C++ multi-plataforma, permite desarrollar para diferentes plataformas a la vez con el mismo código: Android, iOS, Bada, Marmalade, Windows, Linux, etc. Sin embargo, aunque es un framework con una comunidad muy activa, su documentación es bastante mejorable. Se programa en C++ y es posible trabajar con distintos entornos de desarrollo como Visual Studio o Eclipse. El proceso de depuración con eclipse es algo complejo con C++/JNI con varios pasos a seguir para poder depurar. A la vez, su configuración en el entorno Eclipse no es trivial. Finalmente ,tiene relativamente todavía poco tiempo de vida.

LibGDX: Es un framework escrito en Java sobre C++, ya que las partes críticas están escritas con código nativo para conseguir el mejor rendimiento posible. Tiene una comunidad de desarrolladores muy activa y una documentación muy extensa. Se programa con Java y permite la depuración ejecutando la aplicación en el escritorio de la máquina sin tener que pasar por un dispositivo real o uno virtual. En caso de que se quiera probar en un dispositivo real es posible hacerlo sin ningún cambio sobre el código. Este hecho hace que el desarrollo sea más fluido. Tiene un rendimiento bastante alto ante cocos2d-x y otros frameworks existentes para el desarrollo de videojuegos.

Todo lo expuesto anteriormente provoca una inclinación definitiva a utilizar libGDX para implementar el videojuego. Así pues, el videojuego será escrito en Java utilizando el IDE Eclipse que es el soportado oficialmente para Android utilizando el framework libGDX.

Para la realización de este proyecto se han utilizado las siguientes herramientas:

- **Planificación proyecto**
  - Microsoft Project Professional 2010
- **Diseño**
  - Gimp 2.8
  - Inkscape 0.48
  - TexturePacker 3.2.0
  - Microsoft Visio Professional 2010
- **Programación**
  - Entorno de desarrollo Eclipse 4.2.1 Juno
  - Android develompment tools ADT
  - Android SDK

- Framework libGDX 0.9.8
- ObjectAid, herramienta de visualización de código en UML para Eclipse
- Control de versiones Git, utilizando como plataforma GitHub

- **Edición de audio**

  - Audacity 2.0.3
  - GoldWave 5.69

- **Redacción de documentos**

  - OpenOffice 4.0.1

- **Edición de vídeo**

  - Adobe Premiere Pro CS5.5

1.6 Riesgos del proyecto

Se parte de una experiencia previa en desarrollo de videojuegos, ver apartado 9 otros proyectos realizados al final de este documento. Aunque no se dispone de experiencia con la librería escogida ni tampoco con programación sobre Android. A nivel técnico no se prevé que puedan surgir muchos riesgos en líneas generales. No obstante, existe la posibilidad de gastar más tiempo del previsto consultando documentación para lograr los objetivos establecidos.

Existe un riesgo más alto a nivel artístico, ya que no se dispone de ningún conocimiento de arte de ningún tipo. Por lo tanto, muy probablemente a nivel artístico el juego sea un poco justo. Es por este motivo que durante la realización de este proyecto no se va a profundizar en temas de diseño artístico, en su lugar se utilizaran recursos de arte existentes en la red con licencias libres tipo creative commons o dominio público. Al final de este documento, en el apartado 6 se puede ver un listado de los recursos utilizados, juntamente con un link a la obra y el nombre del autor si este esta disponible. Se hará una selección de distintos tipos de recursos gráficos para posteriormente hacer el montaje de toda la parte gráfica del videojuego.

En cuanto a la parte de audio la situación es exactamente la misma, no se dispone de ningún conocimiento para componer música ni efectos. Por lo tanto, se buscarán sonidos y temas musicales con licencias libres. Intentando que estos queden integrados de la mejor manera posible al videojuego.

Finalmente, existe el riesgo de que el gameplay definido en el diseño conceptual del videojuego, una vez puesto en práctica, no sea del todo funcional o necesite de un balanceo para conseguir un juego equilibrado y jugable.

2. Análisis funcional

Basándonos en el aparatado “1.3.1 Definición del videojuego”, se han definido los requisitos funcionales necesarios para este, con la definición de casos de uso. Posteriormente se ha definido un diagrama de clases inicial, el cual ha servido como punto de partida para el desarrollo completo del videojuego. A su vez, se definen las máquinas de estado finitas utilizadas para el desarrollo.

2.1 Requisitos funcionales, casos de uso

En este apartado, se presentan tres diagramas de casos de uso El primer diagrama engloba la vista generalizada del actor “jugador” y los casos de uso que describen los requisitos funcionales del sistema. En los dos diagramas restantes, se describen la vista generalizada del actor “IA”, por IA se entiende la inteligencia artificial del videojuego. Los diagramas muestran los requisitos funcionales de la IA para el personaje principal y los enemigos. La IA interactúa con el sistema, y por tanto, será susceptible de analizar su "uso" de la aplicación:

2.1.1 Diagrama casos de uso actor “Jugador”

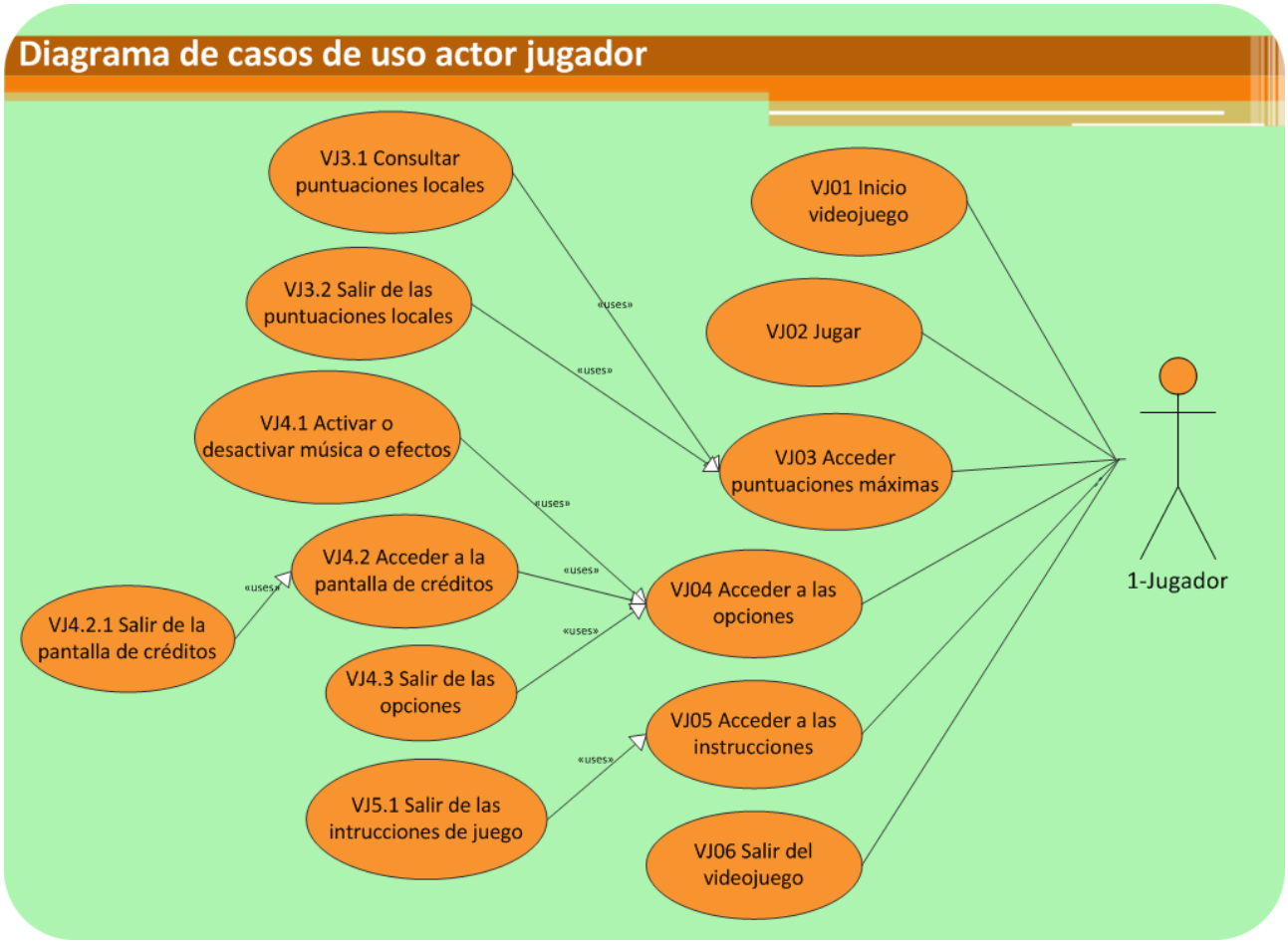


ilustración 3. Diagrama de casos de usos actor jugador

Como se puede observar en la ilustración 3, el sistema solo dispone de un único actor. Que en este caso es el jugador del videojuego, es decir representa todos los posibles usuarios que pueden jugar al videojuego.

En la siguiente tabla se hace un resumen de los casos de uso mostrados en la ilustración 3, estos serán desarrollados en la sección posterior.

2.1.2 Tabla resumen casos de uso

Código	Descripción	Actor
VJ01	Inicio videojuego	Jugador
VJ02	Jugar	Jugador
VJ03	Acceder puntuaciones máximas	Jugador
VJ3.1	Consultar puntuacione locales	Jugador
VJ3.2	Salir puntuacione locales	Jugador
VJ04	Acceder a las opciones	Jugador
VJ4.1	Activar o desactivar música o efectos	Jugador
VJ4.2	Acceder a la pantalla de créditos	Jugador
VJ4.2.1	Salir de la pantalla de créditos	Jugador
VJ4.3	Salir de las opciones	Jugador
VJ05	Acceder a las instrucciones	Jugador
VJ5.1	Salir de las intrucciones	Jugador
VJ06	Salir del videojuego	Jugador

2.1.3 Descripción textual de los casos de uso

2.1.3.1 VJ01 Inicio videojuego

Identificador	VJ01
Nombre	Inicio videojuego
Autor	Xavier Figuera
Resumen	El usuario inicia el videojuego
Actor(es)	Jugador
Precondiciones	Ninguna, el videojuego no había sido iniciado aún
Postcondiciones	El usuario a iniciado el videojuego, a visualizado la pantalla de intro durante un segundo y se encuentra en la pantalla del menú principal.
Flujo normal	1. El usuario accede a la lista de aplicaciones de su dispositivo 2. Busca el acceso al videojuego 3. Ejecuta el programa 4. Se carga la aplicación 5. El videojuego muestra la pantalla de intro

	6. Se muestra el menú principal al jugador
Flujos alternativos	Se podría dar el caso que el usuario accediera al videojuego desde algun acceso directo en alguno de los escritorios del dispositivo.
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.3.2 VJ02 Jugar

Identificador	VJ02
Nombre	Jugar
Autor	Xavier Figuera
Resumen	Jugar al videojuego
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y se encuentra en el menú principal
Postcondiciones	El usuario empieza a jugar
Flujo normal	1. El usuario pulsa el botón play del menú principal 2. El usuario juega 3. El usuario puede volver a jugar una vez finalizada la partida 4. El usuario puede salir del juego en cualquier momento
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.3.3 VJ03 Acceder puntuaciones máximas locales

Identificador	VJ03
Nombre	Acceder puntuaciones máximas locales
Autor	Xavier Figuera
Resumen	Acceder a la pantalla de puntuaciones máximas locales de las partidas realizadas en el dispositivo
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y se encuentra en el menú principal
Postcondiciones	El usuario se dirige a la pantalla con la tabla de las 10 puntuaciones máximas
Flujo normal	1. El usuario pulsa el botón Scores del menú principal 2. Al usuario le aparece la pantalla de puntuación

Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	VJ3.1 Consultar puntuaciones locales VJ3.2 Salir de la puntuaciones locales

2.1.3.4 VJ3.1 Consultar puntuaciones locales

Identificador	VJ3.1
Nombre	Consultar puntuaciones locales
Autor	Xavier Figuera
Resumen	Consulta de las puntuaciones máximas locales de las partidas realizadas en el dispositivo
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y ha pulsado el botón Scores del menú principal
Postcondiciones	El usuario visualiza la tabla de las 10 puntuaciones máximas
Flujo normal	1. El usuario visualiza una tabla con las 10 puntuaciones máximas
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.3.5 VJ3.2 Salir puntuaciones locales

Identificador	VJ3.2
Nombre	Salir puntuaciones locales
Autor	Xavier Figuera
Resumen	Salir de las puntuaciones máximas locales.
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y ha pulso el botón Scores.
Postcondiciones	El usuario sale de la tabla de puntuaciones
Flujo normal	1. El usuario pulsa el botón salir en la pantalla tabla de puntuaciones.
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.3.6 VJ04 Acceder a las opciones

Identificador	VJ04
Nombre	Acceder a la pantalla de opciones
Autor	Xavier Figuera
Resumen	Acceder a la pantalla de opciones del videojuego
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y se encuentra en el menú principal
Postcondiciones	El usuario se dirige a la pantalla de opciones
Flujo normal	1. El usuario pulsa el botón Options del menú principal 2. Al usuario le aparece la pantalla de opciones
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	VJ4.1 Activar o desactivar música o efectos VJ4.2 Acceder a la pantalla de créditos VJ4.3 Salir de las opciones

2.1.3.7 VJ4.1 Activar o desactivar música o efectos

Identificador	VJ04.1
Nombre	Activar o desactivar música o efectos
Autor	Xavier Figuera
Resumen	Activar o desactivar música o efectos de audio del videojuego
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y ha pulsado el botón Options en el menú principal
Postcondiciones	El usuario activa o desactiva la música o efectos del videojuego
Flujo normal	1. El usuario pulsa el botón altavoz para activar o desactivar la música del videojuego 2. El usuario pulsa el botón notas musicales para activar o desactivar los efectos de audio del videojuego
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.3.8 VJ4.2 Acceder a la pantalla de créditos

Identificador	VJ4.2
Nombre	Acceder a la pantalla de créditos
Autor	Xavier Figuera
Resumen	Acceder a la pantalla de créditos
Actor(es)	Jugador
Precondiciones	El usuario se encuentra en la pantalla de opciones
Postcondiciones	El usuario accede a la pantalla de créditos
Flujo normal	1. El usuario pulsa el botón credits para acceder a los créditos 2. El usuario puede desplazar el texto de los créditos deslizando el dedo sobre el texto hacia arriba y hacia abajo
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	VJ4.2.1 Salir de la pantalla de créditos

2.1.3.9 VJ4.2.1 Salir de la pantalla de créditos

Identificador	VJ4.2.1
Nombre	Salir de la pantalla de creditos
Autor	Xavier Figuera
Resumen	Salir de la pantalla de créditos
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y ha pulsado el botón Options en el menú principal y ha accedido a la pantalla créditos
Postcondiciones	El usuario accede a la pantalla de créditos
Flujo normal	1. El usuario pulsa el botón salir para abandonar la pantalla para acceder a los créditos 2. El usuario se encuentra de nuevo en la pantalla opciones
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.3.10 VJ05 Acceder a las instrucciones

Identificador	VJ05
Nombre	Acceder a la pantalla de instrucciones
Autor	Xavier Figuera
Resumen	Acceder a la pantalla de instrucciones
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y ha pulsado el botón ? del menú principal
Postcondiciones	El usuario accede a la pantalla de instrucciones
Flujo normal	1. El usuario pulsa el botón ? para acceder a las instrucciones 2. El usuario puede desplazar el texto de las instrucciones deslizando el dedo sobre el hacia arriba y hacia abajo
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	VJ5.1 Salir de las instrucciones

2.1.3.11 VJ5.1 Salir de las instrucciones

Identificador	VJ5.1
Nombre	Salir de las intrucciones
Autor	Xavier Figuera
Resumen	Salir de la pantalla de instrucciones
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego y ha pulsado el botón ? en el menú principal y ha accedido a la pantalla de instrucciones
Postcondiciones	El usuario accede a la pantalla de instrucciones
Flujo normal	1. El usuario pulsa el botón ? para acceder a la pantalla de instrucciones
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.3.12 VJ06 Salir del videojuego

Identificador	VJ06
Nombre	Salir del videojuego
Autor	Xavier Figuera
Resumen	Salir del videojuego
Actor(es)	Jugador
Precondiciones	El usuario ha iniciado el videojuego
Postcondiciones	El usuario abandona el videojuego
Flujo normal	1. El usuario pulsa el botón exit de la aplicación para salir del videojuego o el botón salir del dispositivo
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.4 Diagrama casos de uso actor "IA"

Por IA se entiende inteligencia artificial, es un elemento fundamental para dotar de realismo a un videojuego. Uno de los retos principales que se plantean a la hora de integrar comportamientos inteligentes es alcanzar un equilibrio entre la sensación de inteligencia y el tiempo de cómputo empleado por este subsistema. Este equilibrio es de gran importancia en el caso de los videojuegos. Este planteamiento gira, generalmente, en torno a la generación de soluciones que, sin ser óptimas, proporcionen una cierta sensación de inteligencia. En concreto, dichas soluciones deberían tener como meta que el jugador se enfrente a un reto que sea factible de manera que suponga un estímulo emocional y que consiga engancharlo al videojuego.

En estos últimos años, la IA ha pasado de ser un componente secundario en el proceso de desarrollo de videojuegos a convertirse en uno de los aspectos más importantes. Actualmente, lograr un alto nivel de IA en un juego sigue siendo uno de los retos más emocionantes y complejos y, en ocasiones, sirve para diferenciar un juego normal de uno realmente espectacular.

Dentro de este proyecto de ha definido una IA básica para dotar del realismo necesario al videojuego propuesto. En los siguientes diagramas, se define esta IA en forma de casos de usos para su posterior implementación.

2.1.4.1 Diagrama casos de uso IA personaje principal (oveja)

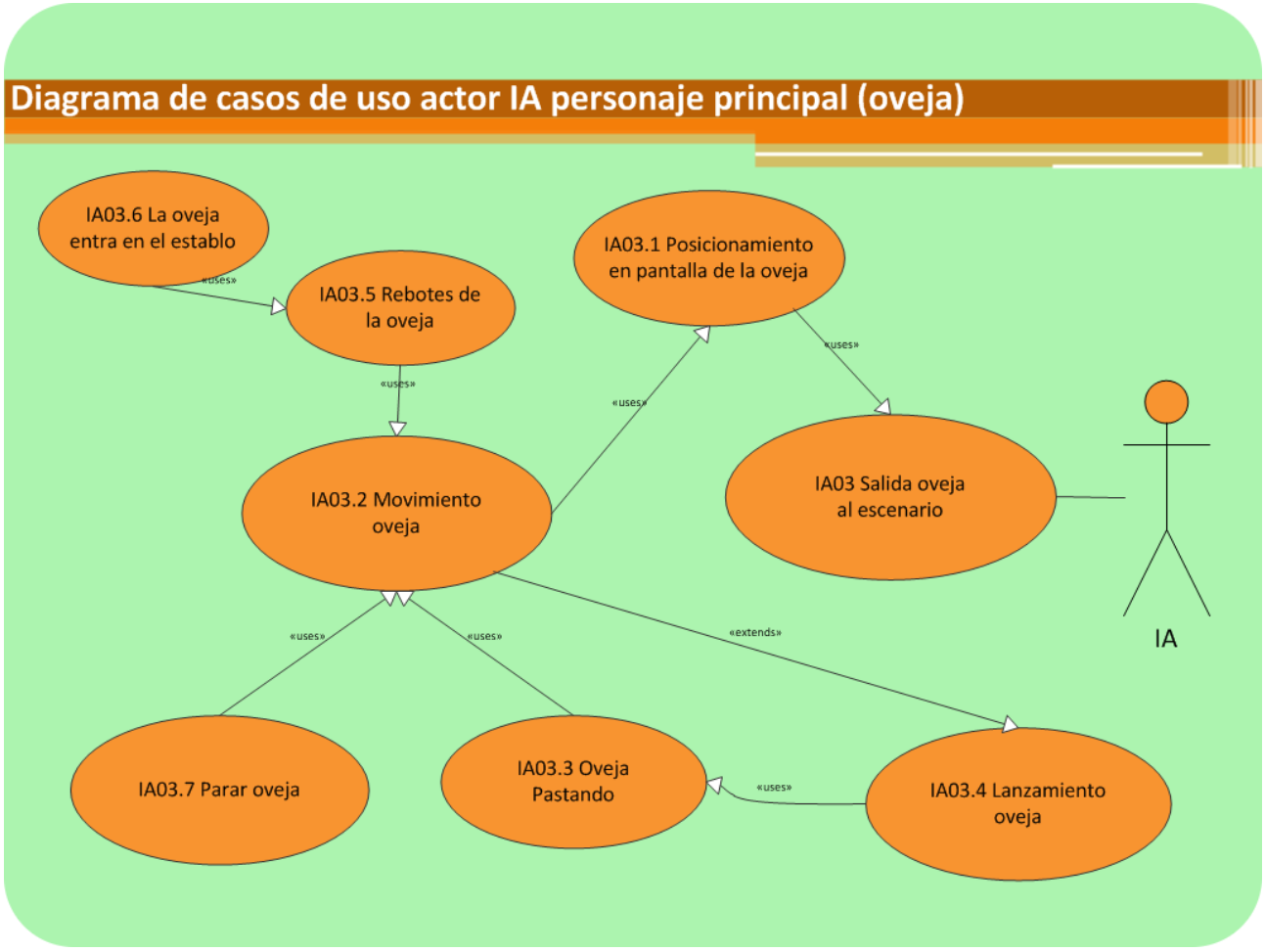


ilustración 4. Diagrama de casos de usos personaje principal

En la siguiente tabla se hace un resumen de los casos de uso mostrados en la ilustración 4, estos serán desarrollados en la sección posterior.

