TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grado en Ingeniería Informática de Gestión y Sistemas de información

Juego de ARPG para Android

Memoria

Georgy Belyakov

Tutor: Catalina Juan Nadal

Curso 2021-2022

TecnoCampus
Mataró-Maresme

Abstract

In this work, intend to create a complete 2D game, in ARPG, for the Android operating system, with an optimal backend adapted to easily introduce new functional and visual elements. It consist of a client application to install on mobile devices.

Resum

En aquest treball, pretén crear un joc 2D complet, a ARPG, per al sistema operatiu Android, amb un backend òptim adaptat per introduir fàcilment nous elements funcionals i visuals. Consta duna aplicació client per instal·lar en dispositius mòbils.

Resumen

En este trabajo, pretende crear un juego 2D completo, en ARPG, para el sistema operativo Android, con un backend óptimo adaptado para introducir fácilmente nuevos elementos funcionales y visuales. Consta de una aplicación cliente para instalar en dispositivos móviles.

Índice

Índice de figuras

Índice de tablas

Glosario de términos

ARPG - (del inglés Action Role-Playing Games) son un tipo de videojuegos del género RPG que se caracterizan por ofrecer combates en tiempo real. Estos juegos ofrecen un sistema de combate similar a los de tipo hack and slash o a los de tipo shooter.[1]

Shooter - es un género que engloba un amplio número de subgéneros que tienen la característica común de permitir controlar un personaje que, por norma general, dispone de un arma (mayoritariamente de fuego) que puede ser disparada a voluntad. Pertenecen al género de acción.[2]

Roguelike - es un subgénero de los videojuegos de rol que se caracterizan por una aventura a través de laberintos, a través de niveles generados por procedimientos al azar, videojuegos basados en turnos, gráficos basados en fichas y la muerte permanente del personaje del jugador.[3]

Sprite - es un tipo de mapa de bits dibujados en la pantalla de ordenador. Generalmente son utilizados para producir una animación, como un personaje corriendo, alguna expresión facial o un movimiento corporal.[4]

1. Introducción

El uso de dispositivos móviles, que incluyen una gran cantidad de funciones que tienen las computadoras, actualmente está muy extendido entre la población. Hoy, la tasa de penetración del sistema operativo móvil de Google, Android, supera el 68% y es el líder indiscutible del mercado a nivel mundial, superando significativamente a todos sus competidores.

Este TFG trata de crear desde cero un videojuego aplicando tecnologías y metodologías modernas, utilizando los conocimientos y habilidades adquiridos durante los cursos académicos.

En este proyecto se crea una aplicación para móviles que trata de ser una versión simple y original de un videojuego para tabletas y smartphones con el sistema operativo Android, que permite a los usuarios acceder en cualquier momento a un juego de acción-aventura con elementos roguelike y shooter de arriba hacia abajo con gráficos de píxeles y controles sencillos, todo ello con la finalidad de que un mayor número de personas pueda encontrar entretenimiento y relajación a su gusto.

La pantalla de inicio ofrece acceso a una selección de diferentes idiomas, solo se proporcionan dos idiomas en esta etapa de desarrollo. Inglés como idioma predeterminado y español como idioma adicional. El idioma seleccionado se guarda y se selecciona automáticamente incluso después de cerrar y volver a cargar la aplicación. La información sobre los resultados actuales del jugador, la configuración de elementos visuales, la limpieza de la base de datos, el manual del juego, el inicio de una nueva partida y la descarga de la última partida se proporcionan después de elegir el idioma.

El juego en sí proporciona al usuario el control mediante dos joysticks sobre un personaje, cuya tarea es destruir a todos los oponentes en el mapa y recorrer el máximo número de mapas. Las mapa de juego se llenan de forma completamente aleatoria, por lo que cada mapa es única. El jugador tiene tres habilidades únicas. Hay doce tipos de oponentes en el juego, cuatro de los cuales son únicos, así como cuatro tipos únicos de bonificaciones que afectan al personaje del juego.

