# GHOST HUNTER

Álvaro Fernández-Alonso Araluce

Javier Abella Gea

# Contenido

# Resumen

Con la creación de esta práctica queremos realizar un juego destinado a móviles bajo sistema operativo Android, integrando técnicas de realidad aumentada en la funcionalidad del juego.

El juego consiste a groso modo en una serie de niveles en los que el jugador se encontrará una serie de pequeños fantasmas los cuales deberá tocar para eliminarlos y poder avanzar por los niveles.

La principal innovación que presenta el juego es la presentación del juego, desarrollándose este en un escenario de realidad aumentada aprovechándose de la posición y la cámara del Smartphone del jugador

# Objetivos.

- Crear un juego con varios niveles.
- Integración de técnicas de realidad aumentada.
- Plataforma Android.
- Funcionalidades táctiles.
- Intuitivo.

# Herramientas.

A continuación expondremos las herramientas utilizadas en el desarrollo de la práctica.

## Android Studio

Android Studio es un entorno de desarrollo integrado (IDE), que proporciona varias **mejoras con respecto al plugin ADT** (Android Developer Tools) para Eclipse. Android Studio utiliza una licencia de software libre Apache 2.0, está programado en Java y es multiplataforma.

Principales características:

- **Soporte** para programar aplicaciones para **Android Wear**.
- **Herramientas Lint** (detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador) para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.

- Utiliza **ProGuard** para optimizar y reducir el código del proyecto al exportar a APK (muy útil para dispositivos de gama baja con limitaciones de memoria interna).
- Integración de la herramienta **Gradle** encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de testing, compilación o empaquetado.
- **Nuevo diseño del editor** con soporte para la edición de temas.
- **Nueva interfaz** específica para el desarrollo en Android.
- Permite la **importación de proyectos** realizados en el entorno **Eclipse**, que a diferencia de Android Studio (Gradle) utiliza **ANT**.
- Posibilita el **control de versiones** accediendo a un repositorio desde el que poder descargar Mercurial, Git, Github o Subversion.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- **Vista previa** en diferentes dispositivos y resoluciones.
- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo xml.

Requisitos sobre W□indows:

- Microsoft Windows 8/7/vista (32 o 64 bits).
- 2 GB de RAM mínimo, recomendados 4 GB de RAM.
- 400 MB de espacio en disco.
- Resolución mínima de pantalla 1280x800.
- Java Developement Kit (JDK) 7 o superior.

## Metaio SDK

Metaio SDK es un kit que proporciona todos los métodos necesarios para crear rápidamente una aplicación de realidad aumentada para un dispositivo móvil.

La interfaz principal es metaio::IMetaioSDK, otra interfaz utilizada es metaio::IGeometry, que recoge todas las funcionalidades directamente relacionadas con la geometrías.

**Requisitos**.

Los requisitos mínimos y adicionales de hardware y software para ejecutar Metaio SDK en la plataforma Android se recogen a continuación.

**Hardware**.

- CPU que soporte el x86
- OpenGL ES 2.x

- Cámara con una resolución de 230x240

**Software**

- Androdi 2.3.3 o superior.

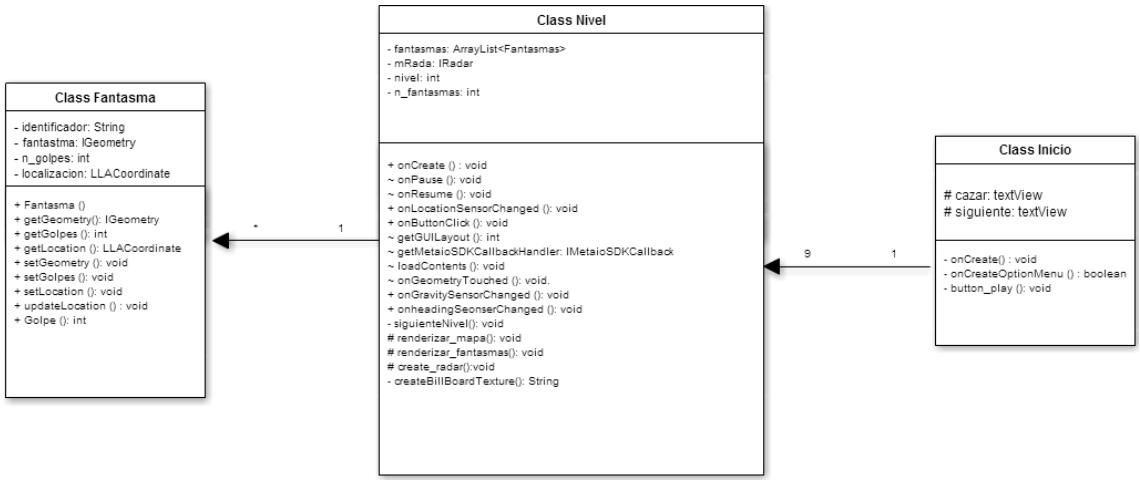
## Comparativa entre Android Studio y Eclipse.