2.1.4.1.1 Tabla resumen casos de uso

Código	Descripción	Actor
IA03	Salida oveja al escenario	IA
IA03.1	Posicionamiento en pantalla de la oveja	IA
IA03.2	Movimiento oveja	IA
IA03.3	Oveja pastando	IA
IA03.4	Lanzamiento oveja	IA
IA03.5	Rebotes de la oveja	IA
IA03.6	La oveja entra en el establo	IA
IA03.7	Parar oveja	IA

2.1.4.1.2 Descripción textual de los casos de uso

2.1.4.1.2.1 IA03 Salida ovejas al escenario

Identificador	IA03
Nombre	Salida oveja al escenario
Autor	Xavier Figuera
Resumen	Salida oveja al escenario
Actor(es)	IA
Precondiciones	El usuario ha iniciado una partida
Postcondiciones	La oveja es instanciada de forma automática sin intervención del usuario
Flujo normal	1. El usuario ha iniciado una partida 2. Las ovejas se empiezan a instanciar sin intervención del usuario 3. El punto 2 se repite en un intervalo de tiempo determinado indefinidamente
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	IA03.1 Posicionamiento en pantalla de las oveja

2.1.4.1.2.2 IA03.1 Posicionamiento en pantalla de las oveja

Identificador	IA03.1
Nombre	Posicionamiento en pantalla de las oveja

Autor	Xavier Figuera
Resumen	Asignación inicial posición de las ovejas en pantalla, siempre es la misma, posición, esta queda fuera del rango visual de la pantalla
Actor(es)	IA
Precondiciones	La partida esta iniciada y la oveja instanciada
Postcondiciones	La oveja queda posicionada en unas coordenadas fuera de la pantalla
Flujo normal	1. Se instancia la oveja. 2. La oveja queda posicionada en unas coordenadas únicas fuera del rango visual de pantalla
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	IA03.2 Movimiento oveja

2.1.4.1.2.3 IA03.2 Movimiento oveja

Identificador	IA03.2
Nombre	Movimiento oveja
Autor	Xavier Figuera
Resumen	Movimiento de las ovejas hacia el escenario
Actor(es)	IA
Precondiciones	La partida esta iniciada, la oveja instanciada y posicionada
Postcondiciones	La oveja se mueve hacia el escenario, empieza a pastar.
Flujo normal	1. La oveja se mueve hacia el escenario.
Flujos alternativos	Ninguno
Inclusiones	IA03.4
Extensiones	IA03.3 Oveja Pastando IA03.7 Parar oveja

2.1.4.1.2.4 IA03.3 Oveja pastando

Identificador	IA03.3
Nombre	Oveja pastando
Autor	Xavier Figuera
Resumen	La ovejas permanecen en un espacio determinado de coordenadas de pantalla mientras no sean lanzadas al escenario por el usuario
Actor(es)	IA

Precondiciones	La oveja ha salido al escenario
Postcondiciones	La oveja se mueve dentro de un espacio determinado de forma indefinida
Flujo normal	1. La oveja se mueve de forma indefinida hasta que sea lanzada por el usuario
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	IA03.4 Lanzamiento oveja

2.1.4.1.2.5 IA03.4 Lanzamiento oveja

Identificador	IA03.4
Nombre	Lanzamiento oveja
Autor	Xavier Figuera
Resumen	La oveja es lanzada por el usuario y se acelera en función de la fuerza de lanzamiento
Actor(es)	IA
Precondiciones	La oveja se mueve
Postcondiciones	La oveja ha sido lanzada
Flujo normal	1. La oveja se mueve de forma indefinida, 2. La oveja es lanzada 3. La oveja acelera en función de la fuerza de lanzamiento
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	IA03.2 Movimiento oveja

2.1.4.1.2.6 IA03.5 Rebotes de la oveja

Identificador	IA03.5
Nombre	Rebotes de la oveja
Autor	Xavier Figuera
Resumen	La oveja ha sido lanzada y rebota por el escenario
Actor(es)	IA
Precondiciones	La oveja ha sido lanzada
Postcondiciones	La oveja rebota por el escenario
Flujo normal	1. La oveja ha sido lanzada y rebota por el escenario 2. La oveja entra en el establo si colisiona con la puerta y esta abierta

Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	IA03.6 La oveja entra en el establo

2.1.4.1.2.7 IA03.6 La oveja entra en el establo

Identificador	IA03.6
Nombre	La oveja entra en el establo
Autor	Xavier Figuera
Resumen	La oveja entra en el establo
Actor(es)	IA
Precondiciones	La oveja esta rebotando por el escenario y la puerta del establo esta abierta
Postcondiciones	La oveja entra en el establo
Flujo normal	1. La oveja entra en el establo si colisiona con la puerta y esta abierta
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.4.1.2.8 IA03.7 Parar oveja

Identificador	IA03.7
Nombre	Parar oveja
Autor	Xavier Figuera
Resumen	La oveja para si se hacen dos pulsaciones encima de ella
Actor(es)	IA
Precondiciones	La oveja esta rebotando por el escenario
Postcondiciones	La oveja se para
Flujo normal	1. La oveja esta rebotando 2. El usuario pulsa dos veces encima de ella 3. La oveja se para
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.4.2 Diagrama casos de uso IA enemigos (lobo)

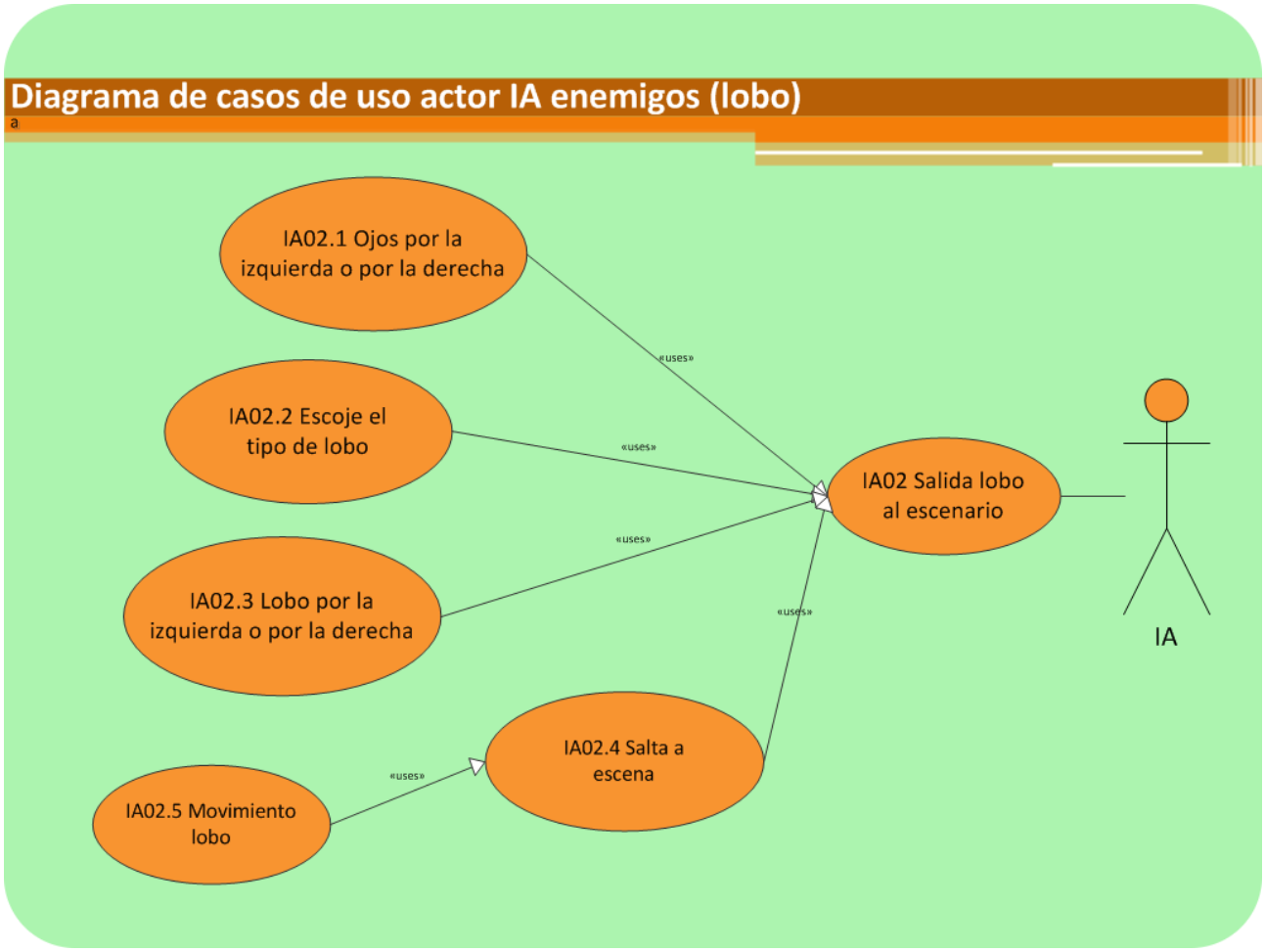


ilustración 5. Diagrama de casos de usos enemigos

2.1.4.2.1 Tabla resumen casos de uso

Código	Descripción	Actor
IA02	Salida lobo al escenario	IA
IA02.1	Ojos por la derecha o por la izquierda	IA
IA02.2	Escoje el tipo de lobo	IA
IA02.3	Lobo por la derecha o por la izquierda	IA
IA02.4	Salta a escena	IA
IA02.5	Movimento lobo	IA

2.1.4.2.2 Descripción textual de los casos de uso

2.1.4.2.2.1 IA02 Salida lobo al escenario

Identificador	IA02
Nombre	Salida lobo al escenario
Autor	Xavier Figuera
Resumen	Salida lobo al escenario
Actor(es)	IA
Precondiciones	El usuario ha iniciado una partida
Postcondiciones	El lobo es instanciado de forma automática sin intervención del usuario
Flujo normal	1. El usuario ha iniciado una partida 2. Los lobos se empiezan a instanciar sin intervención del usuario 3. El punto 2 se repite en un intervalo de tiempo determinado indefinidamente
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	IA02, IA02.1, IA02.2, IA02.3, IA02.4

2.1.4.2.2.2 IA02.1 Ojos por la derecha o por la izquierda

Identificador	IA02.1
Nombre	Ojos por la derecha o por la izquierda
Autor	Xavier Figuera
Resumen	Salida de ojos al escenario paso previo a que salte el lobo
Actor(es)	IA
Precondiciones	El lobo ha sido instanciado
Postcondiciones	Aparecen unos ojos que se cierra y se abren por la derecha o izquierda de la pantalla de forma aleatoria
Flujo normal	1. El lobo ha sido instanciado 2. Aparecen unos ojos que se cierran y se abren 3. El punto 2 se repite en un intervalo de tiempo determinado indefinidamente
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.4.2.2.3 IA02.2 Escoge el tipo de lobo

Identificador	IA02.2
Nombre	Escoge el tipo de lobo
Autor	Xavier Figuera
Resumen	Escoge el tipo de lobo de forma aleatoria de los tres definidos
Actor(es)	IA
Precondiciones	Han aparecido los ojos
Postcondiciones	Ha escogido el tipo de lobo que saltara al escenario
Flujo normal	1. Han aparecido los ojos 2. Ha escogido el tipo de lobo que saltara al escenario
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.4.2.2.4 IA02.3 Lobo por la derecha o por la izquierda

Identificador	IA02.3
Nombre	Lobo por la derecha o por la izquierda
Autor	Xavier Figuera
Resumen	Escoge de forma aleatoria el lado por donde debe aparecer el lobo
Actor(es)	IA
Precondiciones	Se ha seleccionado el tipo de lobo
Postcondiciones	Se ha escogido el lado por donde aparecerá
Flujo normal	1. Se ha escogido el lado por donde aparecerá
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.1.4.2.2.5 IA02.4 Salta a escena

Identificador	IA02.4
Nombre	Salta a escena
Autor	Xavier Figuera
Resumen	El lobo salta a escena
Actor(es)	IA
Precondiciones	Se ha seleccionado el lado por donde debe aparecer y salta
Postcondiciones	El lobo esta en escena
Flujo normal	1. Se ha seleccionado el lado por donde debe aparecer y salta 2. El lobo esta en escena
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	A02.5

2.1.4.2.2.6 IA02.5 Movimiento lobo

Identificador	IA02.5
Nombre	Movimiento lobo
Autor	Xavier Figuera
Resumen	El lobo esta en escena y se mueve
Actor(es)	IA
Precondiciones	El lobo ha saltado a la escena
Postcondiciones	El lobo se mueve
Flujo normal	1. El lobo ha saltado a la escena 2. El lobo se mueve
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	Ninguna

2.2 Diseño inicial diagrama de clases

2.2.1 Diagrama de clases inicial "Fling the sheep"

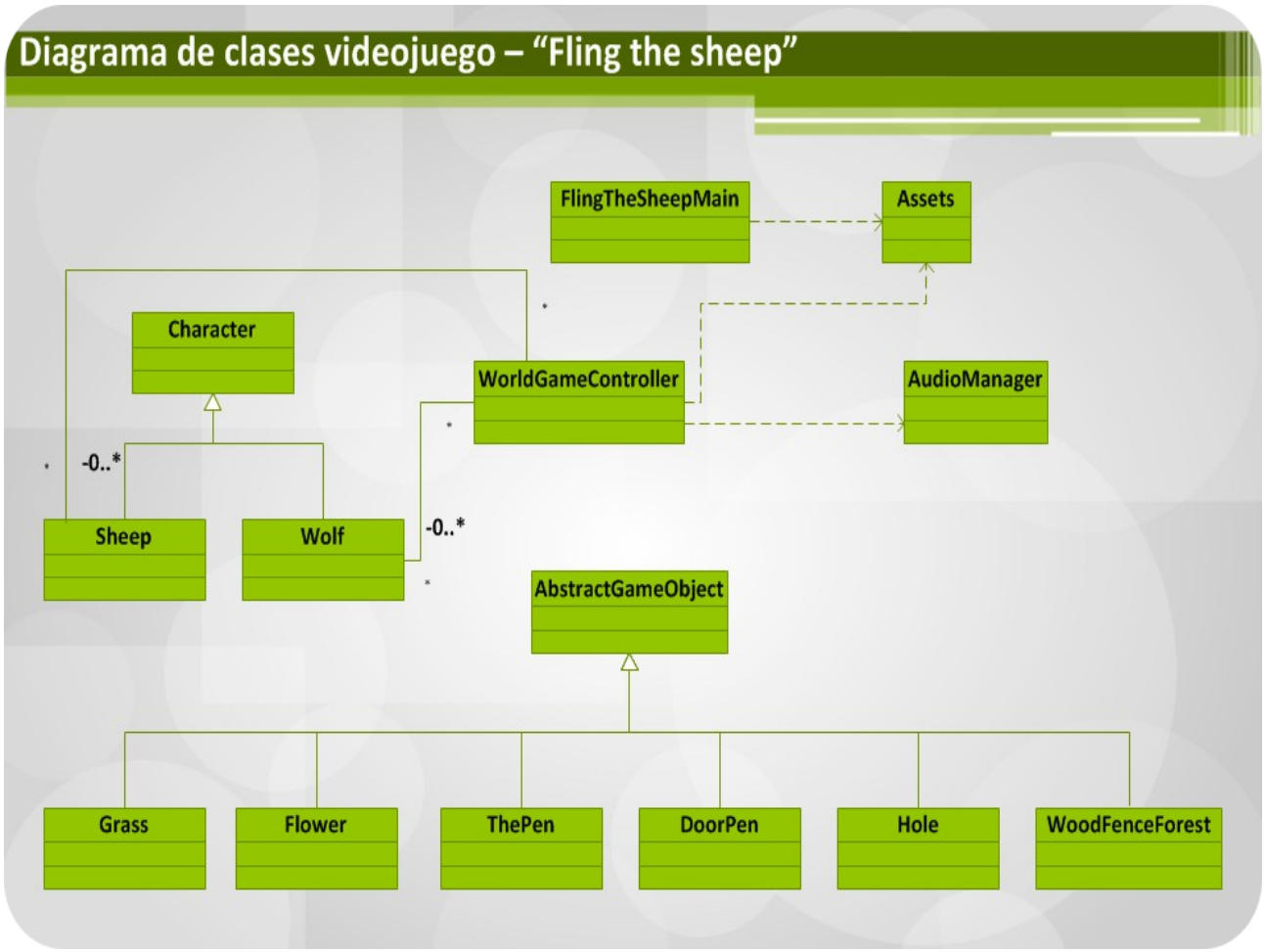


ilustración 6. Diagrama de clases videojuego

En este apartado, se define la estructura inicial de la aplicación a través del diagrama de clases mostrado en la ilustración 6. Esto servirá como punto de partida para el completo diseño de la aplicación para su posterior implementación. A continuación se pasan a detallar a modo resumido que funcionalidad tiene cada una de las clases definidas en este diagrama.

2.2.1.1 Resumen clases videojuego

FlingTheSheepMain: Clase principal desde donde se inicializará la aplicación.

Assets: Esta clase contendrá todos los recursos necesarios del juego, texturas, sonidos, etc. Estos serán inicializados en el momento de iniciar el juego. Existe una dependencia entre la clase FlingTheSheepMain y Assets, ya que la instanciación de Assets esta condicionada por la instanciación de la clase principal FlingTheSheepMain cuando se pone en marcha el videojuego.

WorldGameController: Esta clase contendrá la lógica del videojuego.

AudioManager: Clase destinada para controlar la reproducción de efectos de sonido y audio del videojuego.

Charcater: Es la superclase que describe los comportamientos comunes de los personajes, de ella se extienden los personajes principales y los enemigos.

Sheep: Clase que representa el personaje principal.

Wolf: Clase que representa a los enemigos

AbstractGameObject: Esta superclase recoge todos los comportamientos comunes de los objetos del juego, por objetos se entiende todos los componentes que forman el escenario del videojuego. De esta clase se extienden todos los objetos del videojuego.

Grass: Hierba que aparecerá en el escenario.

Flower: Flores que aparecerán en el escenario.

ThePen: Establo donde hay que hacer llegar las ovejas

DoorPen: Puerta del establo, esta se abrirá y se cerrara aleatoriamente

Hole: Agujeros en el escenario

WoodFenceForest: Valla de madera que aparecerá en el escenario.

Como se ha comentado, este diagrama de clases es un diagrama inicial, por tanto lo más probable es que durante el desarrollo del videojuego se añadan más clases al diseño inicial mostrado en la ilustración 6. En el apartado 5.5 de este documento se muestran los diagramas de clases definitivos.

2.2.2 Diagrama de clases inicial de pantallas

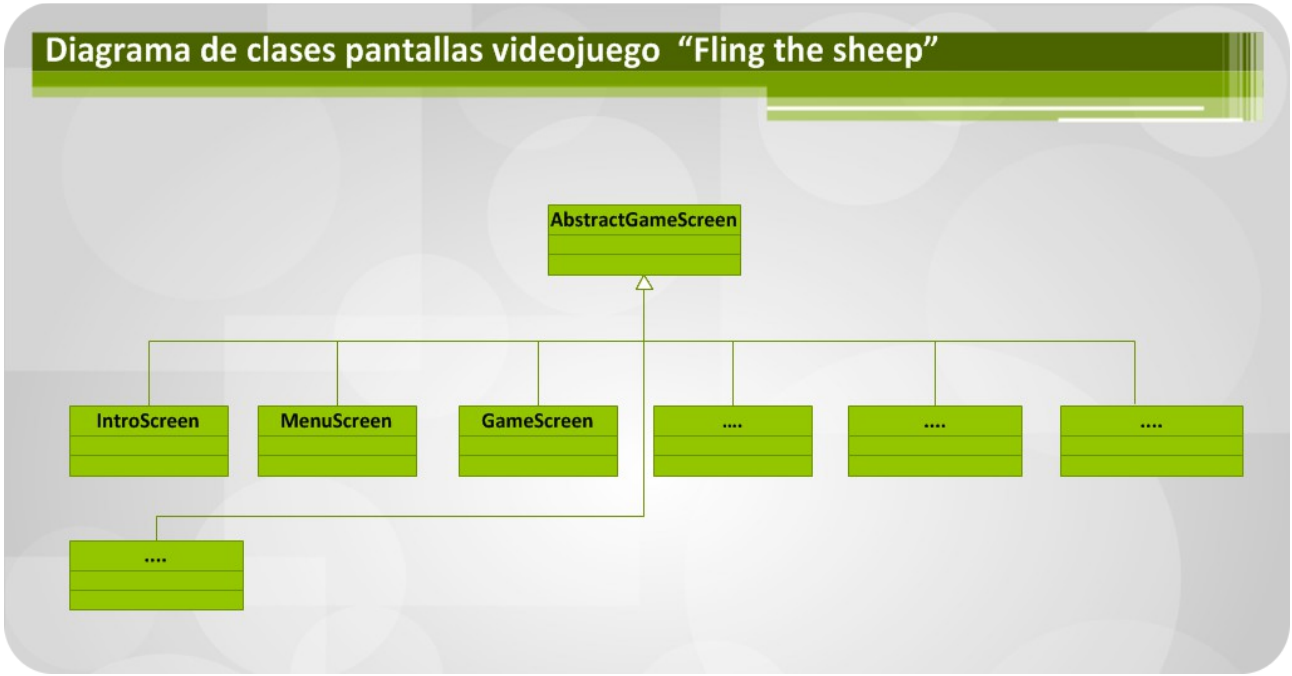


ilustración 7. Diagrama de clases de pantallas

Este diagrama recoge en forma de clases, las pantallas iniciales definidas del videojuego. Se parte de una superclase llamada AbstractGameScreen que recogerá las operaciones comunes de todas ellas y de esta se extienden las pantallas concretas, este diagrama solo muestra las pantallas más básicas definidas inicialmente, estas servirán como base para desarrollar el conjunto de pantallas total necesarias para este proyecto y que quedan recogidas en su totalidad en el apartado 4 "Diseño" de esta documento.

A continuación se pasan a detallar a modo resumido cada una de ellas.

2.2.2.1 Resumen clases pantallas

IntroScreen: Clase para la pantalla de intro.

MenuScreen: Clase que contiene el menú principal del videojuego.

GameScreen: La clase game screen es la que contiene la pantalla donde se desarrolla todo el videojuego.

2.3 Definición de máquinas de estado finitas

Como ya se ha introducido en el punto 2.1.2 la IA aplicada a los videojuegos, dota a estos de realismo. Existen distintas técnicas para la implementación de esta. Una de las más consolidadas es, el uso de máquinas de estados finitas, ya que tienen un coste computacional casi inexistente ya que no tienen ningún coste de procesamiento asignado. Las máquinas de estados permiten de forma fácil definir comportamientos.

Es por este motivo, que han sido utilizadas no solamente para definir los comportamientos de la IA del videojuego, sino también para definir los distintos estados en que se puede encontrar el videojuego durante su tiempo de vida. A continuación se detallan los distintos diagramas de las máquinas de estados utilizadas para este proyecto conjuntamente con una explicación de cada una de ellas.

2.3.1 Máquina de estados del videojuego

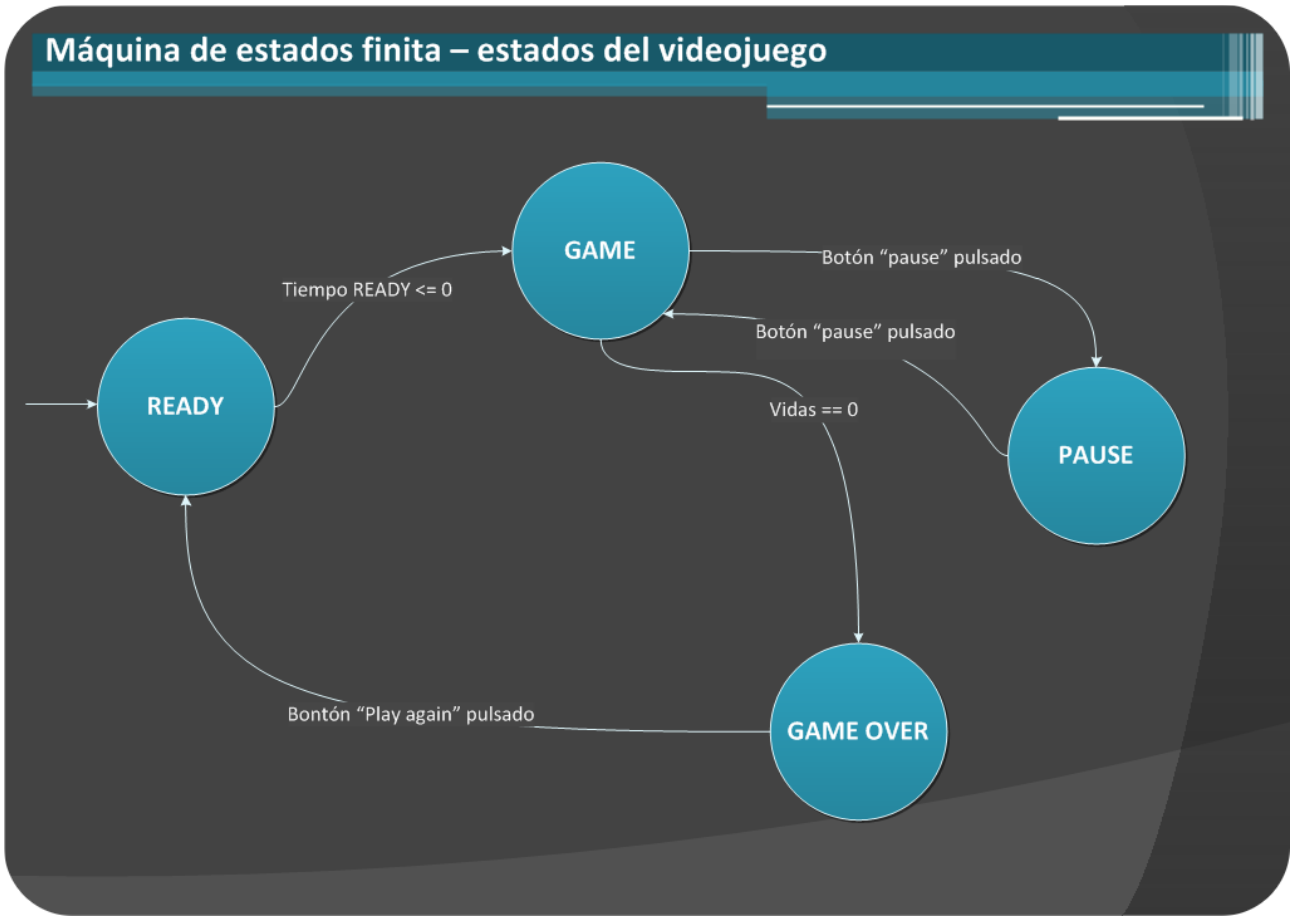


ilustración 8. Máquina de estados del videojuego

La ilustración 8, nos muestra los estados que puede tomar el videojuego durante su tiempo de vida.