2. Estudio previo: contexto, antecedentes y necesidades de información

2.1. Sistema operativo

El campo de los juegos móviles es una industria relativamente joven que se está desarrollando activamente. Durante la última década, los juegos móviles se han vuelto más populares que nunca. Además, gracias a la tienda de aplicaciones de Google, se puede publicar fácilmente una aplicación terminada para su posterior venta.

La razón principal por la que la aplicación se desarrolla específicamente para el sistema operativo Android es que en este momento este sistema operativo es el más extendido en el mundo y cuenta con el 68.89% del mercado mundial.

Android 68.89%, **iOS** 30.39%, **Samsung** 0.41%, **KaiOS** 0.13%, **Unknown** 0.11%, **Nokia Unknown** 0.02%; [5]

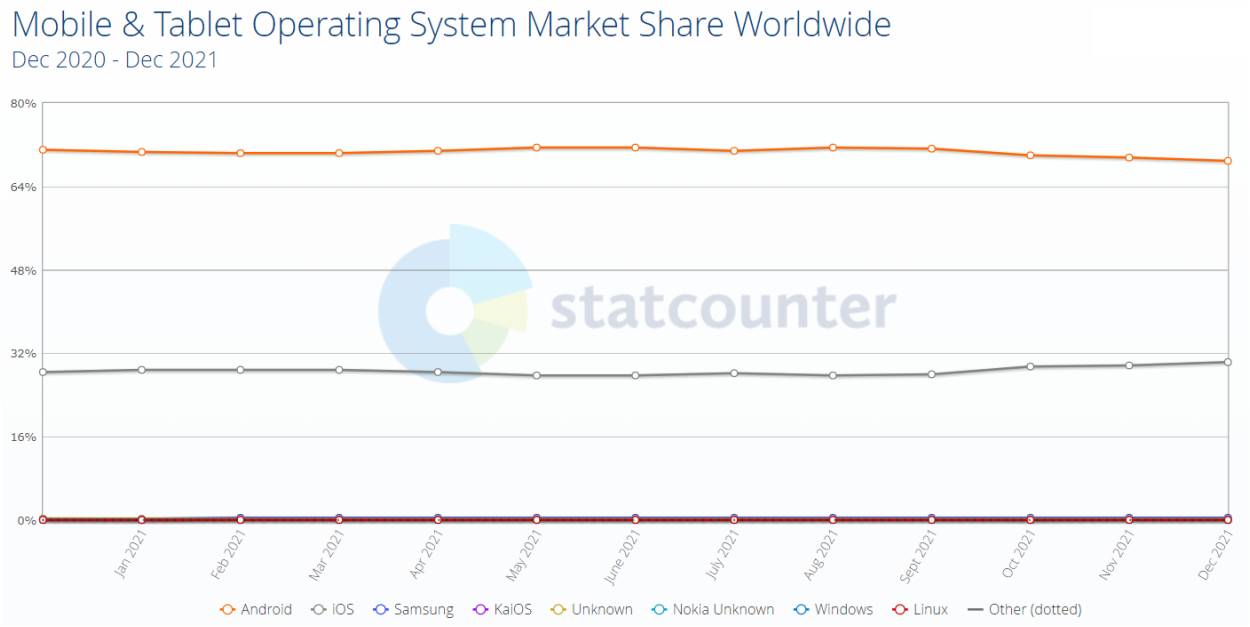

Figura 2.1 – Mobile & Tablet Operating System Market Share Worldwide

El sistema operativo Android se usa en un área extremadamente amplia, se aplica para smartphones, tabletas, libros electrónicos, reproductores digitales, relojes de pulsera, pulseras deportivas, consolas de juegos, computadoras portátiles, netbooks, smartbooks, gafas Google Glass, televisores, proyectores, sistemas para automóviles, robots domésticos y otros dispositivos.