Mostramos en la siguiente tabla a modo de comparativa entre las herramientas de desarrollo Android Studio y ADT Eclipse.

| Características | Android Studio | ADT Eclipse |
|---|---|---|
| Sistema de construcción | Gradle | ANT |
| Refactorización y completado avanzado de código Android | Si | No |
| Diseño del editor gráfico | Si | Si |
| Vista en tiempo real de renderizado de layouts | Si | No |
| Nuevos módulos en proyecto | Si | No |
| Editor de navegación | Si | No |
| Visualización de recursos desde editor de código | Si | No |

# Diseño arquitectónico.

Para el diseño arquitectónico del sistema se ha utilizado metodología UML, en concreto diagramas de clases.




**Class Inicio.**

Será la encargada de iniciar el juego y presentar un menú al usuario, de manera que, a través de una serie de botones, el usuario podrá realizar una serie de acciones, entre la que destaca lanzar el juego en sí

**Class Nivel.**

Será la encargada de lanzar cada una de los niveles que encontraremos a lo largo del juego. La diferencia entre los niveles será el cambio de dificultad que se produce al aumentar el número de fantasmas a los que nos enfrentamos y la posición en la que se encuentran.

También se encarga de mostrar la información del siguiente nivel al terminar el actual.

**Class Fantasma.**

Será la encargada de crear los atributos que darán forma a los fantasmas involucrados en el juego.

## Funcionalidades destacadas.

Para cada una de las clases vamos a comentar las funciones destacadas que podemos encontrar en cada una de ellas:

**Class Inicio.**

- *public boolean onCreateOptionsMenu(Menu menu);*
  Crea el menú de opciones del juego.

- *public void button_play (View v)*
  Lanzará el primer nivel del juego.

**Class Nivel.**

- *protected void loadContents()*
  Renderiza en contenido del nivel.

- *protected void onGeometryTouched(final IGeometry geometry)*
  Se utiliza para contar los golpes que tienen los fantasmas, cuando estos llegan a 0 el fantasmas desaparece del juego, eliminado su geometría y su posición en el radar.

- *private void siguienteNivel()*

Cuando todos los fantasmas han sido eliminados, creamos un cuadro de diálogo para mostrar la información del siguiente nivel al usuario.

- void renderizar_fantasmas()
  Borra todos los fantasmas anteriores y después creará un array con un número de fantasmas específicos para cada nivel y el número de golpes que es necesario para eliminarlos.

- void create_radar()
  Crea el radar que se utilizará para la identificación de la posición de los fantasmas dentro del juego.

- private String createBillboardTexture(String billBoardTitle, String path)
  Se encargará de crear las geometrías que se mostrarán al usuario.

**Class fantasma.**

- public void updateLocation(LLACoordinate locat)
  Será la encargada de actualizar la posición de cada uno de los fantasmas dentro de los niveles generados.

- public int Golpe()
  Disminuirá y retornará el número de golpes necesario para eliminar un fantasma.

## Comunicación.

En el siguiente diagrama podemos ver la comunicación que se produce sobre las interfaces que integran el juego.


## Conclusiones.

Con el presente documento se ha intentado aclarar las posibles dudas que hayan surgido sobre la creación de la aplicación para Android “Ghost Hunter” a la vez que se muestra el diseño y la navegabilidad interna de la aplicación.