Este diagrama define los estados que puede tomar el videojuego una vez iniciado, inicialmente el juego se encuentra en el estado READY, de este estado existe una transición al estado GAME cuando el tiempo de READY finaliza READY <=0. El estado game sera el estado en donde se

desarrollara todo el juego. Des de este estado podemos pasar al estado PAUSE si pulsamos el botón de pausa del juego y desde el estado PUASE volvemos al estado GAME si volvemos a pulsar el botón pause. Finalmente desde el estado GAME haremos la transición al estado GAME OVER si nos arrebatan todas las vidas durante la partida. Desde el estado GAME OVER podremos volver a READY si decidimos hacer una nueva partida.

2.3.2 Máquina de estados comportamiento oveja

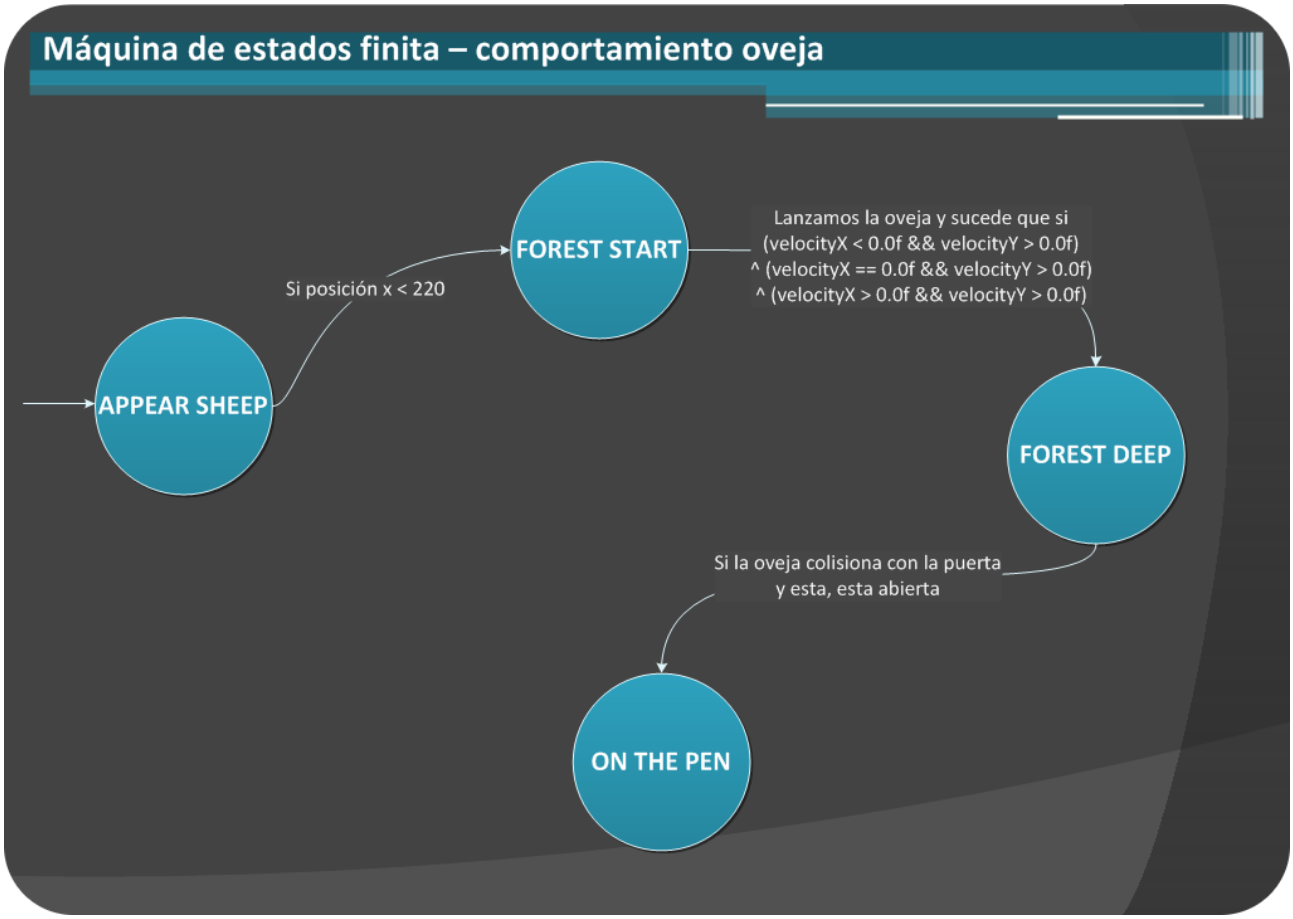


ilustración 9. Máquina de estados comportamiento oveja

La ilustración 9 muestra el comportamiento del personaje principal del videojuego que en este caso es la oveja. Se define el ciclo de vida de la oveja a través de sus comportamientos.

APPEAR SHEEP: La oveja aparece fuera del rango visual de la pantalla y empieza a moverse, para que suceda la transición hacia el siguiente estado FOREST START, la coordenada x debe ser menor que 220.

FOREST START: La oveja seguirá moviéndose en un espacio determinado de la pantalla hasta que sea lanzada en unas determinadas direcciones, la dirección queda determinada, por las variables velocityX, velocityY, estas tomaran valores positivos o negativos determinando la dirección de lanzamiento, esto queda reflejado en la condición de transición del estado FOREST START a FOREST DEEP. Si realizamos el lanzamiento en esas determinadas direcciones haremos la transición al un nuevo estado FOREST DEEP.

FOREST DEEP: La oveja rebota por el escenario, si colisiona con la puerta del corral y esta abierta hacemos la transición al estado ON THE PEN.

ON THE PEN: Finaliza el ciclo de vida de la oveja dentro del juego, esta es devuelta al corral, se incrementan puntos en el marcador de puntuación, y se incrementa una unidad en el contador de ovejas salvadas.

2.3.3 Máquina de estados comportamiento lobo

ilustración 10. Máquina de estados comportamiento lobo

La ilustración 10 muestra el comportamiento de los enemigos del videojuego que en este caso son los lobos. Se define el ciclo de vida del lobo a través de sus comportamientos.

APPEAR EYES: Antes de aparecer los lobos a escena aparecerán sus ojos entre los matorrales del bosque. Cuando acaba el tiempo pasa al siguiente estado tiempo ojos <= 0

APPEAR WOLF: En este estado se seleccionara el tipo de lobo (habrá 3 tipos distintos), el lado de la pantalla por donde aparecerá posición y velocidad de salto como también el tiempo en segundos que tardara a saltar al escenario. Una vez seleccionados todos estos parámetros hará la transición al estado WHEN JUMP.

WHEN JUMP: Este estado simplemente esperara que pase el tiempo para que el lobo salte al escenario. Un vez pase el tiempo el lobo aparecerá en el escenario pasando a la transición JUMP ON SCENE.

JUMP ON SCENE: En este estado se efectuara el salto a escena del lobo, este saltara desde los

matorrales laterales de la pantalla,hacia el escenario dibujando un arco convexo. Para realizar el salto nos apoyaremos en la representación del calculo de la función trigonométrica coseno, solamente en sus valores positivos, por tanto cuando cosy < 0 haremos la transición al estado MOVE ya que el salto habrá finalizado.

MOVE: En este estado los lobos se moverán y podrán ser ahuyentados pulsando repetidas veces encima de ellos, las veces a pulsar queda sujeta al tipo de lobo que sea. Des de este estado, podremos hacer a dos estado distintos dependiendo de lo que suceda. Si conseguimos ahuyentar al lobo pasaremos al estado WOLF SCARED. Si por el contrario el lobo atrapa a la oveja, haremos la transición al estado SHEEP TRAPPED, en ambos estados finalizara el ciclo de vida del lobo.

WOLF SCARED: Se habrá ahuyentado el lobo, se incrementa en uno el contador de lobos ahuyentados y se suman puntos a la puntuación total. En este estado finaliza el ciclo de vida del lobo.

SHEEP TRAPPED: El lobo habrá atrapado una oveja, se nos restara una vida. En este estado finaliza el ciclo de vida del lobo.

2.3.4 Máquina de estados tutorial in game

ilustración 11. Máquina de estados tutorial in game

La ilustración 11 muestra los distintos estados que toma el mini tutorial in game que aparecerá la primera vez que se juega. En este las transiciones entre estados suceden de forma automática

cada 5 segundos desde que se ha activado el primer estado. Cada estado corresponderá a una caja de texto mostrada en pantalla.

3. Diseño

3.1 Estudio comparativo

Un estudio realizado en 2011 por la empresa Newzoo: http://www.newzoo.com, empresa internacional dedicada a los estudios de mercado y enfocada por completo a la industria de los videojuegos. Ha realizado la primera encuesta a jugadores españoles y una de las cosas que dice es: **"La suma del volumen de negocio de juegos online (redes sociales, juegos casuales, juegos para móviles, juegos MMO) superará al del consumo en juegos para consolas en 2011. "**. Se puede ver un resumen del estudio en línea en:

http://www.eurogamer.es/articles/2011-10-26-espana-gastara-1-600-millones-de-euros-en-videojuegos-en-2011

Con esta información, se puede ver que hacer un videojuego casual para una plataforma móvil como Android puede ser muy viable.

Para este proyecto se realizará un estudio comparativo (benchmarking). En la industria de los videojuegos es habitual, sacar títulos fijándose en otros títulos que han conseguido gran éxito en el mercado. La operativa consiste en mejorarlos dentro de otro contexto o hacer un videojuego completamente diferente pero basándose en ocasiones en mecánicas de algún otro título que ha conseguido un gran éxito.

Existen en Google Play gran cantidad de juegos casuales, que pueden servir como inspiración y como punto de inicio para hacer un planteamiento general de cómo diseñar el producto que se quiere desarrollar partiendo de los requisitos definidos en el apartado 1.3.1 de este documento. Se tomarán como referencia algunos títulos para identificar patrones, para ver cómo otros han solucionado ciertas situaciones, como pueden ser problemas de diseño o interacción entre otros. En ningún caso, deberán ser exactamente de la misma temática pero si que se mirará que sean del mismo tipo, es decir casual games.

El primer paso para realizar este estudio ,a sido ir a Google play y verificar si hay algún juego con el mismo nombre que el que se pretende desarrollar “Fling the sheep”. No existe ningún juego a día de hoy con este nombre.

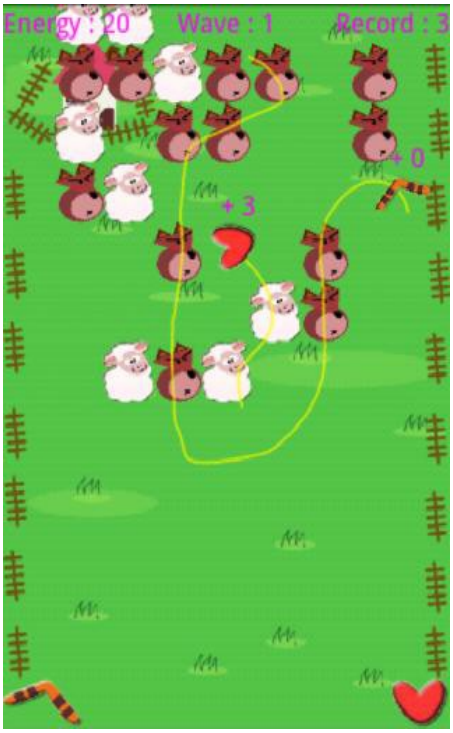
El segundo paso a sido buscar también en Google play videojuegos en los que aparezcan ovejas, solo poniendo en el buscador la palabra "sheep", nos aparecen muchísimos juegos que tienen una temática donde aparecen ovejas. Se ha verificado también poniendo combinaciones de palabras relacionadas con el juego que se quiere implementar, palabras como "sheep wolves" o " sheep farm", etc. los resultados obtenidos son bastante elevados.

Por la cantidad de juegos encontrados y los recursos de que se dispone, como por ejemplo el tiempo, no es posible analizarlos todos. Si que se tomaran algunos juegos después de hacer un repaso de los más destacados. Hay que decir, que después de haber instalado algunos, alguno de ellos no ha acabado de funcionar correctamente en el dispositivo móvil.

Tras estos pasos, se escogerán dos videojuegos de temática similar y dos videojuegos de temática diferente. Los videojuegos escogidos para realizar el análisis comparativo son:

3.1.1 Sheep & Wolf

A continuación se muestran unas pantallas del videojuego:



ilustraciones 12 y 13 pantallas juego Sheep & Wolf

Link: https://play.google.com/store/apps/details?id=jscompany.games.SheepAndWolf_enter3

El juego consta de una mecánica muy simple, disponemos de un boomerang y un corazón , la perspectiva del juego es elevada y vemos cómo van bajando ovejas y lobos de forma aleatoria, se trata de forma táctil dibujar la trayectoria del boomerang para eliminar los lobos sin tocar las ovejas y con el corazón dibujar una trayectoria de forma táctil, para tocar las ovejas y no los lobos, conforme vamos tocando los lobos y las ovejas con el correspondiente objeto iremos sumando puntos .

Tiene cierta similitud con el que se quiere implementar ya que salen también lobos,ovejas y también que lo que se pretende implementar funcionara de forma táctil para lanzar las ovejas al corral y ahuyentar a los lobos que amenazan a las ovejas.

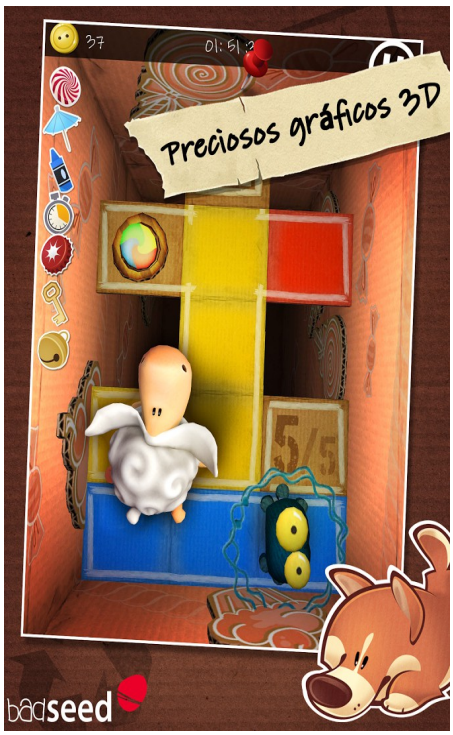
Tiene una perspectiva de cámara igual a la que se quiere implementar. Tiene una mecánica de juego muy sencilla, las opciones que ofrece son muy definidas y claras, 3 botones al inicio del juego grandes que nos indican las posibles opciones que tenemos dentro del juego: Game Start, Game Record, Exit . Un punto interesante de este juego, es que las puntuaciones no se guardan a nivel local de la aplicación sino que son compartidas en una pagina web, así se pueden ver las puntuaciones a nivel mundial. La navegación por las diferentes pantallas es correcta.

Quizá un aspecto negativo del juego sería que visualmente esta muy poco cuidado. Aunque a veces si el juego produce mucha adicción el aspecto visual puede quedar en segundo término en juegos para dispositivos móviles. En otros tipos de plataformas como consolas o PC el aspecto visual es un punto muy importante y que se debe trabajar mucho más a fondo, no obstante esto no significa que en un dispositivo móvil no deba estarlo.

Puntos Positivos: Sencillez , adictivo , puntuaciones a nivel mundial . Navegación correcta.
Puntos Negativos: Poco cuidado visualmente, no tiene pausa
Observaciones: No tiene un final definido a medida que avanzamos la dificultad aumenta.

3.1.2 Sheep Up !

A continuación se muestran unas pantallas del videojuego:



ilustraciones 14 y 15 pantallas juego Sheep Up !

Link:https://play.google.com/store/apps/details?id=com.pgpublish.android.sheepuplive

A diferencia del anterior este visualmente esta mucho más cuidado. Es un juego en 3D donde una oveja de juguete esta guardada dentro de una caja de cartón, Va botando y tiene que ir recogiendo botones e ir subiendo por diferentes niveles para poder salir de la caja, es un juego de plataformas también con una mecánica bastante sencilla.

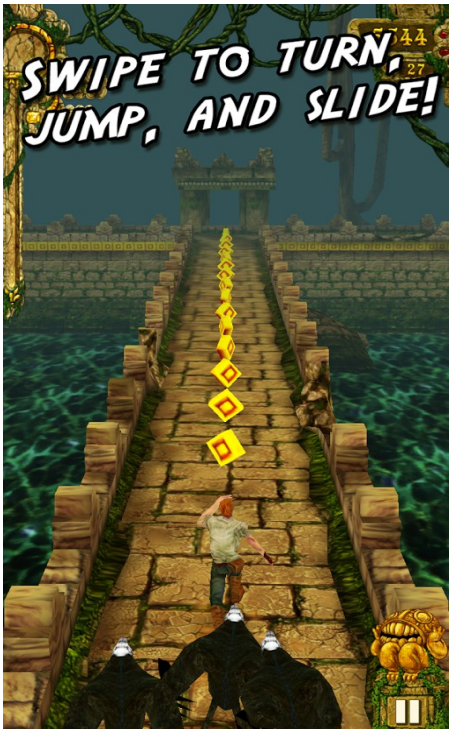
Este dispone de una pantalla que dura unos segundos y seguidamente sale el menú de opciones del juego, a diferencia de lo visto anteriormente , donde directamente salía el menú de opciones de juego. El menú de opciones de juego también es muy directo y claro. Aparte del botón play para iniciar el juego, el botón opciones para configurar el audio del juego, los efectos, el idioma y ver los créditos. Tiene el botón store donde este juego tiene implementada la modalidad freemium para ir comprando diferentes objetos dentro de el que nos darán ciertas ventajas dentro del gameplay. La compra se podrá realizar con monedas virtuales (los botones) que los habremos ido recogiendo durante los diferentes niveles jugados o también podemos comprar cantidades determinadas de botones a cambio de dinero de verdad. Dispone de vinculación a redes sociales como Twitter y Facebook.

Puntos Positivos: Sencillez, adictivo, vinculado a Twitter y el Facebook, el juego se puede parar, Navegación correcta.
Puntos Negativos: La pantalla inicial tarda demasiado en desaparecer y no da ningún tipo de

información al usuario, puede llegar a parecer que se a colgado debido al largo tiempo que tarda, al menos con el dispositivo que se ha verificado.

3.1.3 Temple Run

A continuación se muestran unas pantallas del videojuego:



ilustraciones 16 y 17 pantallas juego Temple run

Link:https://play.google.com/store/apps/details?id=com.imangi.templerun

Es un juego que no tiene final, donde un explorador tiene que evitar la muerte superando diferentes obstáculos, a través de diferentes pasillos, tendremos que dibujar diferentes trazos de forma táctil para que el personaje gire, salte o se deslice por el suelo, al mismo tiempo tendremos que inclinar el teléfono para que el personaje vaya recogiendo monedas que podremos utilizar para adquirir objetos y mejoras en la partida. A medida que vamos recorriendo más y más metros, la dificultad aumenta y nos obligara a realizar más acciones en un espacio de tiempo más corto así pues, los reflejos juegan un papel importante.

También dispone de una pantalla que dura unos segundos seguida de un menú de opciones muy claro y definido. Este igual que el anterior dispone de una tienda donde se pueden comprar mejoras (potenciadores) con dinero virtual o real. Los botones de inicio son muy similares al del anterior juego visto. Dispone de vinculación a redes sociales como Twitter.

Puntos Positivos: Sencillez, Adictivo, vinculado a Twitter, el juego se puede parar, Navegación correcta.
Puntos Negativos: -

3.1.4 Attack of the Spooklings

A continuación se muestran unas pantallas del videojuego:

ilustraciones 18 y 19 pantallas juego Attack of the Spooklings

Link:https://play.google.com/store/apps/details?id=com.picarogames.spooklings_free

Finalmente, nos centramos en este casual con aspecto visual retro. Otro juego que no dispone de final y donde la dificultad va aumentando conforme el tiempo de juego aumenta. El objetivo del juego es matar a todos los Spooklings que quieren atacar nuestro poblado. La forma de matar a los Spooklings es con el dedo deslizándolo sobre la pantalla, sólo que un Spookling llegue a nuestro poblado en la parte inferior de la pantalla, la partida finaliza. Debemos evitar de todas las formas posibles que lleguen a la parte inferior de la pantalla.

También dispone de una pantalla de inicio y un menú inicial de juego claro y directo, no dispone de ninguna vinculación a redes sociales. El juego no se puede parar. La navegación por todas las pantallas es correcta, teniendo pantalla de puntuaciones más altas, pantalla con mini tutorial de cómo jugar y unas opciones para configurar el audio y ver los créditos del juego. Sin embargo, a diferencia de los otros, tiene algunas fallas de diseño a nivel de transiciones entre pantallas y funcionalidades básicas que aparentemente no tienen mucho sentido. Fuera de la pantalla de juego no es posible abandonar la aplicación de ninguna de las maneras, no hay botón de salida y la tecla de ir hacia atrás no funciona, la única manera de salir es matando el proceso. Por otro lado, durante la partida con el botón de ir atrás conseguimos hacer pausa del juego, pero de una forma que no está anunciada al usuario de ninguna manera con algún icono visual que informa al usuario de la funcionalidad. Una vez parado el juego con una tecla que no está indicada en ninguna parte como tal, nos da dos opciones RESTART y QUIT, con QUIT salimos de la aplicación algo que no se puede hacer en ningún otro punto de la aplicación. Con RESTART vuelve a empezar la partida y se pierde la que teníamos empezada. En resumen si pulsamos este botón no está anunciado en ninguna parte por accidente durante una partida perdemos la partida ya que no tenemos la opción de continuar, puesto que o volvemos a empezar o salimos de la aplicación.

Puntos Positivos: Sencillez, Muy adictivo.
Puntos Negativos: Errores en el diseño de menús y transiciones entre pantallas, no tiene vinculación a redes sociales, no tiene pausa.

Todo lo que se ha visto en este estudio comparativo, nos da una idea de que el diseño que se lleve a cabo, debe cumplir las siguientes metas: tener una mecánica de juego adictiva y sencilla, que la disposición de opciones e iconos han ser claras y dejar claro al usuario de forma visual qué funcionalidad tienen, las transiciones y operativas básicas deben estar bien indicadas e implementadas. El diseño visual debe ser lo mas cuidado posible aunque a veces en juegos para este tipo de dispositivos no es el que más domina, sobre todo si el juego tiene una mecánica muy adictiva, aunque si se consiguen las dos cosas mucho mejor. Otro punto importante es tener una mecánica freemium implementada dentro del juego con una tienda para comprar mejoras aplicables a la mecánica de juego. Otro punto importante es la vinculación del juego a redes sociales y que recoja puntuaciones a nivel mundial y no locales del dispositivo, todos estos ingredientes le pueden dar al juego un plus que haga que el usuario vea el juego más atractivo.

Debido a temas de tiempo, inicialmente se descarta para este proyecto la implementación de las siguientes funcionalidades dentro del videojuego: las puntuaciones a nivel mundial, estas de momento quedarán sujetas a nivel local del dispositivo, también se descarta la vinculación a redes sociales y mecánicas freemium.

El resto de puntos comentados como las mecánicas sencillas y adictivas, aspecto visual lo más cuidado posible dentro de los recursos de que se dispone, y la navegación entre pantallas y menús claros y sencillos será lo que se transportará al diseño y la posterior implementación del proyecto. Es posible que durante la implementación del diseño mostrado en este documento se modifiquen algunos puntos por temas de balanceo y equilibrio del producto final.

3.2 Perfiles de usuarios identificados

Los juegos casuales, son juegos con mecánicas simples. Este hecho hace que estén al alcance de cualquier usuario aunque éste no sea un jugador experimentado acostumbrado a utilizar juegos con mecánicas más complicadas que requieren de cierta experiencia previa. Al mismo tiempo, son juegos que pueden ser jugados en cortos espacios de tiempo. Por tanto, el tipo de usuario de estos juegos sera muy variado. Podemos encontrar el perfil de usuario experimentado, tanto en las tecnologías como en los videojuegos y que tiene interés por cualquier tipo de juego. Y por otro lado, encontramos el usuario poco experimentado pero que gracias a los smarphones y la sencillez de las mecánicas de juego provoca que también juegue .

Todos tendrán un objetivo común, pasar un rato jugando al juego. Para mejorar su último récord. Aunque, en este tipo de juegos el usuario también puede jugar sin buscar ningún tipo de meta concreta, simplemente pasar un rato divertido sin necesidad de alcanzar ningún tipo de récord ni objetivo concreto. También es importante mencionar que aunque los perfiles de usuarios son muy variados, existe una fuerte tendencia a que los jugadores de este tipo de juegos, tiendan a ser más grandes que los jugadores de otros tipos de videojuegos y suelen tener un interés y tiempo limitado tanto para jugar como los videojuegos en general, siendo normalmente del sexo femenino.