Android está basado en código abierto como modelo de desarrollo, y admite muchas bibliotecas para aplicaciones: bibliotecas de funciones estándar, bibliotecas de multimedia, bibliotecas de gráficos 2D, bibliotecas de gráficos 3D, bibliotecas de navegadores web, etc. En comparación con las aplicaciones regulares de Linux, las aplicaciones de Android están sujetas a reglas adicionales: Content Providers: intercambio de datos entre aplicaciones; Resource Manager: acceso a recursos como archivos XML, PNG, JPEG; Notification Manager: acceso a la barra de estado; Activity Manager: administración de aplicaciones activas.

Inicialmente el entorno de desarrollo integrado (IDE) utilizado era Eclipse con el plugin de Herramientas de Desarrollo de Android (ADT). También existen plugins para IntelliJ IDEA y NetBeans IDE. Ahora se considera como entorno oficial Android Studio, descargable desde la página oficial de desarrolladores de Android. [6]

Android Studio admite lenguajes de programación, como Kotlin, Java y C ++.

Java es un lenguaje de programación general, concurrente y orientado a objetos, cuyo objetivo es simplificar las dependencias de implementación, permitiendo que los desarrolladores de apps de cliente-servidor, lo escriban una sola vez y sean capaz de aplicarlo a cualquier soporte. [7]

Actualmente Java ocupa un lugar destacado en los rankings de popularidad de lenguajes de programación, segundo en los rankings IEEE Spectrum (2020)[8] y TIOBE (2021)[9].

Ventajas importantes del Android para el proyecto:

- Android es un software popular.
- Android es un software libre y de código abierto.
- Hay muchas de herramientas de desarrollo disponibles para el Android.
- El desarrollo de aplicaciones Android se puede hacer en Java.
- Admite una gran variedad de bibliotecas diferentes.

2.2. App nativa vs app híbrida

App nativa	App híbrida
La aplicación nativa está desarrollada y optimizada específicamente para el sistema operativo determinado y la plataforma de desarrollo del fabricante (Android, iOS, etc). **Ventajas de App nativa frente App híbrida:** Cualquier cambio en la aplicación híbrida requiere mucha atención y mucho tiempo, como resultado, la tasa de actualización de funcionalidad y agregación de nuevas tecnologías es más lenta que en la aplicación nativa. En la aplicación híbrida el rendimiento es más lento, ya que se trata de tecnologías integradas a plataformas móviles, que no siempre es lo ideal. Las aplicaciones de iOS y Android tienen sus propios estándares de diseño y dado que una aplicación híbrida no coincide con ellos, es necesario «adaptarlos» apropiadamente, es decir, requiere el doble de trabajo en el campo de diseño. Una de las serias desventajas de las aplicaciones híbridas es su inseguridad. Mientras que la aplicación nativa se puede cifrar antes de lanzarse a la tienda oficial, ya que se aprovecha los protocolos integrados al dispositivo en cuestión. A la hora de desarrollar una aplicación nativa, existe una gran variedad de	La aplicación híbrida aprovecha al máximo la versatilidad de un desarrollo y tiene la capacidad de adaptación al dispositivo, es decir está desarrollada para varias plataformas al mismo tiempo y está escrita en un lenguaje universal. **Ventajas de App híbrida frente App nativa:** La hibridez provee acceso a los mercados de varias plataformas simultáneamente. Dado que, si la aplicación está escrita para dos plataformas, ingresa simultáneamente a dos mercados. Debido a esto, la cantidad de usuarios potenciales también se duplica junto con la posibilidad de que la aplicación se descargue.

herramientas incluidas en el SDK de una determinada plataforma. Es decir, todo lo que necesita es usar las herramientas en su aplicación nativa. Lo que permite crear de una manera sencilla y rápida un código personalizado de alta calidad para cada aplicación. En el caso de un híbrido, algunas herramientas no están disponibles, lo que genera ciertas dificultades. Dado que una aplicación nativa cumple con los requisitos estándar de una determinada plataforma desde el principio, es poco probable que tenga problemas al iniciar su aplicación en App Store o Play Store.	