Se detectan dos grandes grupos de perfiles de usuario, los experimentados en jugar a videojuegos y los no experimentados en jugar videojuegos, edades diferentes, profesiones variadas, educación variada, sexo variado. Los usuarios de Android en general suelen pagar mucho menos por aplicaciones que otros grupos de usuarios como los de iphone por ejemplo.

Los hábitos de consumo en los estados unidos, ya no de los usuarios de juegos casuales, sino de los usuarios de Android en general se definen de la siguiente manera: sólo un 4% de estos paga por descargar una aplicación contra un 23% y un 31% de los usuarios de la App Store para iPhone y iPad respectivamente, tal y como muestra un estudio realizado por la empresa Statista con las 100 aplicaciones más descargadas.

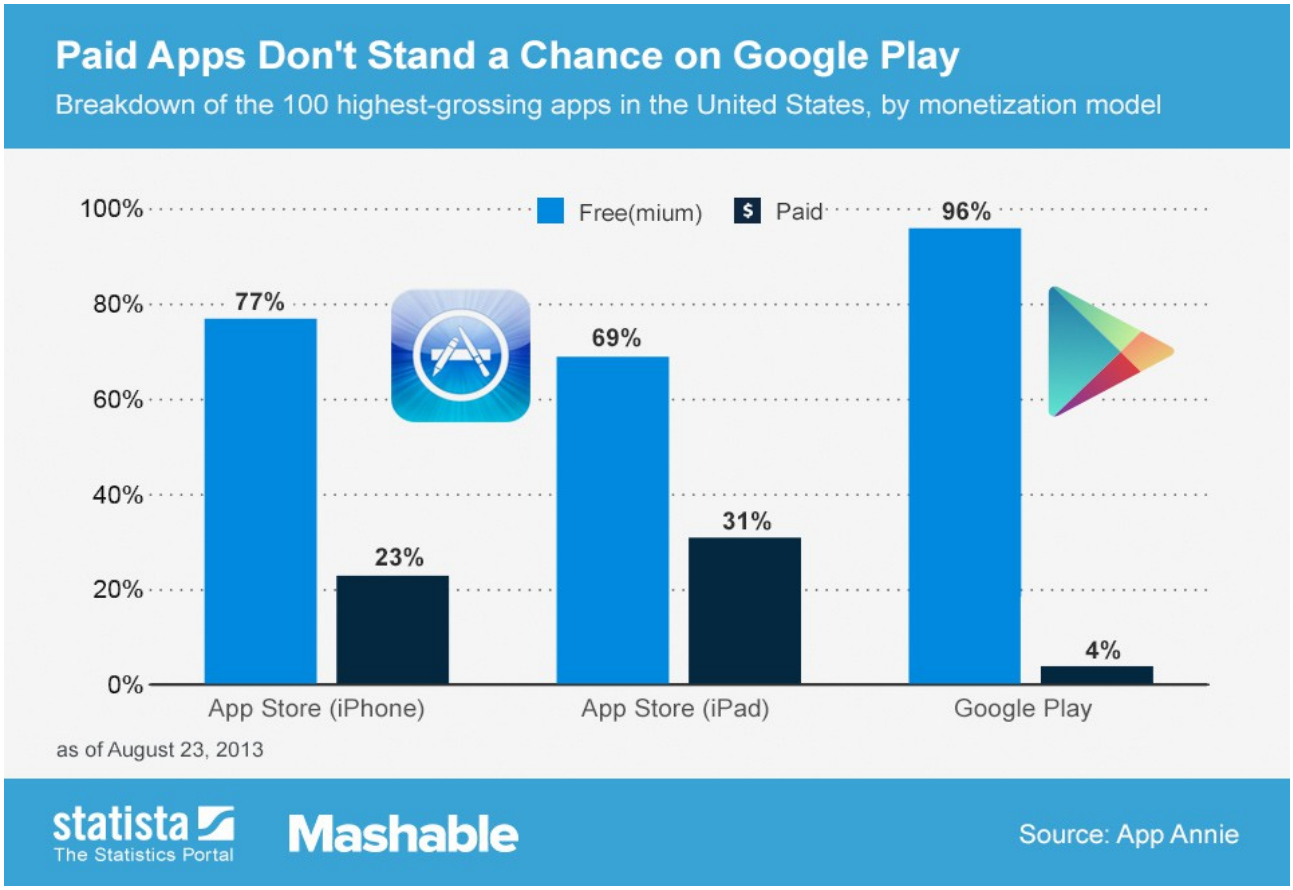


ilustración 20. Aplicaciones de pago vs aplicaciones gratuitas

Se puede ver en linea en el link:
http://www.statista.com/topics/983/mobile-app-monetization/chart/1400/paid-apps-don-t-stand-a-chance-on-google-play/

Se puede ver otro articulo similar en el siguiente link:

http://www.wayerless.com/2013/07/usuarios-de-android-pagan-menos-por-apps-que-los-de-ios/

La mayoría de aplicaciones gratuitas, utilizan la modalidad freemium que no es más que ofrecer un producto básico de forma gratuita y cobrar a los usuarios por opciones más avanzadas o especiales dentro de la misma aplicación. No obstante, también es posible encontrar aplicaciones completamente gratuitas sin la modalidad freemium.

Como ya se ha dicho, se han detectado inicialmente dos grandes grupos dentro de la masa de usuarios que son consumidores potenciales de juegos casuales. Estos son los usuarios asiduos a jugar videojuegos de todo tipo y con experiencia en el uso de tecnologías en general, es decir usuarios experimentados. Por otro lado, encontramos los que no tienen costumbre de jugar mucho y normalmente tampoco tienen una gran experiencia en el uso de tecnologías en general, pero

que debido a la simpleza de los videojuegos casuales, éstos también se convierten en jugadores y en muchas ocasiones se convierten en los usuarios que consumen más este tipo de juegos. Así pues encontramos dos grandes grupos según unas características comunes: experiencia previa o sin experiencia ante los videojuegos y las tecnologías .

3.3 Contextos de uso

3.3.1 ¿ Dónde ?

Al tratarse de un casual game para un dispositivo móvil, el espacio de utilización será muy variado. Por tanto, los usuarios podrán encontrarse en muchos lugares distintos, como pueden ser, en casa, en el tren, en el metro, en el autobús, al aire libre. etc. En definitiva el usuario podrá encontrarse en cualquier lugar.

3.3.2 ¿ Cuándo ?

El tipo de juegos casuales, tienden a ser juegos donde los usuarios realizan partidas cortas, pero al mismo tiempo la naturaleza adictiva de estos, hace que los usuarios puedan quedar atrapados durante varias horas repartidas en ratos diferentes o continuados dependiendo de la situación de cada usuario. Por tanto, el cuándo de este tipo de juegos puede ser muy variado casi en cualquier momento que el usuario pueda dedicarle unos minutos, o ratos más largos: durante su tiempo libre, mientras se desplaza ya sea en tren, en metro o en algún otro medio de transporte que no necesite estar pendiente del vehículo, cuando esta en la cama, etc.

3.3.3 ¿ En que entorno ?

Los entornos de utilización pueden también ser muy variados promovidos por dónde y el cuándo. Por lo tanto, el entorno tecnológico podrá también ser variado, puede existir o no conexión a la red, en un entorno social muy diferente que podrá ser más privado o más público y consecuentemente se dibujará un entorno ambiental variado según el momento.

4.Diseño y “Concept Art”

4.1 Elementos presentes en la interfaz de la aplicación

Con todo lo que se ha visto en el apartado 3.1 estudio comparativo, queda patente que la interfaz de usuario debe ser clara y sencilla. Será necesario que contenga las siguientes pantallas:

- Pantalla de intro inicial que aparecerá durante un segundo.
- Pantalla menú inicial del juego con las opciones: Play, Scores, Options, (?) How to play
- Pantalla de nivel donde se desarrolla el videojuego.
- Pantalla high scores que mostrara la tabla de récords con las 10 puntuaciones más altas.
- Pantalla how to play, instrucciones de como jugar, también se integraran dentro del videojuego y aparecerán durante la primera partida que se juegue.
- Pantalla opciones, podremos activar y desactivar el sonido y los efectos de audio. A su vez desde esta pantalla podremos acceder a la pantalla de créditos.
- Pantalla de créditos, pantalla dónde se mostraran lo créditos del videojuego.
- Pantalla Ready ?, pantalla de juego con el mensaje sobre impreso antes de empezar a jugar, aparecerá durante 3 segundos.
- Pantalla con los botones resume, quit, sobreimpresos en la pantalla de juego, se accederá

a ella con el botón de pausa, pausando el juego. Desde ella podremos reanudar el juego o abandonar el juego.

- Pantalla game over con el resumen de los récords conseguidos durante la partida, puntos,tiempo jugado, numero de lobos ahuyentados y ovejas colocadas en el corral. Desde esta pantalla tendremos dos botones, Play again y Exit, uno nos permitirá volver a jugar y el otro nos permitirá abandonar el juego, tal y como su nombre indica.

4.2 Flujos de interacción

En la siguiente ilustración se muestran los flujos de interacción definidos con las pantallas enumeradas en el apartado anterior. Los flujos muestran la navegación que debe existir entre las pantallas del videojuego.

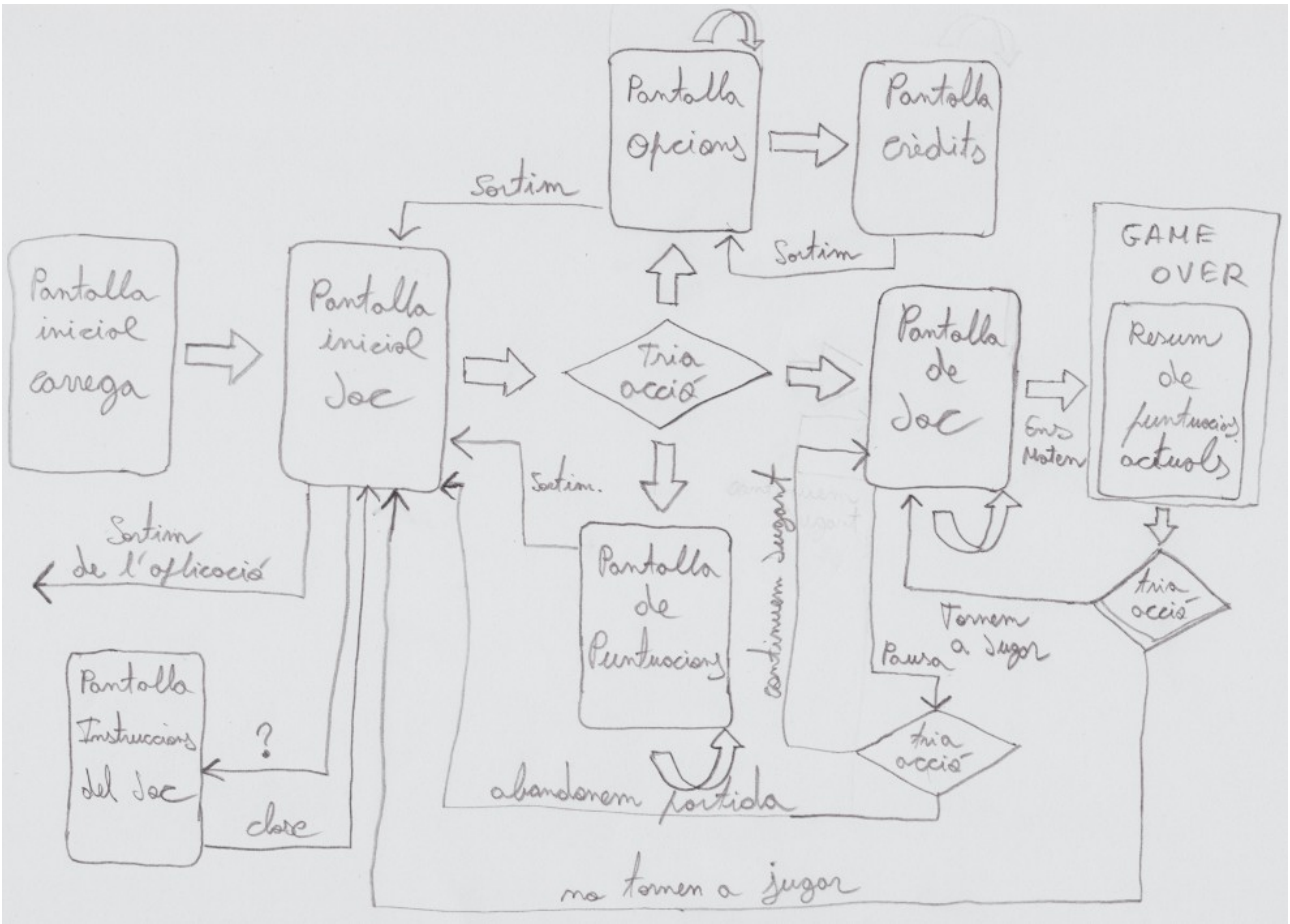


ilustración 21. Flujos de interacción de la aplicación entre sus distintas pantallas

4.3 Prototipaje

4.3.1 Sketches

En este apartado se muestran los primeros esbozos realizados a mano alzada, estos intentan plasmar el diseño del juego en su totalidad. Estos esbozos servirán como punto de partida en una segunda fase para realizar el prototipaje de todas las pantallas que intervienen en el videojuego utilizando todo el material recopilado en la red.

Tal y como ya se ha dicho en este mismo documento anteriormente, no se dispone de conocimientos de diseño suficientes como para abordar un diseño de arte desde cero. Por lo tanto se han utilizado recursos ya disponibles en la red para el montaje de todos los gráficos del videojuego. Todos los recursos utilizados están bajo licencias creative commons o dominio público . A continuación se muestran los bocetos.

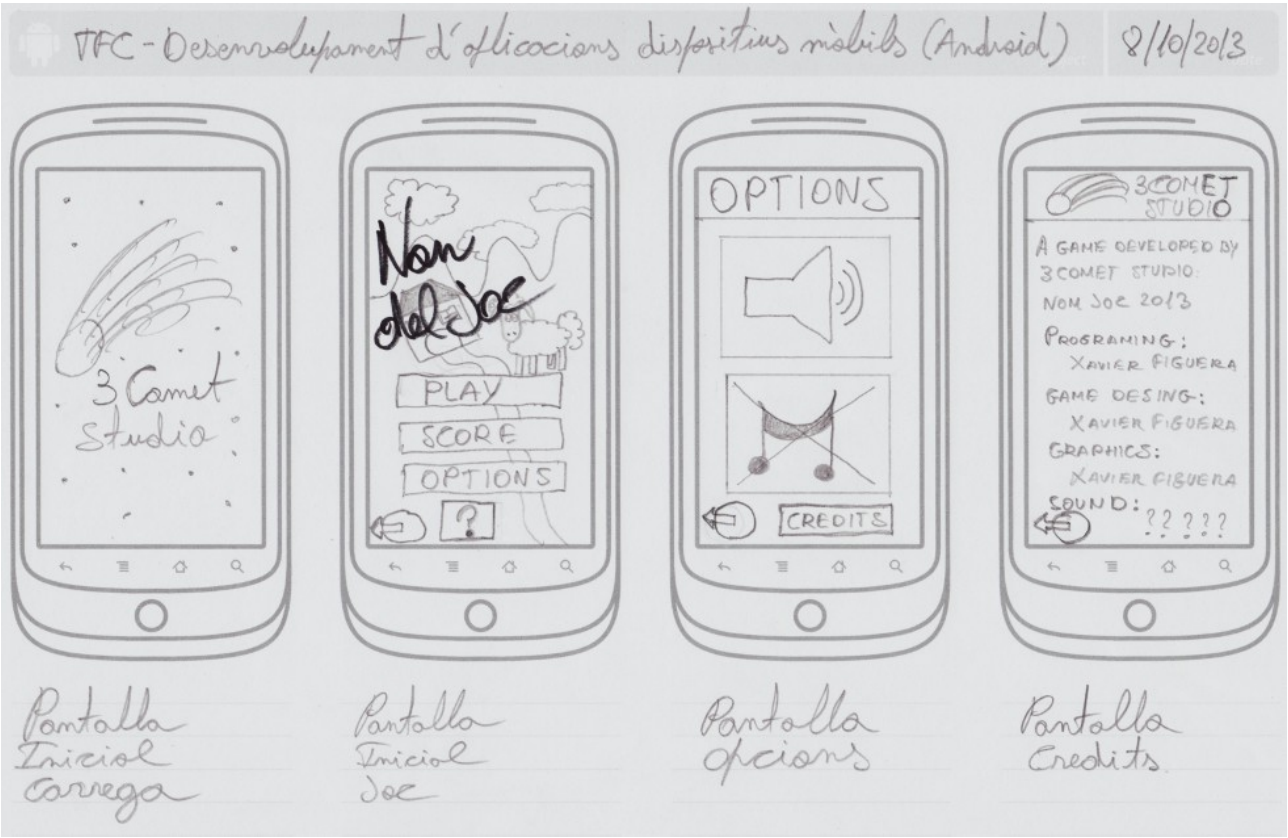


ilustración 22. Bocetos pantallas videojuego realizadas a mano alzada

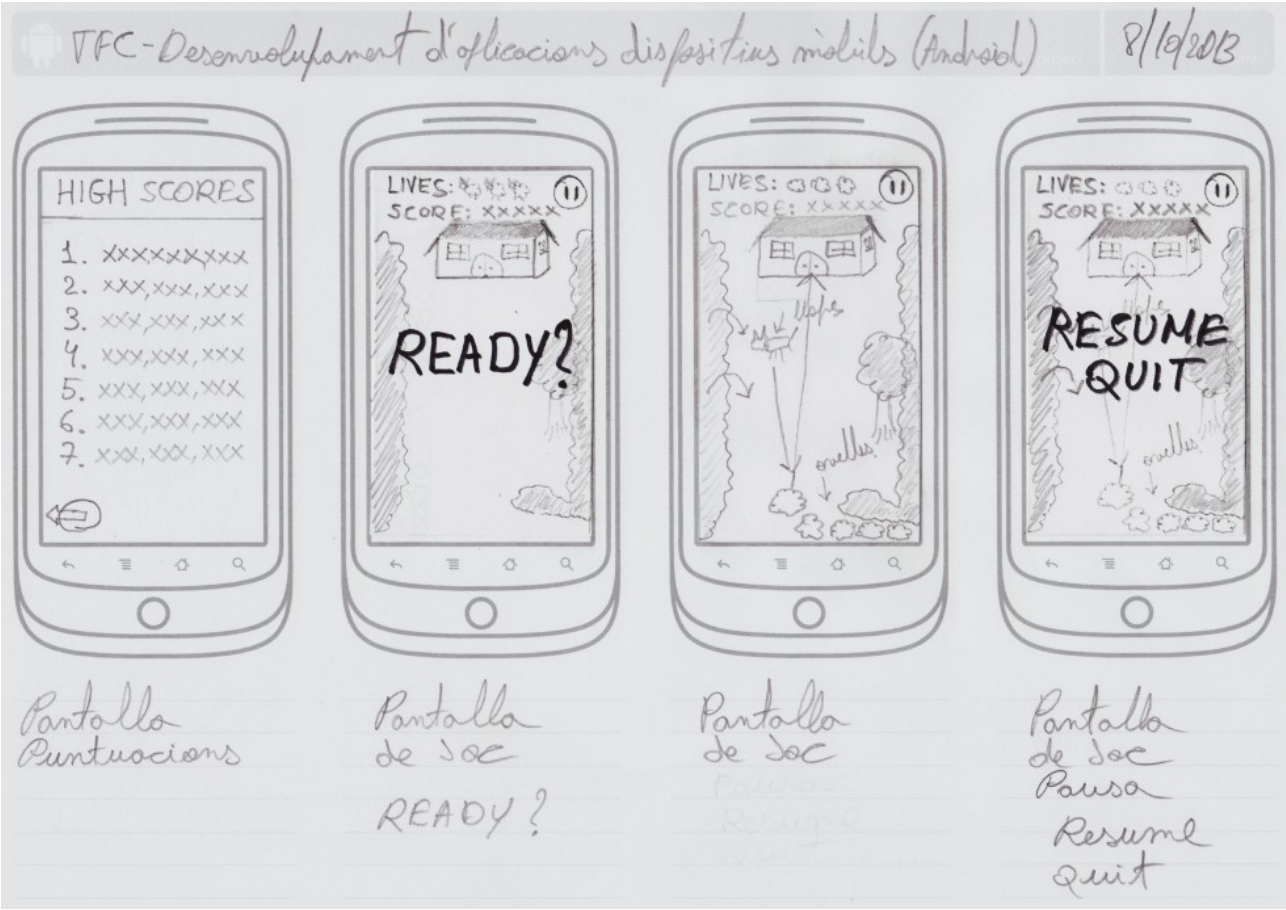


ilustración 23. Bocetos pantallas videojuego realizadas a mano alzada

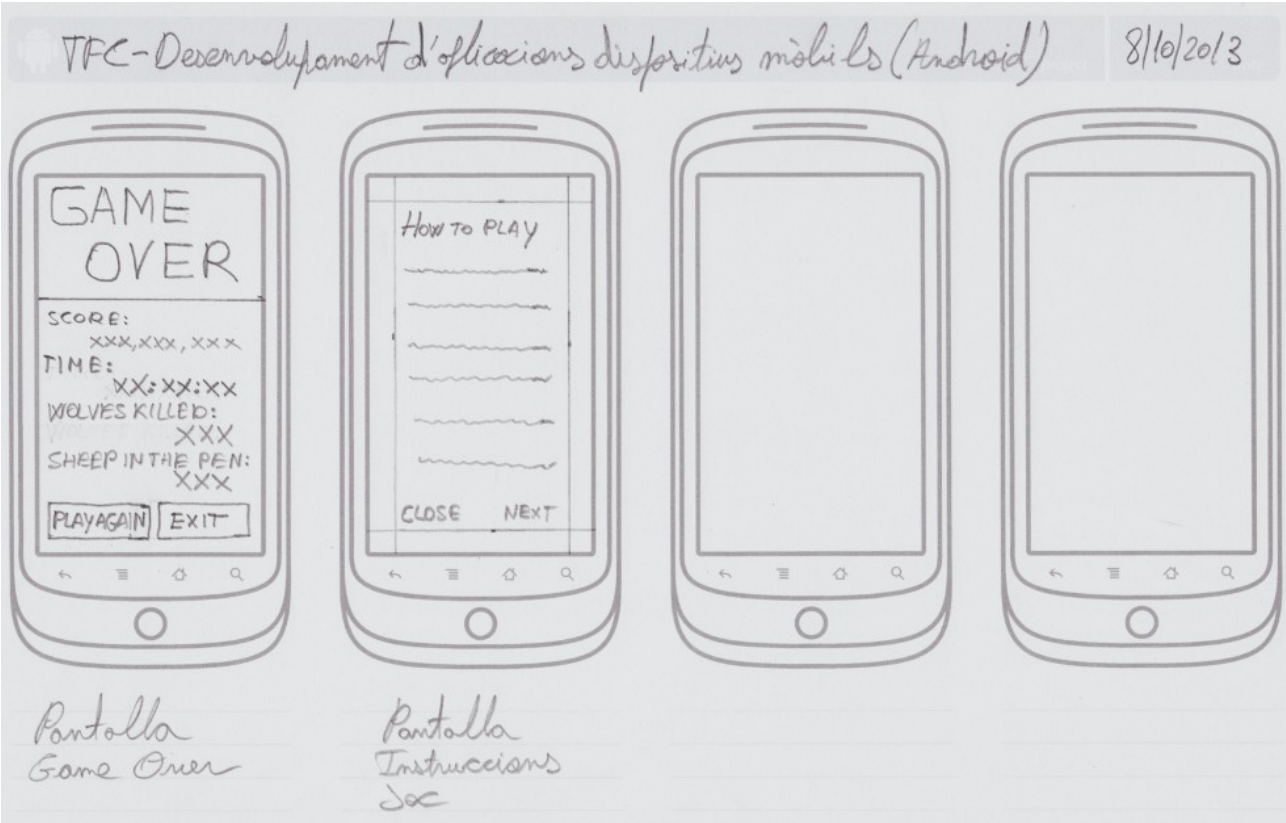


ilustración 24. Bocetos pantallas videojuego realizadas a mano alzada

En la siguiente página, se muestran las transiciones definidas entre pantallas con los sketches realizados, ilustración 24.1. Las transiciones toman como base el diagrama de flujos de interacción definido en el apartado 4.2 de este documento, ver ilustración 21.

4.3.2 Transiciones definidas entre pantallas

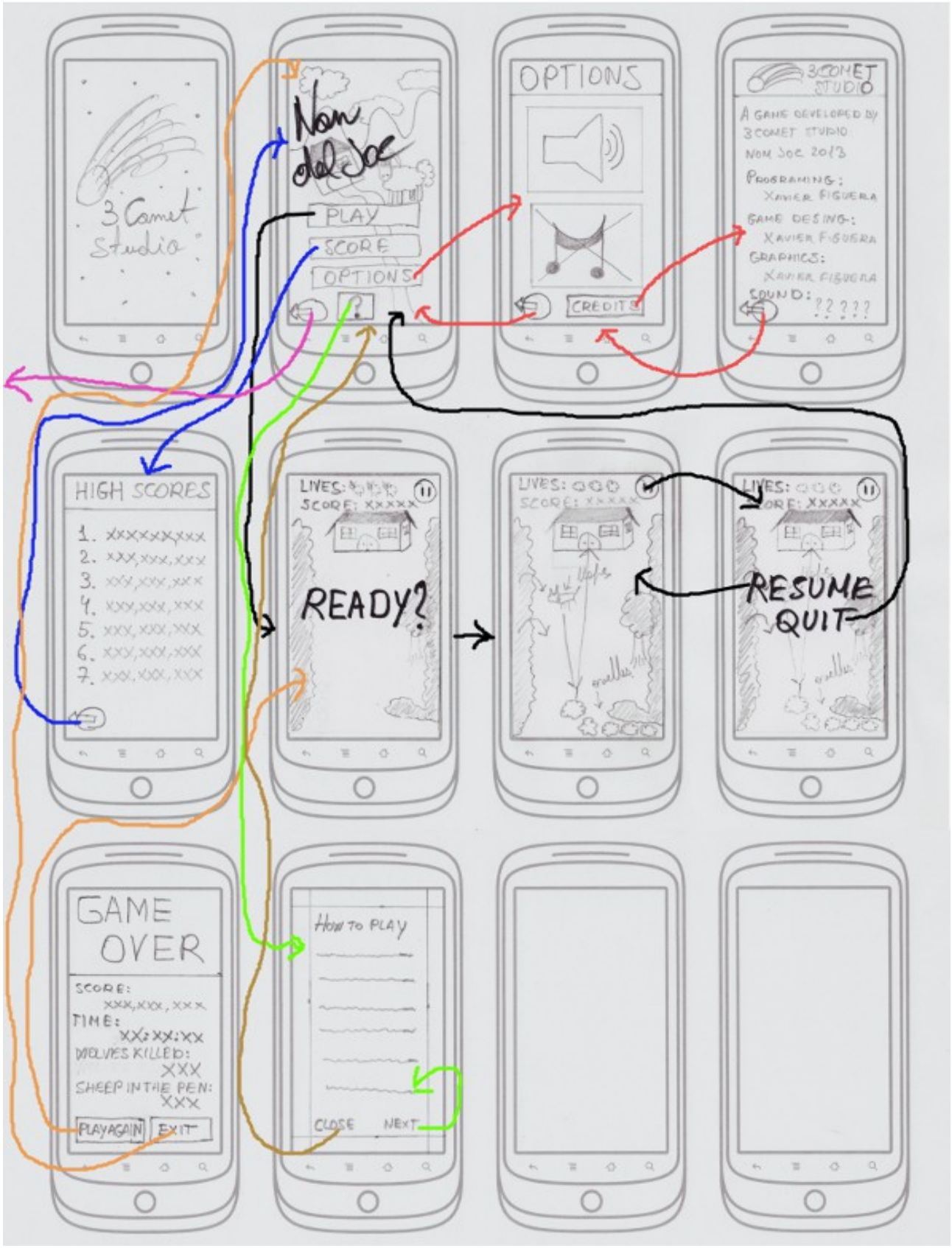


ilustración 24.1. Transiciones definidas entre pantallas

4.3.3 Prototipo horizontal de alta fidelidad

4.3.3.1 Pantalla inicial

Pantalla de inicio, al ejecutar el juego aparecerá esta pantalla durante un segundo.

4.3.3.2 Pantalla menú principal del juego

Pantalla de menú principal del juego, desde esta pantalla podremos empezar a jugar (botón Play), consultar la tabla de puntuaciones (botón Scores) e ir a la pantalla de opciones del juego (botón Options). El interrogante de la parte inferior derecha nos servirá para ir a la pantalla de instrucciones "HOW TO PLAY" para saber cómo se juega.

4.3.3.3 Pantalla de opciones

La pantalla de opciones, nos permitirá deshabilitar la música del juego y los efectos sonoros pasando el dedo sobre los iconos del altavoz y las notas musicales, con el altavoz deshabilitaremos la música y con las notas musicales desactivaremos los efectos de audio. La flecha naranja nos conducirá de nuevo a la pantalla de inicio del juego, el botón créditos nos llevara a la pantalla de créditos.

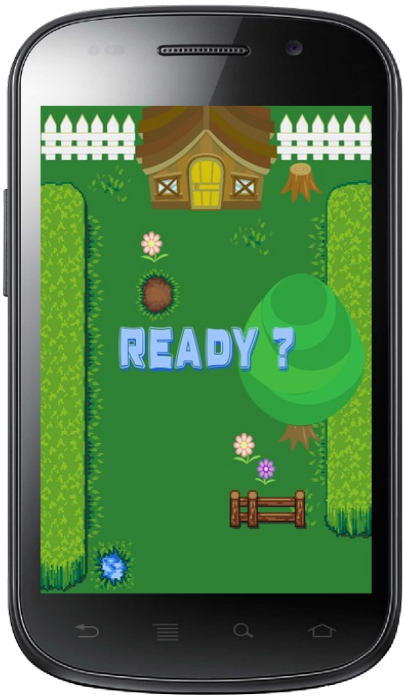
4.3.3.4 Pantalla de créditos

Pantalla de créditos, en ella se muestran los nombres de quienes han diseñado y programado el juego, con la flecha naranja inferior volvemos a la pantalla de opciones. Deslizando el dedo sobre el texto se podrá desplazar hacia arriba y hacia abajo.

4.3.3.5 Pantalla tabla de puntuaciones

Pantalla tabla de puntuaciones, nos mostrara una tabla con las 10 puntuaciones más altas de todas las partidas jugadas. Esta tabla se guardara de forma local en el dispositivo.

4.3.3.6 Pantalla de juego READY ?



Pantalla de juego READY ?, Aparecerá durante unos segundos una vez iniciada la partida. Después de 3 segundos pasara a la pantalla de juego que se muestra en el siguiente apartado.

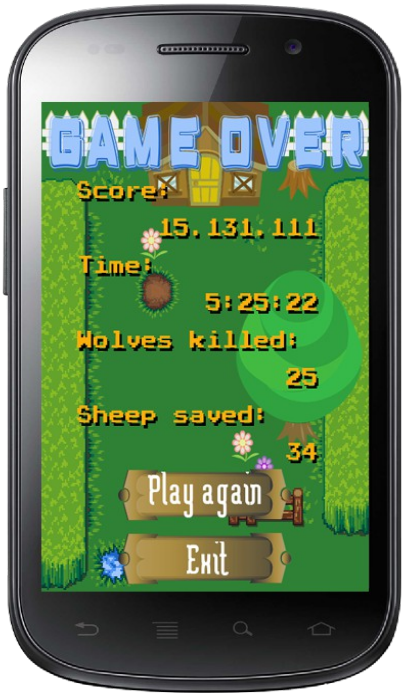
4.3.3.7 Pantalla de juego

La pantalla de la izquierda, es donde se desarrollará todo el gameplay, en la parte superior izquierda de la pantalla se muestran los puntos conseguidos y las vidas. En la parte superior derecha hay un botón que permitirá pausar el juego, una vez pausado mostrará la pantalla de pausa con los botones RESUME y QUIT, la pantalla se muestra en el apartado siguiente. En la parte derecha se muestra otra pantalla idéntica a la de la izquierda, donde se puede ver el gameplay básico representado con flechas: los lobos saldrán de los extremos de la pantalla yendo hacia las ovejas, las ovejas al mismo tiempo saldrán de la derecha de la pantalla y las tendremos que lanzarlas deslizando el dedo sobre ellas y lanzandolas hacia el establo, La puerta se irá abriendo y cerrando aleatoriamente, los lobos una vez puestos en escena los podremos ahuyentar deslizando el dedo sobre ellos un número de veces determinado en función del color del lobo, como esta indicado en el apartado 1.3.1 de este documento.

4.3.3.8 Pantalla de pausa botones RESUME y QUIT

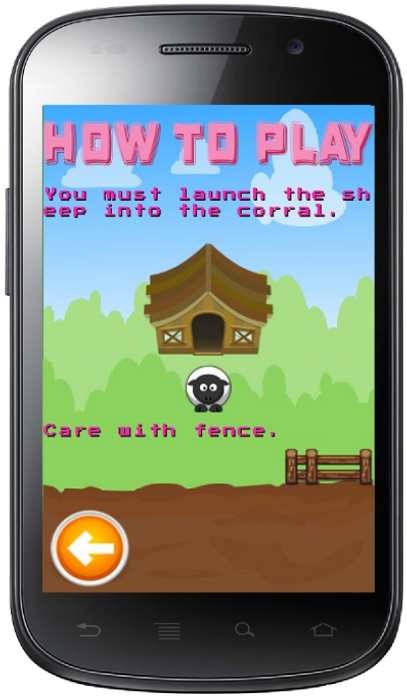
Pantalla de pausa que aparecerá cuando se pause el juego. Con la opción RESUME continuaremos jugando, con la opción QUIT se saldrá de la partida y pasará a la pantalla del menú principal del juego descrita en el punto 3.2.2.

4.3.3.9 Pantalla GAME OVER



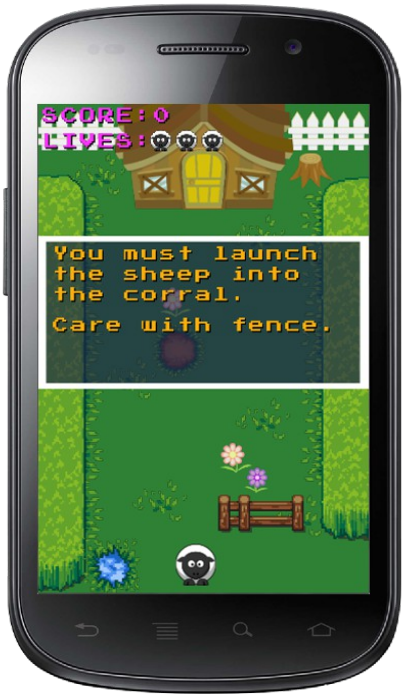
Pantalla GAME OVER, aparecerá cuando se hayan agotado todas las vidas. Mostrará un resumen de los puntos conseguidos en la partida actual, tiempo invertido en la partida, lobos ahuyentados y ovejas salvadas. Con los botones de la parte inferior podremos volver a jugar (botón play again) o salir del juego con el botón exit, este botón nos conducirá a la pantalla de inicio de juego descrita en el apartado 3.2.2.

4.3.3.10 Pantalla de instrucciones de juego HOW TO PLAY



Pantalla de instrucciones del juego, nos hará una breve explicación de cómo jugar. Deslizando el dedo sobre el texto se podrá desplazar hacia arriba y hacia abajo. El juego dispondrá de unas mini instrucciones in game que aparecerán en la primera partida que se juegue en forma de cajas de texto, esta pantalla se muestra en el siguiente apartado.

4.3.3.11 Pantalla de instrucciones in game



Esta pantalla muestra el modo como quedarán integradas las instrucciones de juego dentro del gameplay. Estas instrucciones aparecerán solamente la primera vez que se juegue.

Todo el diseño mostrado en este apartado, ha sido realizado con todos los recursos encontrados en la red, todos ellos quedan recogidos en el apartado 6. Con el programa Gimp se ha montado todo el diseño de alta fidelidad mostrado en este apartado. Se ha trabajado con capas para disponer de flexibilidad en la construcción del diseño. A su vez, el hecho de tener los diseños por capas, nos permitirá poderlo exportar a la implementación del videojuego de una manera cómoda y rápida, en el apartado 5 implementación, se hace referencia a como convertir el diseño realizado en capas a recursos gráficos del videojuego.

En la siguiente ilustración se muestra una captura del diseño de la pantalla de juego descrita en el apartado 4.3.3.7 realizado con Gimp.

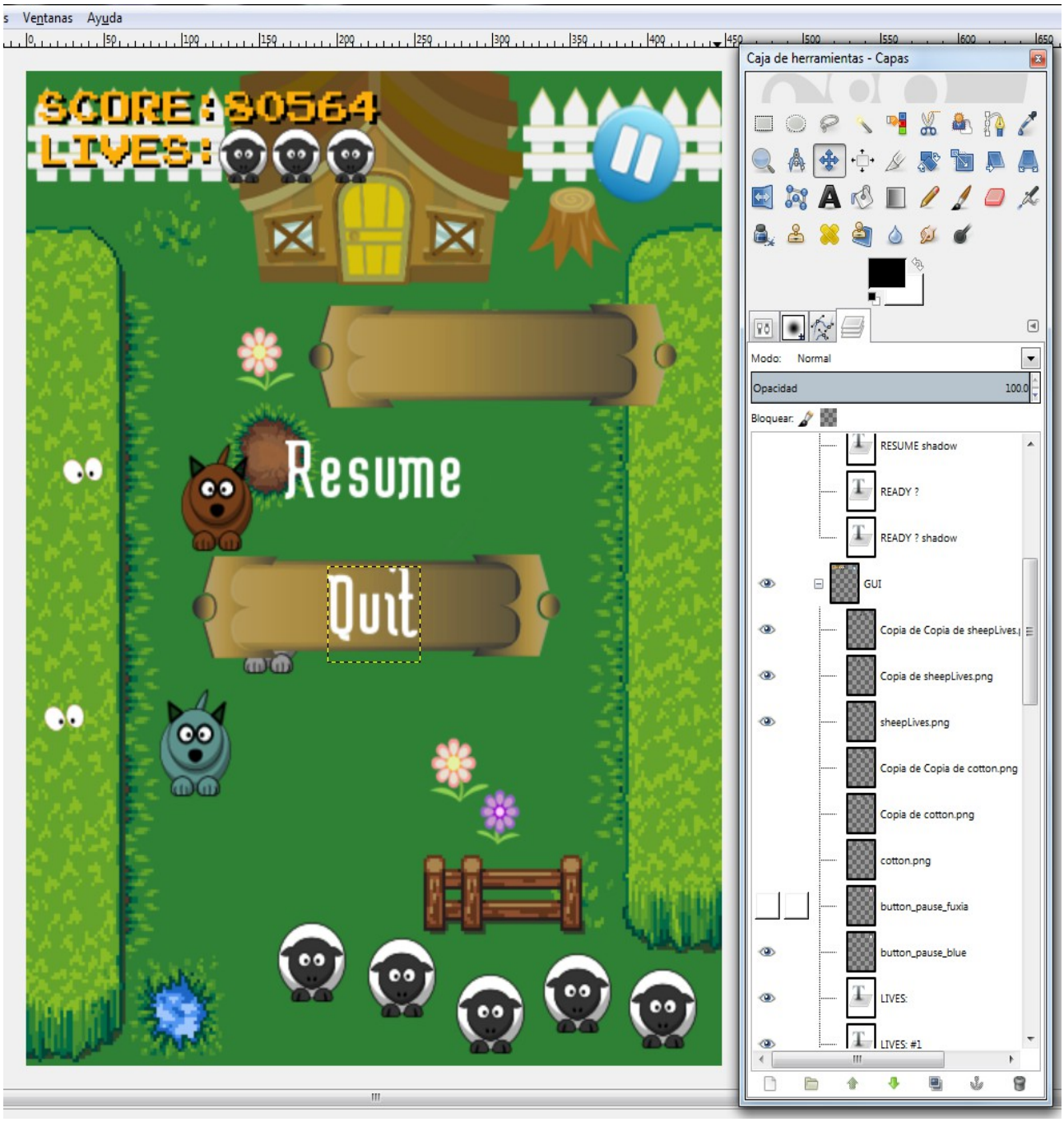


ilustración 24.2. Diseño pantallas con Gimp

5.Implementación

5.1 Arquitectura global

5.1.1 Arquitectura del framework libGDX

LibGDX es un framework multiplataforma que actualmente soporta Windows,Linux,Mac OS X, Android, iOS y HTML5. Esto permite escribir el código una sola vez y desplegarlo en las distintas plataformas comentadas, sin tener que modificar el código. LibGDX permite bajar a más bajo nivel si se desea, ya que da acceso directo al sistema de ficheros, dispositivos de entrada, de sonido y a OpenGL a través de la interfaz unificada OpenGL ES1.1 y 2.0.

LibGDX ofrece a su vez, un conjunto de APIs que ayudan en las tareas comunes de desarrollo de videojuegos, como son la presentación de sprites, texto, la creación de interfaces de usuario, la reproducción de efectos de sonido y secuencias musicales y los distintos tipos de cálculos necesarios en la programación de videojuegos.

En algunas ocasiones LibGDX deja de lado el lenguaje Java y recurre a código nativo para conseguir un mejor rendimiento. No obstante, todas estas funciones quedan ocultas detrás de las APIs de Java. De manera que en el momento de desarrollar un videojuego no hay que preocuparse de realizar una compilación cruzada entre Java y C++.

LibGDX es un framework más que un motor de videojuegos, que ayuda al desarrollo de estos, gracias a las abstracciones de gran alcance que ofrece y deja en manos del programador elegir como escribir su videojuego o aplicación.

5.1.2 Visión de los módulos de libGDX

LibGDX comprende varios módulos que proporcionan los servicios necesarios para crear la arquitectura típica de un videojuego.

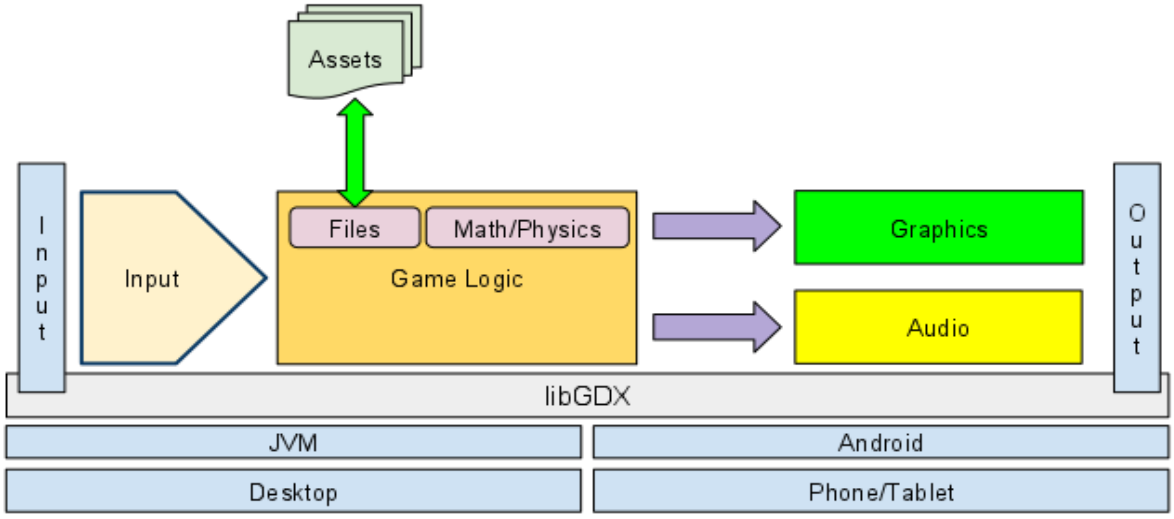


ilustración 25. Diagrama de los módulos que intervienen en un videojuego.

A continuación de describen los módulos mostrados en la ilustración 25.

Input: Proporciona un modelo de entrada unificado para todas las plataformas: Soporta teclado, pantalla táctil, acelerómetro y ratón si esta disponible.

Graphics: Permite el dibujo de las imágenes en pantalla usando el hardware proporcionado por la implementación de Open GL ES, las versiones soportadas son GL versión 1.0 ,1.1, 2.0.

Files: Proporciona los métodos necesarios para las operaciones de lectura / escritura totalmente independientes a los medios de comunicación existente.

Audio: Proporciona las herramientas necesarias para la reproducción de sonido para todas las plataformas.

Math: Módulo que proporciona los cálculos matemáticos necesarios orientados al desarrollo de videojuegos.

Physics: LibGDX proporciona un wrapper completo al motor de físicas en 2D open source desarrollado en C++ llamado Box2D.

5.2. Arquitectura de un videojuego

5.2.1 Programa vs Aplicación

Una aplicación convencional sigue una secuencia lógica lineal, reacciona cuando sucede algún evento, es decir responde a eventos como por ejemplo cuando se pulsa una tecla. Por el contrario un videojuego es más complejo, hace cálculos y dibuja en pantalla constantemente, aunque no se produzca ningún evento.

5.2.2 Estructura básica de un videojuego

Programar videojuegos es una forma completamente diferente de programar software. La diferencia principal entre una aplicación convencional y un videojuego radica en la gestión de eventos tal y como se ha indicado anteriormente: un programa ofrece respuestas a eventos y, una vez son servidos, permanece a la espera de una nueva orden, en cambio, un videojuego es un programa que debe actuar en tiempo real, tiene que estar haciendo cálculos y dibujando en pantalla constantemente. No se puede esperar a que suceda un evento para poder actuar: aunque el jugador no haga nada, no presione una sola tecla, no mueva el ratón, el juego tiene que calcular el tiempo que lleva jugando en ese nivel, si le va a atacar algún enemigo, si está cargando energía y, por supuesto, dibujar en pantalla todo lo que va sucediendo.

La mayoría de videojuegos están construidos bajo la misma estructura básica sobre la cual corre el programa. Esta estructura puede tener variaciones, pero de forma general se presenta de la siguiente manera:

Inicialización: En esta primera etapa el juego se sitúa en su estado inicial. Se realizan las inicializaciones por lo que respecta a las librerías o motores que vayan a utilizarse, de variables y estructuras de datos referentes a los atributos de las entidades o personajes, escenarios, configuraciones, etc., y de los diferentes recursos físicos que vayan a emplearse, tales como gráficos o sonidos.

Ciclo de juego: En esta parte esta toda la acción, este ciclo se repite de manera indefinida hasta que el jugador pierda, gane o salga del juego. Este punto se puede dividir en tres partes que podemos llamarlas: entrada, proceso y salida

Como **entrada** se entiende, el punto donde se captura todo lo que hace el jugador, como presionar los botones de control, mover el ratón, presionar las flechas del teclado, tocar la pantalla táctil y el resto de información que recibe el juego.

En la parte de **proceso**, es donde se procesa toda la información que se recibió en la entrada y tiene lugar la lógica del videojuego. Por tanto, es donde se realizan los cálculos de físicas si el juego los requiere, así como también los cálculos de la inteligencia artificial de este, conjuntamente con el tratamiento gráfico.

La **salida** es la encargada de enviar al jugador toda la información que se ha procesado en la etapa anterior "proceso". Se muestra al jugador el resultado de lo que hizo con la visualización en pantalla del estado actual del videojuego. También es en este punto donde se hace sonar la música y efectos de sonido.

Por último y fuera del ciclo de juego esta la parte de **finalización**, esta parte es la encargada de liberar todos los recursos y memoria utilizada durante el transcurso del videojuego.

Es importante diferenciar estas tres etapas del ciclo del videojuego y ver qué tarea se delega en cada una de ellas: en la entrada tan sólo nos ocupamos de captar los posibles eventos; el proceso se encarga en primer lugar de procesar la entrada y más adelante de la lógica del videojuego, y, por último, la capa de salida se encarga de visualizar y reproducir la música.

Con estos puntos claros se puede mostrar la estructura principal de un videojuego. En el siguiente apartado se muestra esta estructura básica a modo de esquema.