[10]

Tabla 2.2 – App nativa vs app híbrida

2.3. Apps similares

Actualmente, Google Play está lleno de juegos del estilo: Roguelike&Arcade, y todos tienen altos números de instalación y uso, lo que indica que el interés del usuario en juegos como este conjunto de géneros es realmente grande. Como ejemplo, considere el juego "Archero". [11]

Archero combina principalmente elementos de juegos de rol hipercasuales y de acción. El bucle de juego hipercasual es muy simple, al igual que los controles: solo te mueves y disparas. Además, permite sesiones de juego frecuentes y más cortas, que es otro elemento básico de los juegos casuales. El objetivo es completar cadenas de mazmorras, cada una tiene decenas de niveles, lo que también lo convierte en un juego roguelike.

Archero se convirtió rápidamente en un juego exitoso en 2019 cuando se lanzó, y para 2021, obtuvo más de $64 millones en ingresos. Archero comenzó fuerte, alcanzando un máximo de 4,5 millones de descargas en la segunda mitad de 2019. En general, Archero obtuvo más de 45 millones de descargas de todos los tiempos. El 28% de los jugadores de Archero son menores de 25 años, el 28% son jugadores de 25 a 34 años, el 44% son personas mayores

de 35 años. Por lo tanto, Archero cubre todos los grupos de edad. La mayoría de los usuarios de Archero juegan de 3 a 10 minutos al día (alrededor del 33%). Un poco menos del 25% de los usuarios juegan de 1 a 3 minutos y el 18% de los usuarios pasan de 10 a 30 minutos al día jugando este juego. [12]

2.4. Conclusión

Para un procesador móvil, lo más importante es el consumo de energía, la compacidad y el calentamiento, y solo entonces, el rendimiento. En este sentido, los juegos para móviles se esfuerzan por ser lo más sencillos posibles, evitando cálculos y análisis complejos salvo que sean absolutamente necesarios. La cantidad de objetos activos, la frecuencia de las actualizaciones y los controles de acción, la visualización gráfica alta, la presencia de componentes no utilizados u objetos que no se ven en la pantalla: todos estos aspectos afectan en gran medida a la capacidad de respuesta del procesador del teléfono móvil. Por lo tanto, la optimización de procesos es uno de los aspectos más importantes del desarrollo de aplicaciones móviles, especialmente para juegos.

Las aplicaciones nativas e híbridas tienen sus ventajas y desventajas, sus fortalezas y debilidades. Después de un análisis exhaustivo de la tarea, se decidió que este proyecto se desarrolla como una aplicación nativa. Para atraer la mayor cantidad de usuarios potenciales, es necesario utilizar el sistema operativo más popular, que para este año 2021-2022 es Android.

La clave del éxito de los juegos móviles es su simplicidad. La mayoría de los usuarios buscan una aplicación de entretenimiento simple que se pueda usar durante un breve descanso.

Por lo tanto la finalidad de esta aplicación tiene naturaleza entretenida, permitiendo al usuario relajarse en un corto plazo, usando nuevas interacciones visuales e interactivas en el juego. Estilo gráfico simple, los personajes y enemigos inusuales, y las características únicas del juego, batallas rápidas y cortas, todo esto sirve para mantener la tensión durante toda la partida y ayuda a atraer a los usuarios potenciales.

3. Objetivos y Alcance

3.1. Objetivos del producto

- Crear competencia y rivalidad entre los jugadores.
- Contener una gran cantidad de elementos activos únicos con sus propias características y apariencias, para no crear un juego demasiado simple y monótono.
- Visualizar el juego de la forma simple e intuitiva, si es necesario, agregar explicaciones para una acción en particular y su consecuencia.
- Crear un mecanismo de generación aleatoria de los mapas para mantener la imprevisibilidad y el alto interés de la próxima partida.
- Crear un mecanismo de generación de mapas del juego con un aumento gradual de la complejidad del juego, que sean lo suficientemente difíciles, pero no demasiados para el usuario promedio.
- Guardar todos los datos sobre los usuarios y su progreso en una base de datos local con Room.