5.3 Esquema de la estructura básica de un videojuego

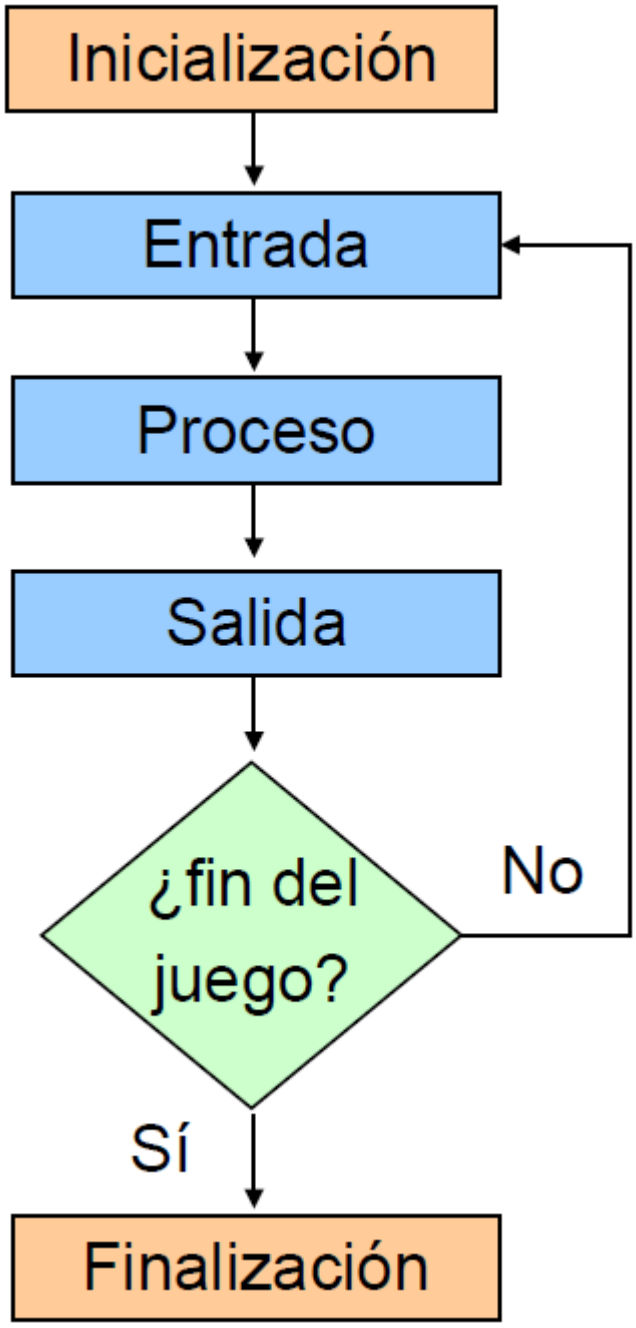


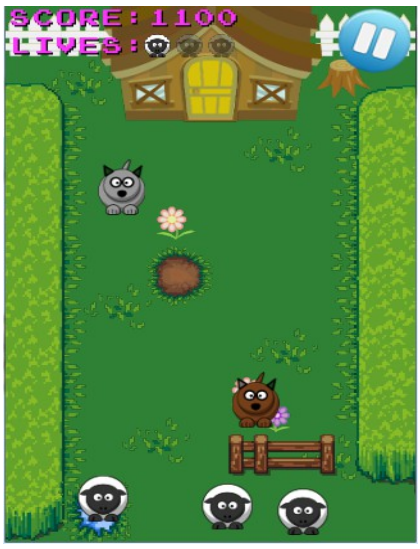
ilustración 26. Estructura básica de un videojuego

Inicialización: Creación de estructuras de datos, carga de recursos gráficos y sonidos

Entrada: Gestión de pantalla táctil, joystick, teclado.

Proceso: Lanzamos una oveja al corral, rebota la oveja, salta un lobo al escenario etc.

Salida: Mostrar gráficos y reproducir sonidos.



Finalización: Liberar recursos.

5.4 Patrones de diseño

5.4.1 Introducción

En el diseño de cualquier aplicación como por ejemplo un videojuego,no es posible abordarlo por completo. El proceso de diseño de una aplicación suele ser iterativo y se realiza en diferentes etapas de forma que se vaya refinando con el tiempo. Conseguir un diseño perfecto a la primera es muy difícil de conseguir. El diseño de aplicaciones es complejo y la más importante y que más impacto tiene, no sólo sobre el producto final, sino también sobre su vida futura. Ya que es en el diseño en donde se definen las estructuras y entidades que se van a encargar de resolver el problema planteado. Como de bien se definan estas estructuras y entidades influirá en gran medida , en el éxito o fracaso del proyecto y en la viabilidad de su mantenimiento.

Los patrones de diseño son formas reconocidas y probadas de resolver problemas de diseño que son recurrentes en el tiempo. No es más que basarse en experiencias anteriores en problemas similares aplicado soluciones probadas anteriormente en determinados contextos para alcanzar un mejor diseño más rápidamente.

Existen distintos patrones, para distintos problemas de diseño. No obstante, en este documento sólo se hará hincapié en los patrones utilizados para este proyecto. Quedando fuera del alcance de este documento los demás patrones existentes.

5.4.2 Patrones de creación

Estos patrones nos proporcionan una solución relacionada con la construcción de clases, objetos y otras estructuras de datos. Algunos ejemplos de patrones pueden ser *Abstract Factory, Builder, Singleton*, estos ofrecen mecanismos de creación de instancias de objetos y estructuras escalables dependiendo de las necesidades.

5.4.3 Singleton

Para este proyecto se ha utilizado este patrón ya que era necesario tener una única instancia de algunas determinadas clases. En java cuando se utiliza el operador *new* es posible crear una instancia de un objeto. No obstante a veces es necesario tener una única instancia de un objeto, por ejemplo para poder acceder a ella des de cualquier punto de la aplicación. Seria un ejemplo,la pelota de un juego de pinball en donde podría ser conveniente mantener una única instancia de este objeto.

La solución para garantizar que solo existe una instancia de un objeto se consigue impidiendo que ninguna otra clase pueda acceder al constructor, por este motivo de declara el constructor como mínimo de forma protegida o privada y se proporciona un único punto (controlado) en donde se proporciona la única instancia. En la siguiente ilustración se muestra el diagrama de clases del patrón *Singleton.*

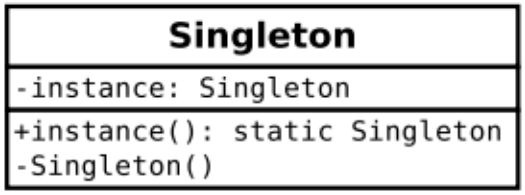


ilustración 27. Singleton

A continuación en el listado 5.4.3, se muestra una implementación básica del patrón *Singleton* utilizada en el proyecto para la clase que maneja el audio del videojuego.

Listado 5.4.3

```
1.      package com.threecomet.flingthesheep.util;
2.
3.      import com.badlogic.gdx.audio.Music;
4.      import com.badlogic.gdx.audio.Sound;
5.
6.      public class AudioManager
7.      {
8.              public static final AudioManager instance = new AudioManager();
9.
10.             private Music playingMusic;
11.             private Sound playingSound;
12.
13.             //Singleton
14.             private AudioManager()
15.             {
16.
17.             }
18.
19.             public void ResunmeMusic()
20.             {
21.                if(playingMusic != null) playingMusic.play();
22.             }
23.
24.             public void PlayMusic(Music music,float volume,boolean loop)
25.             {
26.                if(!GamePreferences.instance.music) return;
27.                playingMusic = music;
28.                music.setLooping(loop);
29.                music.setVolume(volume);
30.                music.play();
31.             }
32.
33.             public void StopMusic()
34.             {
35.                if(playingMusic != null) playingMusic.stop();
36.             }
37.
38.             public void PauseMusic()
39.             {
40.                if(playingMusic != null) {
41.                  if(playingMusic.isPlaying()) {
42.                      playingMusic.pause();
43.                  }
44.                }
45.             }
46.
47.             public void PlaySound(Sound sound,float volume)
48.             {
49.                if(!GamePreferences.instance.sound) return;
50.                playingSound = sound;
51.                sound.play(volume);
52.             }
53.
54.             public void PlaySound(Sound sound,float volume,float pitch,float pan)
55.             {
56.                if(!GamePreferences.instance.sound) return;
57.                playingSound = sound;
58.                sound.play(volume, pitch, pan);
59.             }
60.     }
```

Esta clase maneja el audio del videojuego, tanto efectos de sonido, como la música, es una clase

que debe estar accesible des de cualquier punto de la aplicación y no tiene ningún sentido que se instancie repetidas veces sólo sera necesaria una instancia de dicha clase para poder manejar los audios y músicas. Como se puede ver en la linea 14 del listado 5.4.3 se declara el constructor privado y en la linea 8 se proporciona el punto donde se proporciona la única instancia de dicha clase a través de la variable *instance* declarada de forma publica para que se accesible desde cualquier otra clase y a su vez declarada de forma *static* y *final,* Con *final* le decimos que esta variable ya no puede ser modificada una vez declarada, es decir sera una constante. Con *static* le indicamos que podremos utilizar la variable *instance* sin la necesidad de hacer ninguna instancia con el operador *new* de la clase AudioManager.

Existen varias clases en el videojuego que son singleton. A continuación se nombran todas ellas: **AudioManager**, **Assets**, **WorldGameController**, **CounterTime, GamePreferences.**

5.5 Diagramas de clases definitivos

Se ha dividido en varios diagramas para facilitar su lectura, se pueden ver los diagramas en los siguientes apartados de este documento. A continuación se pasan a comentar las clases descritas en ellos.

5.6 Diagramas núcleo videojuego

En el paquete com.threecomet.flingthesheep en el directorio FlingTheSheep-android encontramos las siguientes clases:

MainActivity: Hereda de AndroidAplication que a su vez hereda de Activity, es donde se crea la aplicación, crea una instancia de FlingTheSheepMain.

En el paquete com.threecomet.flingthesheep en el directorio FlingTheSheep encontramos la clase **FlingTheSheepMain** donde se instancia la pantalla de intro del juego dentro del método create.

Siguiendo el esquema estructural de un videojuego descrito en el punto 5.3 de este documento, la ilustración 28 detalla el diagrama de clases que engloba la inicialización del juego y el ciclo de vida.

La **Inicialización** queda recogida en la clase Assets, esta clase es un singleton que instancia otras clases que a su vez guardan todos los recursos del videojuego, por recursos se entiende, texturas para los gráficos, archivos de audio, ficheros con información necesaria para el juego. Una vez inicializado, el hecho de ser un singleton nos permite acceder a estas clases a través de la clase assets des de cualquier punto de la aplicación. En la ilustración 29 se detallan todas estas clases que engloba la clase assets. Esta clase se encuentra el el paquete com.threecomet.flingthesheep.game. A continuación se comenta cada una de ellas con una breve descripción:

LevelDecoration: Enumera todas las texturas de los objetos de decoración del videojuego, flores, hierba, etc. Son objetos que no tienen ninguna interacción con el juego simplemente decoran.

AssetFontsScene2D: Engloba los tipos distintos de fuente que se utilizan en el videojuego, tipo de letra, medida y color.

AssetSound: Enumera todos los efectos de sonido del juego.

AssetMusic: Contiene todas las músicas del juego.

FileHandleHighScores: Inicializa la tabla de puntuaciones local, y la actualiza cada vez que se realiza una partida de juego.

LevelObjects: Engloba todos los objetos que tienen interacción con el videojuego: por interacción se entiende objetos que pueden colisionar con otros o que tienen algún papel importante dentro de videojuego, no son solo objetos de decoración.

Character: Contiene el protagonista del juego, el personaje que lleva el jugador, en esta caso las ovejas.

NPCs: En esta clase quedan englobadas las texturas de los enemigos, en esta caso los lobos.

GUI: En esta clase se engloban todas las texturas utilizadas para la creación de la interfaz de usuario, botones, imágenes.

La clase **Constants** es una clase que declara varias constates estáticas que son accesibles des de cualquier punto de la aplicación y se utilizan tanto en la inicialización como en el ciclo de vida del videojuego. La podemos encontrar en el paquete com.threecomet.flingthesheep.util.

El ciclo de vida del videojuego: entrada, proceso y salida, queda repartido en las siguientes clases

Entrada: Gestión pantalla táctil, se realiza en la clase **WorldGameController** en el método llamado handleInput, a su vez cada personaje(character) y enemigo(NPC) gestiona su propia entrada de eventos a través de los métodos sobre escritos touchDown, fling y tap. Ver ilustración 30 clases **Sheep** y **Wolf**.

Proceso: Este apartado realiza la lógica del videojuego y es responsable de esto el método update de la clase **WorldGameController**. Dentro de este método existe implementada la maquina de estados descrita en el apartado 2.3.1 con todos los estados posibles que el juego puede tener durante su ciclo de vida, los estados son READY,GAME,PAUSE,GAME OVER. Cada personaje y enemigo lleva implementado su propia lógica y maquina de estados dentro de sus clases correspondientes, Ver ilustración 30, clases **Sheep** y **Wolf,** estas máquinas de estados son las descritas en los apartados 2.3.2 y 2.3.3 respectivamente.

Salida: La salida descrita en el apartado 5.3, queda recogida en la clase **GameScreen**, Ver ilustración 31.

Finalización: La finalización, liberación de recursos queda a cargo del método sobre escrito **hide** repartido por las distintas clases de la aplicación.

Otras clases que intervienen en el ciclo de juego recogidas en la ilustración 28 son: **CounterTime**, **GamePreferences**, **AudioManager,** a continuación se detalla la funcionalidad de cada una de ellas:

CounterTime: Esta clase engloba el calculo de tiempo transcurrido desde que empieza el juego hasta que termina, hace la función de contador de tiempo interno.

GamePreferences: Esta clase, hace persistente la configuración escogida por el usuario en las opciones. Es posible configurar el juego con o sin sonido al igual que los efectos de audio, esta clase es la encargada de guardar esta configuración aunque abandonemos la aplicación.

AudioManager: Clase que se encarga tanto de la reproducción de audio como los efectos de sonido, es su responsabilidad interrogar la configuración actual del sonido y efectos para determinar si tiene que realizar la reproducción de audio o efectos, así como también determinar el volumen del sonido o si un audio hay que reproducirlo de forma cíclica. En el listado 5.4.3 de este documento se puede ver el código de esta clase como ejemplo de singleton.

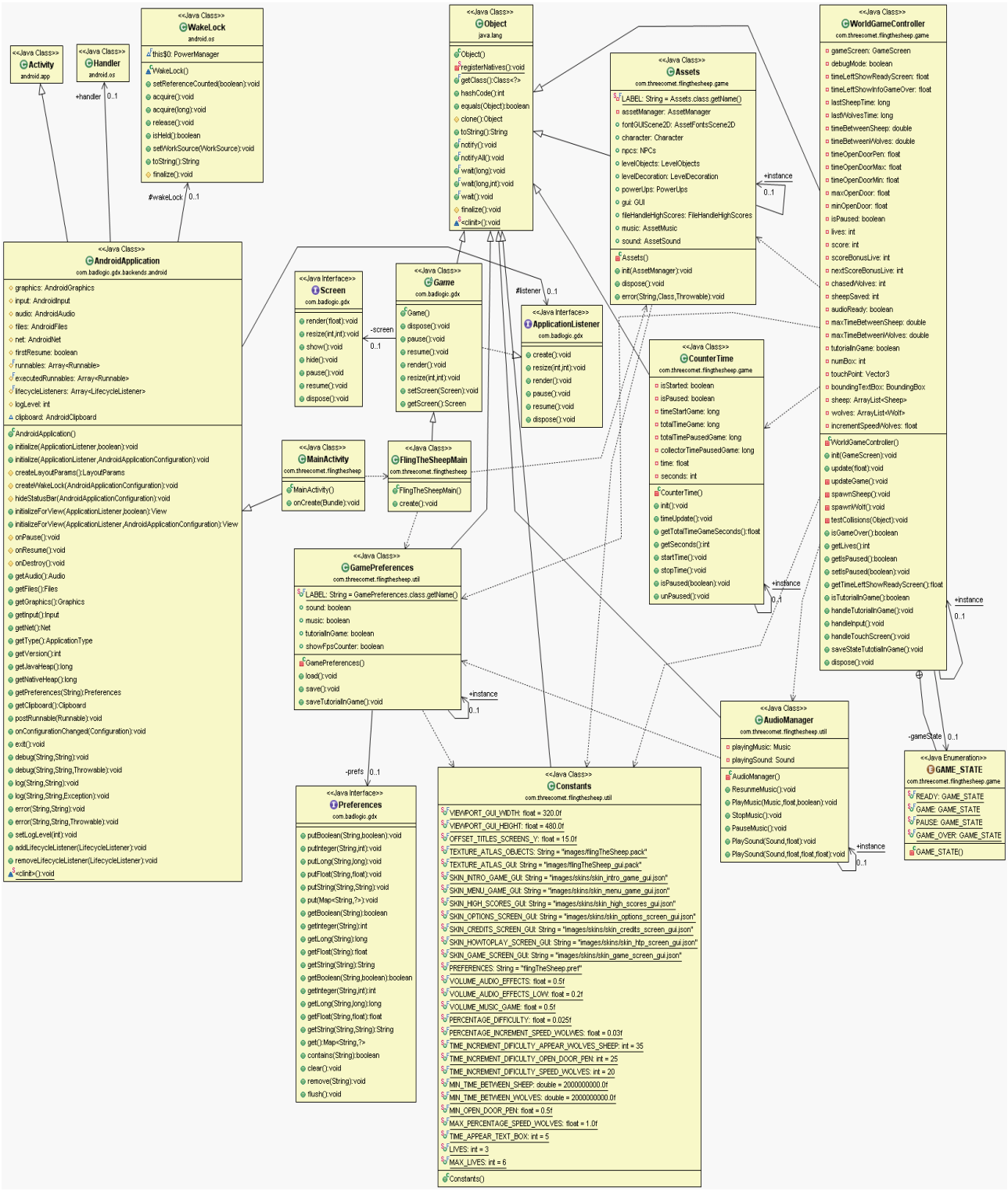


ilustración 28. Diagrama de clases núcleo videjuego

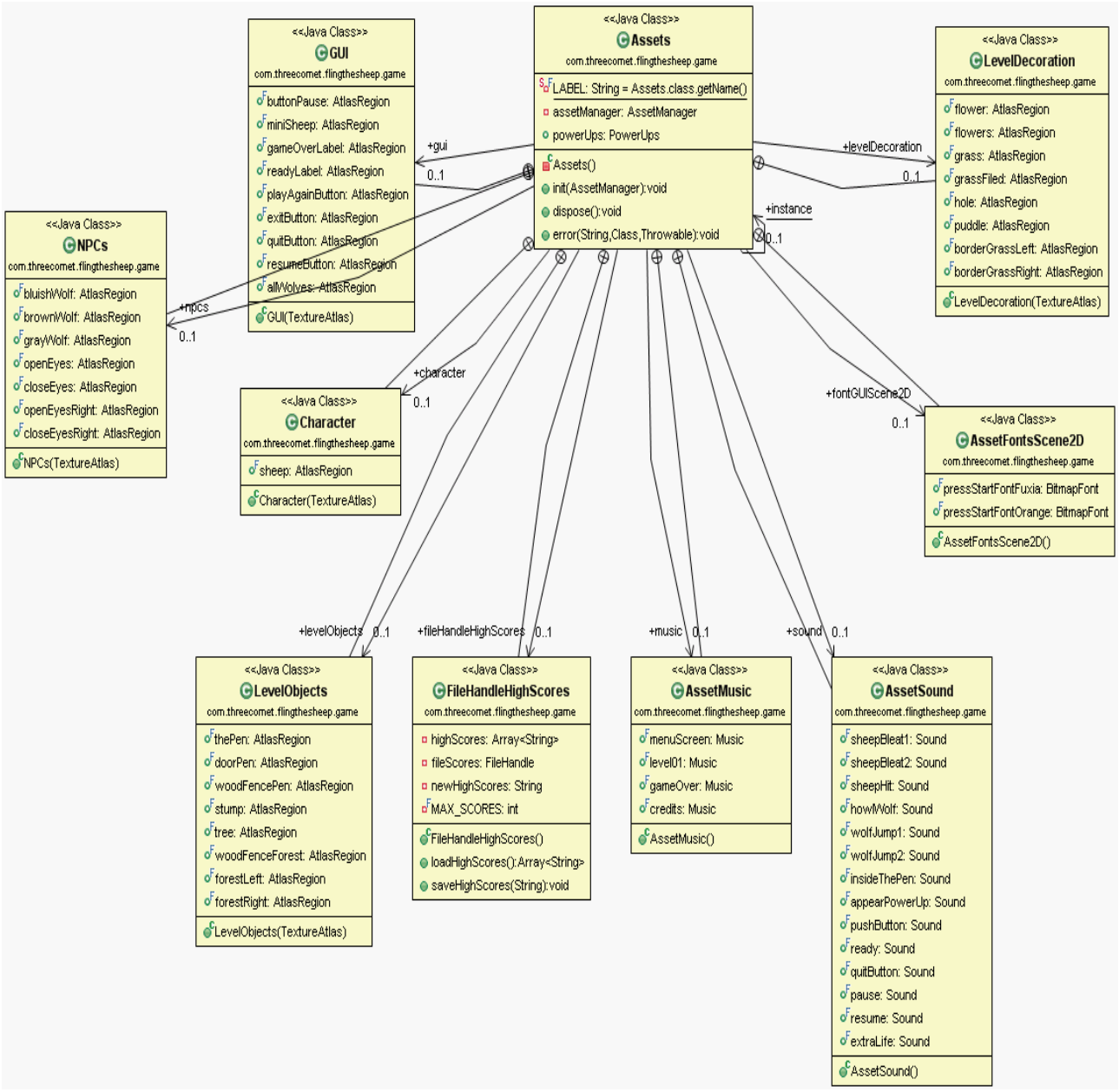


ilustración 29. Diagrama de clases Assets

5.7 Diagrama de clases objetos videojuego

En la ilustración 30 de este documento, se detalla el diagrama de clases referente a los objetos del videojuego. Por objetos se entiende personaje principal, enemigos, juntamente con todos los elementos que forman el conjunto del escenario del videojuego.

En primer lugar cabe destacar la existencia de la clase abstracta **AbstractGameObject**. Esta clase abstracta define los métodos comunes de todos los objetos del juego. Por las necesidades de implementación no se han definido métodos abstractos, no obstante la clase se define como abstracta ya que va a ser la superclase de todos los objetos del videojuego y por tanto no se van a realizar instancias de esta. A continuación se enumeran las clases que se extienden de esta y que recogen todos los objetos del juego, ya sean con o sin interacción en el.

Grass, Hole, Flower, Flowers, Stump, ForestLeft, ForestRight, ThePen, WoodFenceForest, WoodFencePen, DoorPen, BorderGrassLeft, BorderGrassRight, PotionPurple , AubergineRed, Puddle, Tree, GrassField.

Estas clases corresponden a los objetos que constituyen el escenario del videojuego. Las clases **Sheep** y **Wolf** corresponden al personaje del videojuego y a los enemigos respectivamente, estas clases no solamente contienen el método para ser dibujadas en pantalla igual que las anteriores, sino que también poseen toda la lógica para el comportamiento de cada una de ellas, este punto correspondería a la fase de proceso de la estructura básica de un videojuego, detallada en el punto 5.3 de este documento. Existe para cada personaje y enemigo una maquina de estados implementada que indica en que estado se encuentra en cada momento, aplicando la lógica correspondiente para ese estado. Ver las enumeraciones SHEEP_STATE y WOLF_STATE de la ilustración 30, estas corresponden a las máquinas de estado definidas en los apartados 2.3.2 y 2.3.3 respectivamente.

Finalmente, en la ilustración 30 aparecen las clases **GameScreen** y **WorldGameController.**

La clase **GameScreen** como se vera en el apartado 5.8 es la clase que engloba todo lo relacionado con la pantalla del juego o nivel. En ella se instancian todos los objetos descritos anteriormente para la creación del nivel así como también la creación de la GUI del videojuego, botones, textos sobreimpresos y cajas de textos. Es por este motivo que en la clase GameScreen existen dependencias u asociaciones hacia todos los objetos, dependiendo del tipo de declaración de estos.

La clase **WorldGameController** contiene la implementación de gran parte de la lógica del videojuego, esta clase formaría parte de la etapa de proceso dentro de la estructura básica de un videojuego descrita en el apartado 5.3 de este documento.