3.2. Objetivos del cliente

- Crear una aplicación de entretenimiento.
- Crear un modelo de control de personajes moderno, intuitivo, simple y fácil de usar.
- Mantener una alta concentración y reacción del cliente durante la partida.
- Mantener el interés del juego tanto de los jugadores "nuevos" como de los "antiguos".
- Crear un servicio user friendly, con una interfaz que ayude a los usuarios a utilizarlo.
- Crear una aplicación que estimule interactivamente las capacidades psicológicas y mejore la atención, y la percepción del usuario.
- Crear una aplicación en sistema operativo Android a la que los usuarios puedan acceder en cualquier momento.

3.3. Target

Este servicio no tiene restricciones de edad debido a la ausencia de cualquier material desagradable, pero principalmente está orientado a jóvenes de 18 a 30 años.

4. Metodología

Un modelo de desarrollo es una representación abstracta de un proceso de software, cada modelo representa el proceso de desarrollo de software de una manera en particular.[13] Hay varios modelos para perfilar el proceso de desarrollo, cada uno de las cuales cuenta con pros y contras.

Modelo en cascada: este modelo de desarrollo sigue estrictamente un camino previamente planificado, solo cuando se completa una etapa, comienza otra. La falta de flexibilidad y la incapacidad de cambiar los hitos planificados, son importantes factores negativos de este modelo para el proyecto actual.

Modelo en espiral: las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a ninguna prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior.[14] Un rasgo distintivo de este modelo es la especial atención a los riesgos que afectan a la organización del ciclo de vida.

Desarrollo ágil de software: en el desarrollo de software, las prácticas ágiles incluyen el descubrimiento de requisitos y la mejora de soluciones a través del esfuerzo colaborativo de equipos multifuncionales y autoorganizados con sus clientes y usuarios finales, planificación, desarrollo evolutivo, entrega temprana, mejora continua y respuestas flexibles a los cambios en los requisitos, la capacidad y la comprensión de los problemas a resolver.[15]

Codificación y corrección: este modelo de desarrollo es, más que una estrategia predeterminada, el resultado de una falta de experiencia o presión que se ejerce sobre los desarrolladores para cumplir con una fecha de entrega. Sin dedicar tiempo de forma explícita para el diseño, los programadores comienzan de forma inmediata a producir código. Antes o después comienza la fase de pruebas de software y los inevitables errores que se encuentran han de eliminarse antes de poder entregar el software.[16]

4.1. Modelo en espiral

Debido a que las dos cosas más importantes durante el desarrollo de este proyecto son la flexibilidad y la atención al riesgo, este proyecto utiliza un modelo de desarrollo de software en espiral.

Cada vuelta de la espiral corresponde a un fragmento o versión del software, en él se especifican los objetivos y características del proyecto, se evalúa su calidad y se planifica el trabajo de esta vuelta de la espiral. Por lo tanto, los detalles del proyecto se aclaran y concretan de manera consistente, como resultado de lo cual se identifica una opción razonable, que se lleva a la implementación.

Cada bucle se divide en 4 sectores:

1. Definición de un objetivo.
2. Evaluación y resolución de riesgos.
3. Desarrollo y pruebas.
4. Planificación de una iteración posterior.

En cada ciclo, se recopilan en espiral varios modelos de procesos de desarrollo de software. Como resultado, la salida es un producto terminado.

El desarrollo por iteraciones refleja el ciclo en espiral de creación de sistemas. La finalización incompleta del trabajo en cada etapa le permite pasar a la siguiente etapa, sin esperar a la finalización completa del trabajo en la etapa actual. Con un método de desarrollo iterativo, el trabajo que falta se puede completar en la próxima iteración. [17]

La tarea principal es mostrar un producto viable lo antes posible, activando así el proceso de especificación y complemento de los requisitos.