Existe también en esta clase una asociación hacia la clase **GameScreen**, ya que des de la clase **WorldGameController** que es donde se encuentra gran parte de la lógica, es necesario acceder a la instancia de GameScreen para interactuar con varios elementos visuales que manipulan la lógica del videjuego.

En la siguiente hoja se muestra la ilustración 30 correspondiente a todo los descrito en este apartado.

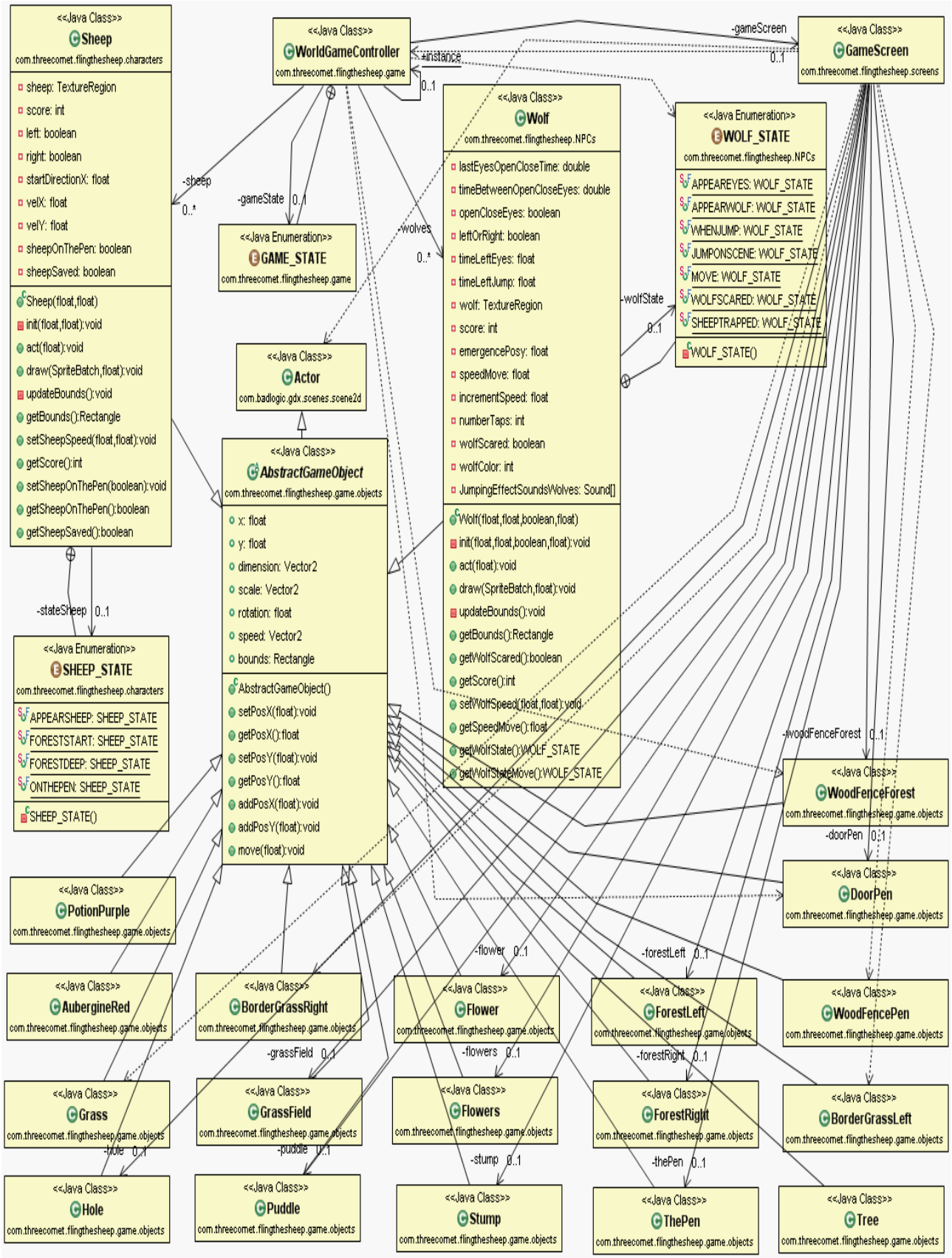


ilustración 30. Diagrama de clases objetos

5.8 Diagrama de clases pantallas videojuego

En la ilustración 31, quedan recogidas todas las clases que intervienen en la creación de pantallas del videjuego, por pantallas se entine la pantalla de intro, la del menú principal, la de opciones etc. Estas clases engloban todo lo necesario para la construcción y visualización de estas. A continuación se detallan cada una de las clases:

AbstractGameScreen: Esta clase abstracta recoge todos los métodos abstractos que son implementados en cada una de las clases que a continuación se detallan(pantallas) con la lógica necesaria según las necesidades de cada pantalla. No se crea ninguna instancia de esta clase.

IntroScreen: Corresponde a la pantalla de intro del videojuego, descrita en el apartado 4.3.3.1

MenuScreen: Corresponde a la pantalla del menú inicial del juego, des de aquí podemos escoger si jugar, consultar la tabla de puntuaciones local, ir a las opciones o a la pantalla de ayuda del videojuego, descrita en el apartado 4.3.3.2

OptionsScreen: Pantalla de opciones, descrita en el apartado 4.3.3.3

CreditsScreen: Pantalla de créditos,descrita en el apartado 4.3.3.4

HighScoresScreen: Pantalla en donde se recoge la puntuación local del videojuego,descrita en el apartado 4.3.3.5

GameScreen: Pantalla general, esta clase engloba toda la lógica del nivel del juego, a su vez engloba la pantalla inicial de nivel READY ?, la pantalla de pausa del juego y la de game over, descritas en los apartados 4.3.3.6, 4.3.3.7, 4.3.3.8, 4.3.3.9

HowToPlayScreen: El videojuego dispone de un tutorial in game que solo aparece la primera vez que jugamos a el y en las siguientes partidas ya no vuelve aparecer más, descrita en el apartado 4.3.3.11. No obstante, existe una pantalla en donde podemos ver este tutorial fuera del juego, esta clase esta destinada a contener los datos de esta pantalla, descrita en el apartado 4.3.3.10

TextGUI: Esta clase contiene todos los textos sobreimpresos que existen en el videojuego. En ella esta la lógica de cada uno de estos textos para que sean posicionados y visualizados desde las distintas partes del videojuego.

TUTORIAL_INGAME_STATE: Esta enumeración enumera los estados de la maquina de estados utilizada para mostrar el tutorial in game dentro del videojuego la primera vez que jugamos, la máquina de estados esta implementada dentro de la clase TextGUI, la máquina corresponde a la descrita en el apartado 2.3.4.

En la siguiente hoja se muestra la ilustración correspondiente a todo los descrito en este apartado.

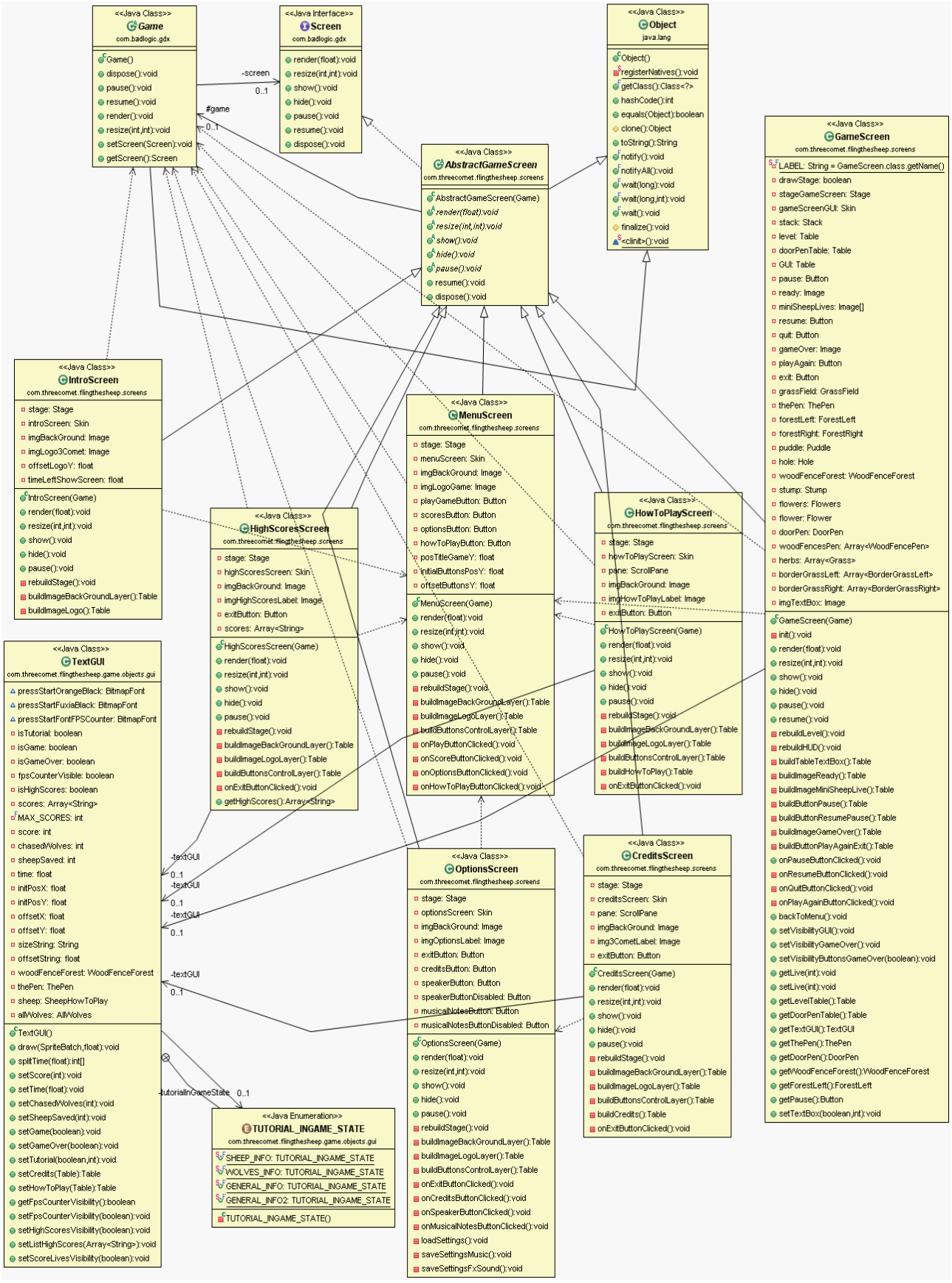


ilustración 31. Diagrama de clases pantallas

5.9 Configurar el entorno de desarrollo

Antes de empezar a desarrollar juegos con libGDX, es necesario instalar y configurar el entorno de desarrollo. Para ello se utilizará Eclipse como entorno de desarrollo integrado IDE. A su vez se creara un proyecto que nos servirá como base para el desarrollo de este.

Es posible desarrollar con otros entornos distintos a Eclipse, pero el hecho que Eclipse sea el entrono de desarrollo soportado oficialmente, ha sido motivo suficiente para escogerlo para este proyecto como entorno de desarrollo.

5.9.1 Instalación

Para poder instalar el entorno de desarrollo nos hará falta primero instalar el Java Development Kit(JDK), para posteriormente poder instalar eclipse.

Una vez instalados, será necesario bajar el framework libGDX del siguiente enlace:

http://libgdx.badlogicgames.com/download.html

También sera necesario instalar el Android SDK, seguidamente tendremos que ejecutar eclipse y instalar los plugins necesarios para poder desarrollar con Android. El plugin necesario para desarrollar con Eclipse sobre Android de llama ADT (Android Developer Tools). Para instalar este plugin debemos dirigirnos a Help-->Install new software tal y como muestra la siguiente ilustración:

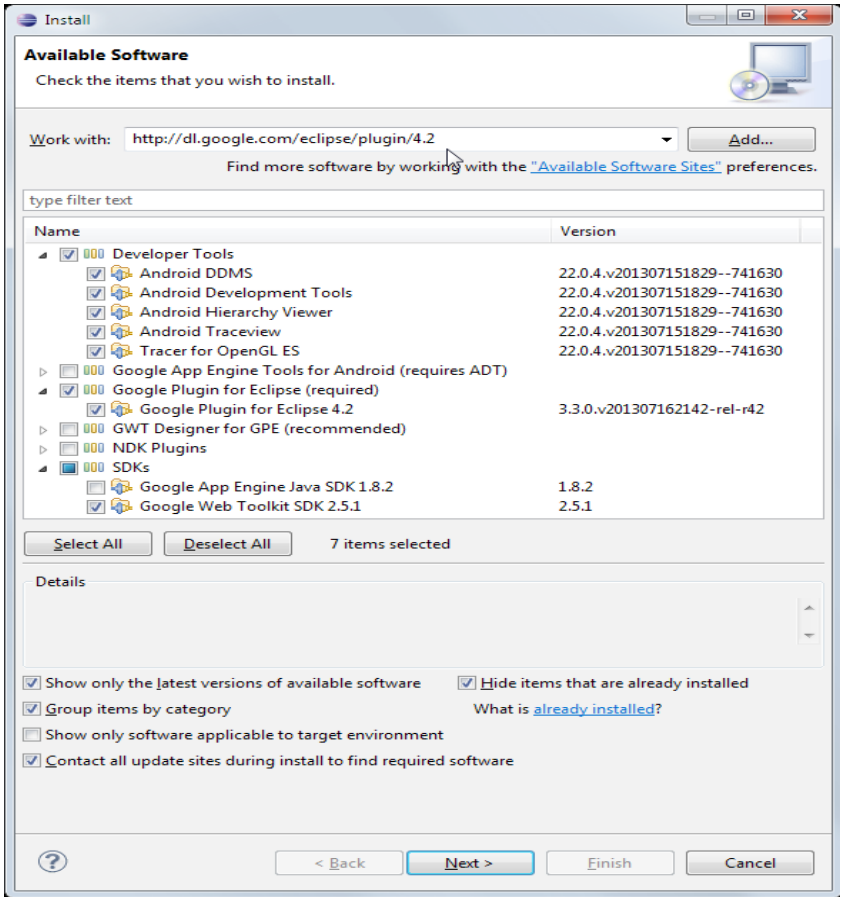


Ilustración 32. Instalación plugin ADT en Eclipse

Una vez el entorno de desarrollo preparado, será el momento de descomprimir el framework libGDX, se recomienda descomprimirlo en la raíz C:\libGDX en esta caso ya que trabajamos bajo Windows.

Una vez descomprimido, dentro de la la carpeta encontramos una aplicación llamada gdx-setup-ui.jar. Esta aplicación nos ayuda a crear un proyecto ya configurado con las librerías necesarias listo para ser importado a Eclipse y empezar a desarrollar con libGDX para Android. O para cualquier otra de las plataformas soportadas por el framework. En este caso se configura el proyecto para Android y para Escritorio, de esta manera podremos verificar durante el desarrollo del proyecto los avances sin necesidad de instalarlo en un dispositivo Android ya sea físico o virtual.

Las siguientes ilustraciones muestran la configuración a realizar.

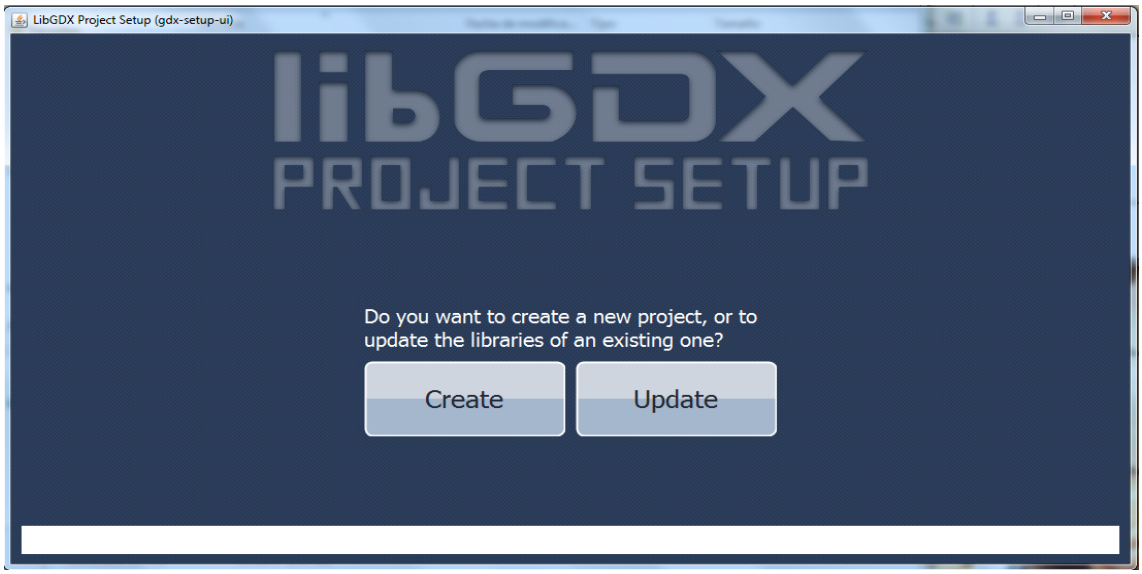


ilustración 33. Creación proyecto libGDX.

Habrá que pulsar en el botón Create, y con ello accederemos a la siguiente ilustración.

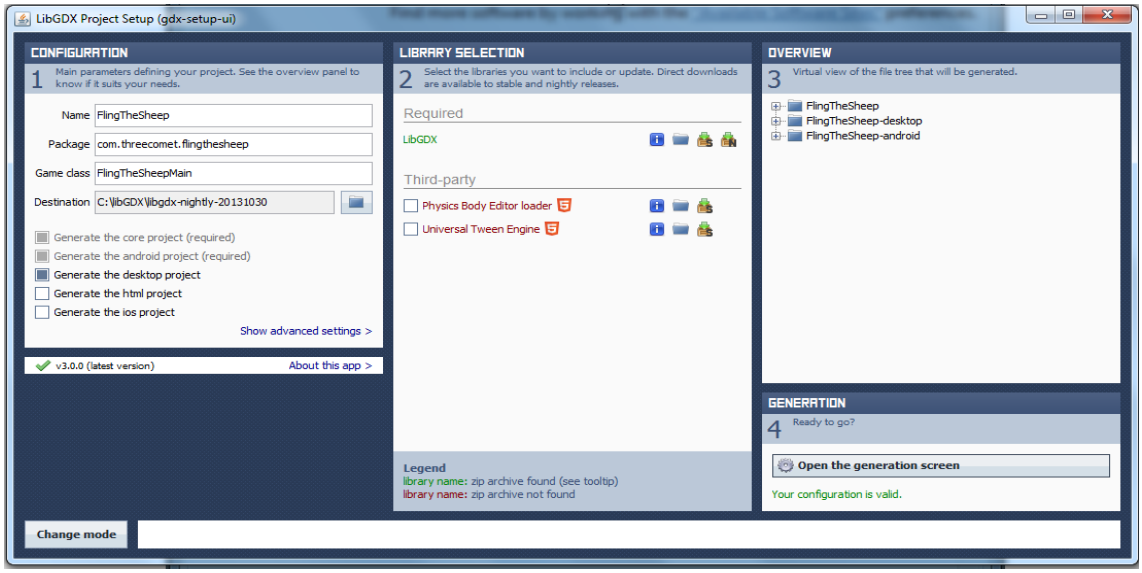


ilustración 34. Creación proyecto libGDX.

Una vez configuradas todas las opciones, le daremos al paso 4 Open the generation screen y generaremos el proyecto. Esto proyecto sera importable a eclipse desde la opción Import del propio IDE.

La configuración del proyecto también es posible hacerla a mano, no obstante esta aplicación nos ahorra tiempo y posibles errores en el momento de configurarlo dentro del entorno, ya que para hacerlo a mano existen distintos pasos a seguir para conseguir lo mismo que se consigue con esta aplicación con escasos clicks de ratón.

5.10 Traspaso del diseño a texturas

Todo el diseño realizado en el apartado 4, será necesario pasarlo a texturas para posteriormente utilizarlo en la implementación del juego como recursos gráficos.

Una textura es una imagen digital cualquiera, en cualquiera de los formatos habituales, jpg, png, gif, tga, etc. Para poderla cargar dentro de un videojuego y manipularla hará falta una librería especializada para ello, esto se podria programar, para para el caso de este proyecto utilizamos el propio framework libGDX para realizar esta tarea que a su vez envuelve el uso de OpenGL.

Las texturas en OpenGL deben ser siempre potencias de 2,16,32,64,128,256 etc. del contrario nos dará problemas y no funcionará.

El primer paso sera desmontar todas las capas pieza por pieza por decirlo de alguna manera y crear ficheros de cada una de las capas de forma separada para guardarlas todas en una carpeta con un nombre identificativo. Ver la siguiente ilustración:

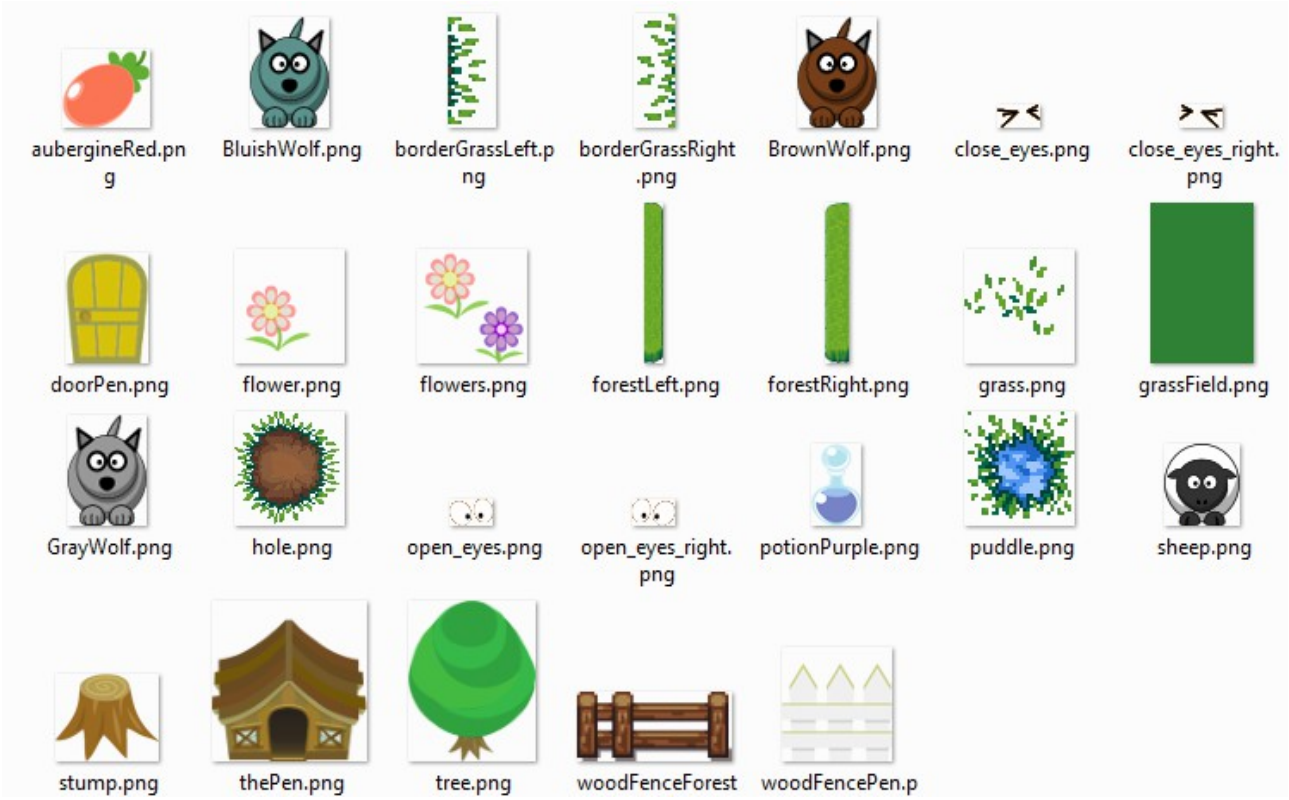


ilustración 35. Capas separadas por imágenes

Una vez realizado este paso, deberemos juntarlas todas de nuevo en una única textura que sea potencia de 2. Tratándose de un juego para móvil, las texturas no deberían ser mayores de 1024x1024 y no se podrán cargar muchas de ellas en memoria por razones obvias de recursos del tipo de dispositivo.