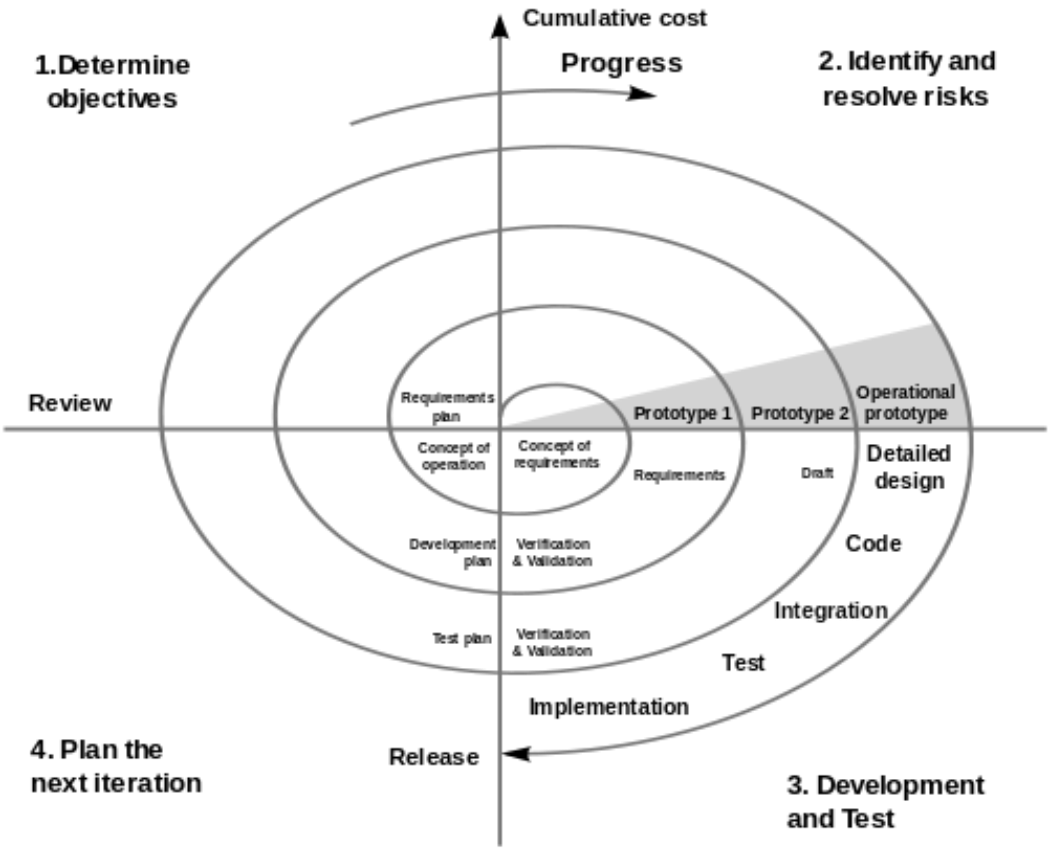

Figura 4.1 – Spiral model (Boehm, 1988)

4.1.1. Definición de un objetivo

En esta parte, se realiza el estudio del proyecto. Se determina a los elementos más importantes para el ciclo actual que otorgan un valor positivo al producto. Se crea un plan de requerimientos y tareas que deben realizarse.

4.1.2. Evaluación y resolución de riesgos

Considera las diversas tecnologías disponibles y evalúa cuál es la más adecuada para su uso en el desarrollo de este proyecto. Se está realizando un estudio de viabilidad del producto nuevo tanto a nivel técnico como económico y medioambiental. En conclusión, se realiza un análisis completo y se determina el tiempo de desarrollo de cada tarea.

4.1.3. Desarrollo y pruebas

Al inicio de esta fase se investiga y se aprende lo necesario para el desarrollo durante este ciclo. Este proyecto no es un estudio teórico o una recopilación de datos de investigación. La información principal que se busca durante el desarrollo de este proyecto es la descripción y documentación de las bibliotecas y funciones específicas que utiliza Android.

Para el desarrollo del proyecto se utiliza tanto el sitio oficial de Android para desarrolladores [18], como un sitio de preguntas y respuestas con programadores profesionales y aficionados: Stack Overflow [19].

El repositorio de GitHub[20][21] se utiliza para almacenar el proyecto. Cada ciclo es un producto-prototipo terminado, que se desarrolla en una rama diferente, marcando el nombre del prototipo y su versión.

Se trabaja en sprints de una semana en la que se van desarrollando las funcionalidades necesarias, en la última parte del sprint el principal objetivo es encontrar y solucionar el máximo número de errores posible.

La transición del ciclo procede de acuerdo con el plan, incluso si no se completa todo el trabajo planificado.

4.1.4. Planificación de una iteración posterior

La etapa final analiza el trabajo realizado y el resultado, vuelve a verificar la necesidad de elementos inacabados, considera los problemas y dificultades que surgieron durante el desarrollo, comprueba la viabilidad y la necesidad de continuación de desarrollo, realiza ajustes en las tareas para un nuevo ciclo.

5. Desarrollo

5.1. Requerimientos funcionales

Al menos debe tener los siguientes requisitos funcionales:

1. Permitir al usuario crear una nueva partida.
2. Permitir al usuario continuar el juego actual.
3. Permitir al usuario eliminar todos los datos.
4. Crear el mundo del juego en el formato de una cadena secuencial de mapas con una dificultad de paso que aumenta gradualmente.
5. Crear un mecanismo para generar automáticamente mapas de forma aleatoria.
6. Crear un mecanismo para devolver el jugador al menú principal en caso de pérdida.
7. Crear un mecanismo para transferir el jugador a un nuevo mapa.

5.2. Requerimientos tecnológicos

Al menos debe tener los siguientes requisitos tecnológicos:

1. Guardar los datos localmente aplicando Room.
2. Crear al menos 2 joysticks virtuales.
3. Usar lenguaje de programación: Java.
4. Usar IDE: Android Studio.
5. Usar sistema operativo: Android.

5.3. Bases de datos

Una base de datos es una herramienta que recopila datos, los organiza y los relaciona para que se pueda hacer una rápida búsqueda y recuperar con ayuda de un ordenador.[22] Una base de datos almacena datos y los conecta en una unidad lógica junto a los metadatos necesarios para su procesamiento. Las bases de datos son instrumentos de gran utilidad para gestionar grandes ficheros y facilitar la consulta de información. En muchas, además, puede

definirse un esquema de permisos que establece qué personas o programas pueden acceder a los datos.[23]

La base de datos tiene un papel importante en este proyecto. Para guardar todos los elementos, se utiliza el Room. Room es una librería de base de datos creada por el equipo de Android en Google y que simplifica la tarea de trabajar con bases de datos en Android.[24] A travez de la base de datos, podemos guardar el estado actual del juego, los resultados del jugador y la configuración técnica general de la aplicación, lo que nos permite interrumpir y continuar a usar la aplicación en cualquier momento desde la etapa donde la dejó el usuario.