Para juntarlas utilizaremos una aplicación llamada Texture Packer, esta nos juntara las texturas de repartiéndola de forma que quepan en una tamaño de textura definido que en este caso sera de 1024x1024, en caso de caber en una única textura, el programa generara 2 texturas siempre optimizadas a potencias de 2. La siguiente ilustración muestra el programa Texture packer.

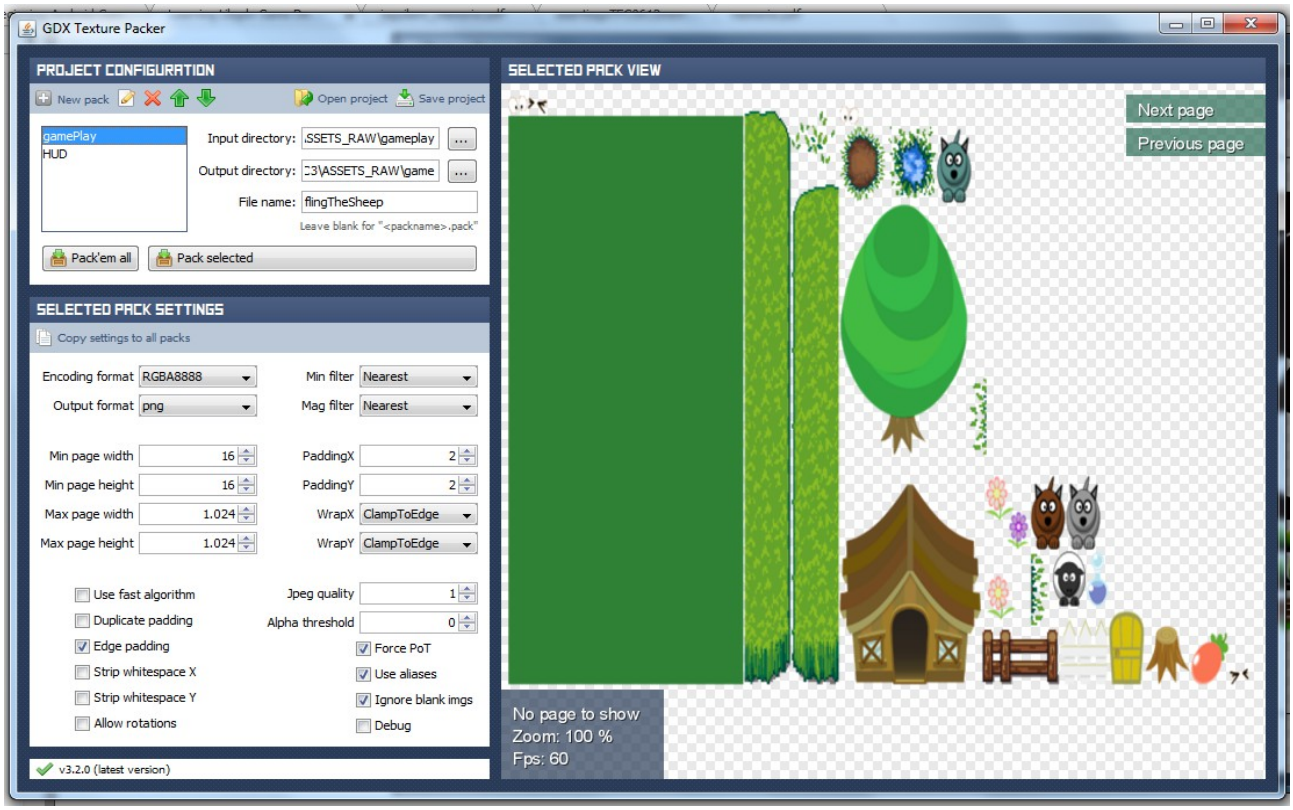


Ilustración 36. Texture packer

El formato de salida de las textura sera png, ya que necesitamos mantener el canal alpha de estas activado para los objetos sean correctamente dibujados en la escena del videojuego.

Conjuntamente con la textura, la aplicación nos generara un archivo de texto que contendrá el nombre de la textura y la posición donde se encuentra dentro de la textura generada de 1024x1024. Ver la siguiente ilustración en la siguiente página.

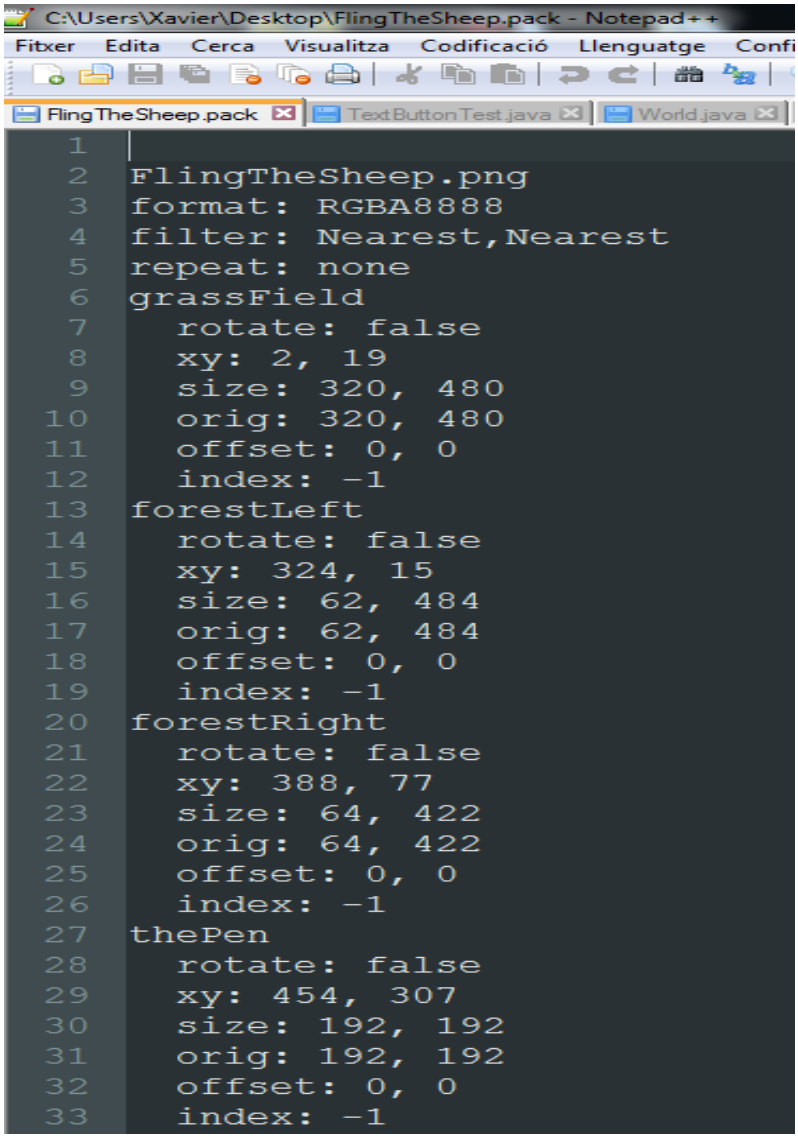


ilustración 37. Posición texturas

De esta manera podremos cargar las texturas en la fase inicialización, en la clase Assets descrita en la ilustración 29 en forma de diagrama de clases.

5.11 Implementación de las máquinas de estado

La implementación de las máquinas de estado se realiza con una estructura enum conjuntamente con una estructura switch case. El siguiente listado muestra la declaración del la estructura enum implementada en la clase WorldGameController, la implementación de todas las máquinas se realiza del mismo modo, por tanto se toma esta como ejemplo en representación de todas las implementadas en el videojuego.

Listado 5.11.1

```
1.      private enum GAME_STATE {READY,GAME,PAUSE,GAME_OVER}
2.      private GAME_STATE gameState;
3.
4.      gameState = GAME_STATE.READY;
```

La siguiente tabla muestra un trozo de la estructura switch case implementada en el método update de la clase WorldGameController.

Listado 5.11.2

```
1.      public void update(float deltaTime)
2.      {
3.        switch(gameState)
4.        {
5.         case READY:
6.             if(audioReady) {
7.                 AudioManager.instance.PlaySound(Assets.instance.sound.ready,
8.               Constants.VOLUME_AUDIO_EFFECTS);
9.                 audioReady = false;
10.            }
11.            if(timeLeftShowReadyScreen > 0)
12.            {
13.              if(deltaTime < 0.3f ) {
14.                timeLeftShowReadyScreen -=deltaTime;
15.              }
16.              if(timeLeftShowReadyScreen < 0)
17.              {
18.                gameScreen.setVisibilityGUI();
19.                timeLeftShowReadyScreen = 0;
20.                CounterTime.instance.startTime();
21.
22.                AudioManager.instance.PlayMusic(Assets.instance.music.level01,
23.                   Constants.VOLUME_MUSIC_GAME, true);
24.                gameState = GAME_STATE.GAME;
25.
26.              }
27.           }
28.           break;
29.        case GAME:
30.            updateGame();
31.            if(timeOpenDoorPen > 0)
32.            {
33.                if(deltaTime < 0.3f ) {
34.                timeOpenDoorPen -= deltaTime;
35.            }
36.            if(timeOpenDoorPen <= 0) {
37.               gameScreen.getDoorPen().setVisibility(!gameScreen.getDoorPen().getVisibility());
38.             .........
```

En este listado podemos ver el estado READY y un trozo del estado GAME de la máquina de estados definida en el apartado 2.3.1 de este documento, inicialmente la máquina toma el estado READY como se puede ver en la linea 4 del listado 5.11.1, este estado se ira repitiendo a cada vuelta del ciclo de vida del videojuego hasta que cambie el estado de juego a GAME, esto sucederá una vez transcurridos 3 segundos tal y como se puede ver en la linea 24 del listado 5.11.2. Hay que tener en cuenta que un videojuego, tal y como se ha definido en el apartado 5.2.2 de este documento, forma un bucle infinito, por tanto la función update de la linea 1 de este listado se repetirá de forma indefinida durante la ejecución del nivel de juego y cada vuelta ejecutará el estado en que se encuentre la máquina de estados.

6.Conclusiones

El desarrollo del proyecto ha sido tal y como se esperaba en lineas generales, no han habido grandes desviaciones de tiempo ni durante el diseño ni en el posterior desarrollo de este. Cabe destacar que el hecho de poder probar el proyecto ejecutándolo en la propia máquina donde se ha desarrollado, sin tener que utilizar un dispositivo real o virtual cada vez que se ha querido verificar, ha hecho que el desarrollo sea mucho más fluido que sino se hubiera contado con el escenario descrito. Esto no significa que en ciertos momentos del desarrollo no se haya probado con un dispositivo real Android. Dejando a parte la gran ventaja que esto significa ya que una vez desarrollado, el esfuerzo realizado nos sirve para distintas plataformas casi sin tener que tocar nada del código.

Por tanto, el hecho de haber escogido libGDX para el desarrollo ha sido una buena elección. Es también una buena elección escoger un framework o un motor focalizado al desarrollo de videojuegos, si se quiere implementar uno ya sea para Android o para cualquier plataforma.

Existe la posibilidad de desarrollarlo sin ningún tipo de framework o motor, pero hay que tener claro que se quiere hacer exactamente y a donde se quiere llegar, es decir si lo que se quiere es centrarse en la programación de videojuegos, es mucho mejor partir de un framework o motor ya que de esta forma nos desligamos de programar ciertos apartados que están más focalizados a la programación de frameworks o motores para videojuegos que no a la programación de videojuegos en si.

Yendo más allá de donde el programador se quiere centrar, existe otro factor importante a tener en cuenta en el momento de plantear un proyecto, llamado tiempo, con el tiempo de que se ha dispuesto para este proyecto, queda aún más justificado la elección de este framework para realizarlo. Ya que probablemente sin el, debido al esfuerzo mayor en programar apartados que no son exclusivos del videojuego, pudieran haber existido desviaciones de tiempo importantes en relación al calendario propuesto.

Ha quedado fuera del proyecto por temas de tiempo realizar animaciones a todos los personajes del videojuego para conseguir una mejor experiencia de usuario y un nivel de acabado superior. Fuera del proyecto también ha quedado la implementación no solo de unas puntuaciones máximas a nivel local sino también unas a nivel mundial consultables a través de una página web. A su vez no solo guardaría la puntuación máxima sino también el tiempo jugado, las ovejas salvadas y los lobos ahuyentados. También ha quedado fuera de este proyecto la realización de un blog dedicado al videojuego. Todos estos apartados se implementaran en breve fuera de este TFC.

7.Recursos utilizados

7.1 Portada documento

Jake Maymar
http://www.flickr.com/photos/maymar/5334789030/
http://creativecommons.org/licenses/by-sa/2.0/deed.en

7.2 Recursos utilizados para el diseño del videojuego

Para realizar todo el diseño visual del videojuego se ha recurrido a diferentes fuentes de arte

sujetas a licencias creative commons y dominio público. A continuación se detalla una lista de cada recurso, con su autor en el caso de que no haya sido publicado de forma anónima, juntamente con el link donde se ha encontrado.

7.2.1 Fotografía pantalla intro videojuego y usuarios

Michael Karrer
http://www.flickr.com/photos/michael_karrer/78438470/
http://creativecommons.org/licenses/by-nc/2.0/deed.en

Jodi Womack
http://www.flickr.com/photos/jodiwomack/6217578512/
http://creativecommons.org/licenses/by/2.0/deed.en

Tony Alter
http://www.flickr.com/photos/78428166@N00/8279647916/
http://creativecommons.org/licenses/by/2.0/deed.en

Gage Skidmore (Nina Dobrev)
http://www.flickr.com/photos/gageskidmore/5980906910/
http://creativecommons.org/licenses/by-sa/2.0/deed.en

7.2.2 Iconos y botones

7.2.2.1 Altavoz

Autor: http://www.iconshock.com
Link: https://www.iconfinder.com/icons/67771/audio_speaker_icon#size=128
Licencia: Free for commercial use

7.2.2.2 Nota Musical

Autor: http://www.dezinerfolio.com/
Link: https://www.iconfinder.com/icons/58679/music_icon#size=128
Licencia: Free for commercial use

7.2.2.3 Flecha salida

Autor:http://taytel.deviantart.com/
Link:https://www.iconfinder.com/icons/41696/arrow_back_grey_left_orange_icon#size=128
Licencia:Free for commercial use (Include link to authors website)

7.2.2.4 Icono de pausa

Autor:http://www.webdesignerdepot.com/
Link: https://www.iconfinder.com/icons/40716/blue_button_pause_icon#size=64
Licencia: Free for commercial use

7.2.2.5 Icono interrogante (help)

Autor:http://www.doublejdesign.co.uk/
Link:http://www.iconarchive.com/show/ravenna-3d-icons-by-double-j-design/Help-icon.html

Licencia: http://creativecommons.org/licenses/by/3.0/

7.2.2.6 Botones (Play, Score, Options.....)

Autor:yd
Link:http://opengameart.org/content/buttons-variations
Licencia:http://creativecommons.org/publicdomain/zero/1.0/

7.2.3 Recursos gráficos 2D componentes escenario

7.2.3.1 Ojos lobos

Autor:Creek23
Link: http://opengameart.org/content/gnu-the-gnu-mascot
Licencia: http://creativecommons.org/licenses/by-sa/3.0/

7.2.3.2 Ovejas y lobos

Autor: lemmling
Link: http://openclipart.org/user-cliparts/lemmling
Licencia: http://openclipart.org/share

7.2.3.3 Fondos de pantallas OPTIONS, HIGH SCORES, HOW TO PLAY

Autor:Sam
Link: http://opengameart.org/content/country-field
Licencia: http://creativecommons.org/publicdomain/zero/1.0/

7.2.3.4 Escenario

Daniel Eddeland
Link:http://opengameart.org/content/lpc-farming-tilesets-magic-animations-and-ui-elements
Licencia:http://creativecommons.org/licenses/by-sa/3.0/

7.2.4 Fuentes

7.2.4.1 Thinckhead
Autor: Scotty Ulrich
Link: http://www.dafont.com/es/scotty-ulrich.d101?l[]=10
Licencia: http://creativecommons.org/publicdomain/zero/1.0/

7.2.4.2 Misfortune
Autor: Scotty Ulrich
Link: http://www.dafont.com/es/scotty-ulrich.d101?l[]=10
Licencia: http://creativecommons.org/publicdomain/zero/1.0/

7.2.4.3 Tabaquera
Autor: deFharo
Link: http://www.dafont.com/es/fernando-haro.d3419?l[]=10
Licencia: http://creativecommons.org/publicdomain/zero/1.0/

7.2.4.4 Press Start Font

Autor: desconegut
Link: http://www.zone38.net/font/#pressstart
Licencia: http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL

7.2.5 Otros

Todos los demás recursos no nombrados en esta lista, como el fondo de pantalla del menú principal de inicio del juego y las letras del titulo del juego "Fling the sheep", impresas sobre la pantalla del menú inicial, son de elaboración propia.

7.2.6 Audio y efectos de sonido

7.2.6.1 Menu

Autor: mrpoly
Link:http://opengameart.org/content/menu-music
Licencia: http://creativecommons.org/publicdomain/zero/1.0/

7.2.6.2 Game Over

Autor: Macro
Link:http://opengameart.org/content/rest-outro-loop
Licencia:http://creativecommons.org/licenses/by/3.0/

7.2.6.3 Game Play

Autor:Rezoner
Link:http://opengameart.org/content/trance-menu
Licencia:http://creativecommons.org/licenses/by/3.0/

7.2.6.4 Crédits

Autor: -
Link:http://opengameart.org/content/1-minute
Licencia:http://creativecommons.org/publicdomain/zero/1.0/

7.2.6.5 Ready

http://opengameart.org/content/ui-accept-or-forward
http://creativecommons.org/licenses/by/3.0/

7.2.6.6 Pause_Resume

http://opengameart.org/content/completion-sound
http://creativecommons.org/licenses/by/3.0/

7.2.6.7 Quit

http://opengameart.org/content/fly-swatter-squish-sound
http://creativecommons.org/publicdomain/zero/1.0/

7.2.6.8 Sheep

http://opengameart.org/content/sheep-sound-bleats-yo-frankie
http://creativecommons.org/licenses/by/3.0/

El resto de efectos de audio han sido obtenidos de la siguiente página: http://www.pond5.com/

8.Lineas abiertas del proyecto

Se hará un repaso del proyecto por si existe algún bug para solucionarlo. Posteriormente de adaptara para que sea funcional en distintas resoluciones de pantalla. Se añadirán los power-ups que se definieron al principio del proyecto, por temas de tiempo han quedado fuera de la entrega. A su vez, se balanceara el gameplay, añadiendo alguna mecánica de juego extra si procede.

Se publicara el juego en Google Play y se creara un blog del juego, se mirara la posibilidad de hacer funcionar el juego en el blog con webGL. http://projects.3comet.com/

Se ampliara la tabla de puntuaciones de la siguiente manera: la tabla no solamente guardara los puntos, también guardara el tiempo jugado, número de lobos ahuyentados, número de ovejas salvadas y nombre del usuario con 5 caracteres. Se creara un un registro de puntuaciones global, consultable desde el blog del videojuego.

Finalmente se crearan más niveles de juego, con distribuciones diferentes, algún nivel podría también tener mecánicas diferentes de juego con los mismos personajes…….....

9.Bibliografía utilizada

Titulo	Beginning Android Games (Second Edition)	**ISBN**	978-1-43024-677-0
Autor	Mario Zechner / Robert Green		
Editorial	Apress		
Contenido	Desarrollo de videojuegos para Android		

Titulo	Learning libGDX Game Development	**ISBN**	978-1-78216-604-7
Autor	Andreas Oehlke		
Editorial	PACKT Publishing		
Contenido	Desarrollo de videojuegos para Android con libGDX		

10.Otros proyectos realizados

10.1 Remake Gradius Konami

Remake del famoso shooter de Konami lanzado para ordenadores MSX en 1986, programado con C++ con SDL. Aunque este juego existe para multitud de plataformas. Ver articulo wikipedia:

http://es.wikipedia.org/wiki/Gradius_videojuego

Blog del proyecto: http://projects.3comet.com

10.2 Tales of Nimria mini RPG

Mini RPG en 2D, programado en C++ con DirectX, este mini RPG trata de salvar a una princesa capturada por unas malvadas criaturas.

Blog del proyecto: http://projects.3comet.com

10.2 Motocalipsis Bar 66 RTS

RTS en 3D con ambiente motero, los hechos de la historia suceden en un pequeño pueblo de Texas. Programado en Unity con C#.

Blog del proyecto: http://motocalipsisbar66.blogspot.com.es/

Creative Commons 4.0 license by-nc-sa

Attribution-NonCommercial-ShareAlike 4.0 International

==========

Creative Commons Corporation ("Creative Commons") is not a law firm and does not provide legal services or legal advice. Distribution of Creative Commons public licenses does not create a lawyer-client or other relationship. Creative Commons makes its licenses and related information available on an "as-is" basis. Creative Commons gives no warranties regarding its licenses, any material licensed under their terms and conditions, or any related information. Creative Commons disclaims all liability for damages resulting from their use to the fullest extent possible.

Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

> *Considerations for licensors: Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright. More considerations for licensors:*
> *wiki.creativecommons.org/Considerations_for_licensors*
>
> *Considerations for the public: By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor's permission is not necessary for any reason--for example, because of any applicable exception or limitation to copyright--then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of*

the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable. More_considerations for the public:
wiki.creativecommons.org/Considerations_for_licensees

==========

Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 -- Definitions.

a. Adapted Material means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. Adapter's License means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. BY-NC-SA Compatible License means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

d. Copyright and Similar Rights means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database

Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

e. Effective Technological Measures means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

f. Exceptions and Limitations means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

g. License Elements means the license attributes listed in the name of a Creative Commons Public License. The License Elements of this Public License are Attribution, NonCommercial, and ShareAlike.

h. Licensed Material means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

i. Licensed Rights means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

j. Licensor means the individual(s) or entity(ies) granting rights under this Public License.

k. NonCommercial means not primarily intended for or directed towards commercial advantage or monetary compensation. For purposes of this Public License, the exchange of the Licensed Material for other material subject to Copyright and Similar Rights by digital file-sharing or similar means is NonCommercial provided there is no payment of monetary compensation in connection with the exchange.

l. Share means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

m. Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

n. You means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning.

Section 2 -- Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

a. reproduce and Share the Licensed Material, in whole or in part, for NonCommercial purposes only; and

b. produce, reproduce, and Share Adapted Material for NonCommercial purposes only.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a) (4) never produces Adapted Material.

5. Downstream recipients.

a. Offer from the Licensor -- Licensed Material. Every recipient of the Licensed Material automatically

receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

b. Additional offer from the Licensor -- Adapted Material. Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.

c. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties, including when the Licensed Material is used other than for NonCommercial purposes.

Section 3 -- License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

a. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

b. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

c. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

b. ShareAlike.

In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the following conditions also apply.

1. *The Adapter's License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-NC-SA Compatible License.*

2. *You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. You may satisfy this condition in any reasonable manner based on the medium, means, and context in which You Share Adapted Material.*

3. *You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, Adapted Material that restrict exercise of the rights granted under the Adapter's License You apply.*

Section 4 -- Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. *for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database for NonCommercial purposes only;*

b. *if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material, including for purposes of Section 3(b); and*

c. *You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.*

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 -- Disclaimer of Warranties and Limitation of Liability.

a. UNLESS OTHERWISE SEPARATELY UNDERTAKEN BY THE LICENSOR, TO THE EXTENT POSSIBLE, THE LICENSOR OFFERS THE LICENSED MATERIAL AS-IS AND AS-AVAILABLE, AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE LICENSED MATERIAL, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHER. THIS INCLUDES, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OR ABSENCE OF ERRORS, WHETHER OR NOT KNOWN OR DISCOVERABLE. WHERE DISCLAIMERS OF WARRANTIES ARE NOT ALLOWED IN FULL OR IN PART, THIS DISCLAIMER MAY NOT APPLY TO YOU.

b. TO THE EXTENT POSSIBLE, IN NO EVENT WILL THE LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE) OR OTHERWISE FOR ANY DIRECT, SPECIAL, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, EXEMPLARY, OR OTHER LOSSES, COSTS, EXPENSES, OR DAMAGES ARISING OUT OF THIS PUBLIC LICENSE OR USE OF THE LICENSED MATERIAL, EVEN IF THE LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES, COSTS, EXPENSES, OR DAMAGES. WHERE A LIMITATION OF LIABILITY IS NOT ALLOWED IN FULL OR IN PART, THIS LIMITATION MAY NOT APPLY TO YOU.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 -- Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 -- Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 -- Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

==========