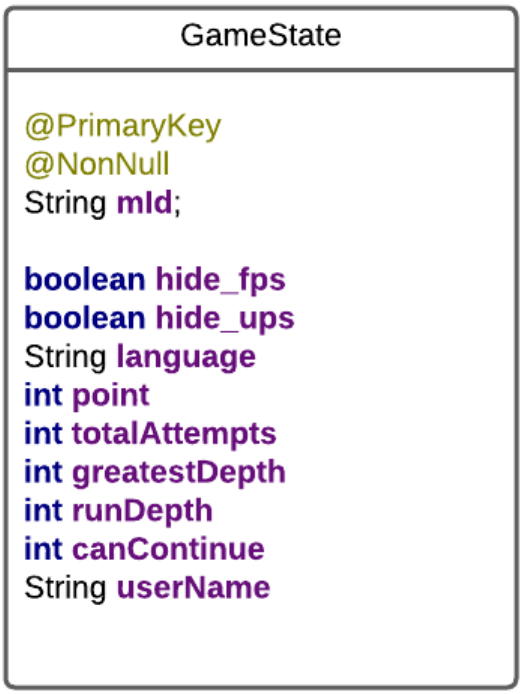

Figura 5.3.1 – BBDD GameState

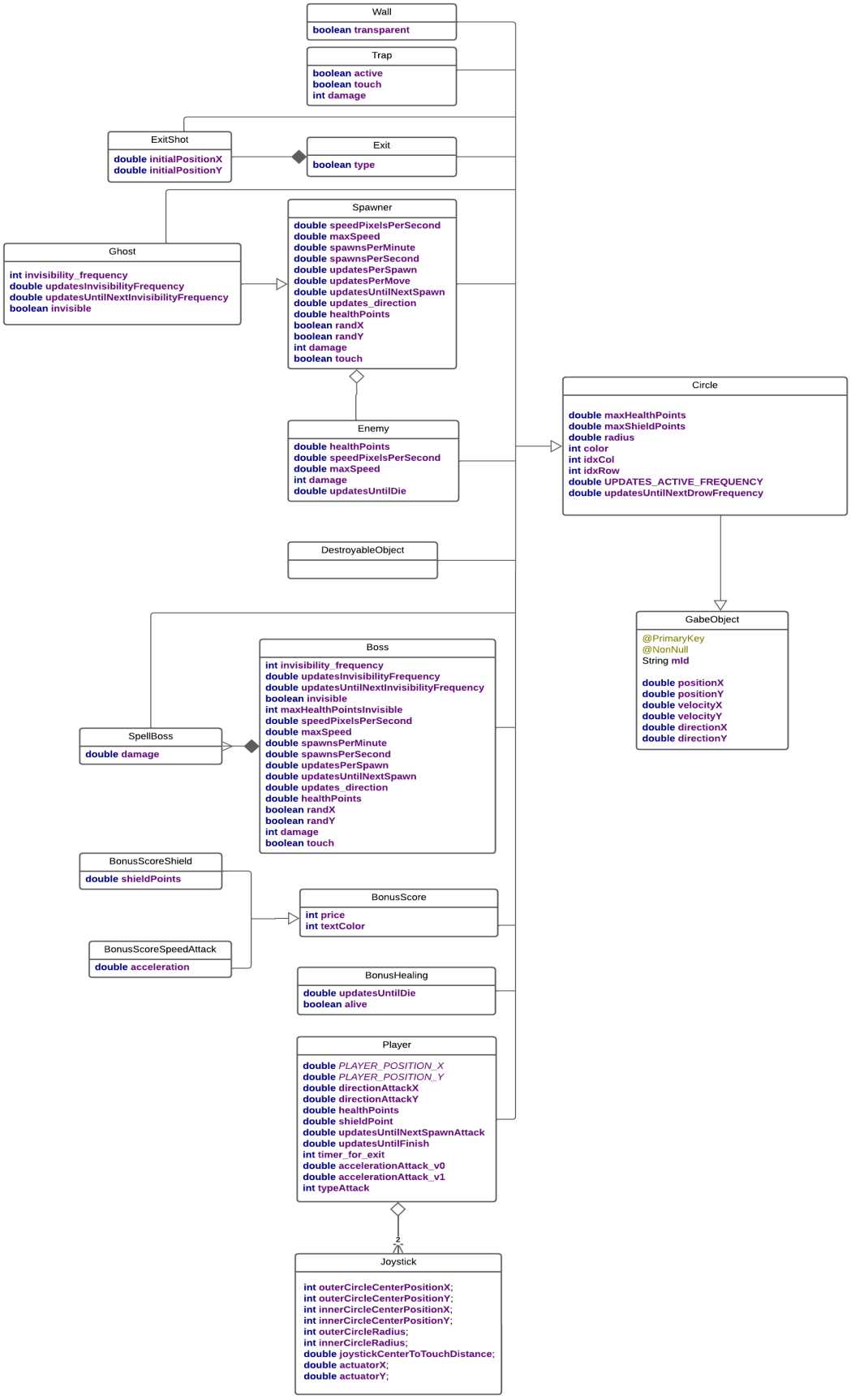

Figura 5.3.2 – BBDD GameObject

5.4. Implementación

La aplicación se ha desarrollado en Android Studio utilizando Java como lenguaje de programación y SQLite para la base de datos.

Se ha establecido una estructura en paquetes siguiendo el patrón de diseño MVC (Modelo-Vista-Controlador) según las funciones de las clases que contiene y son los siguientes:

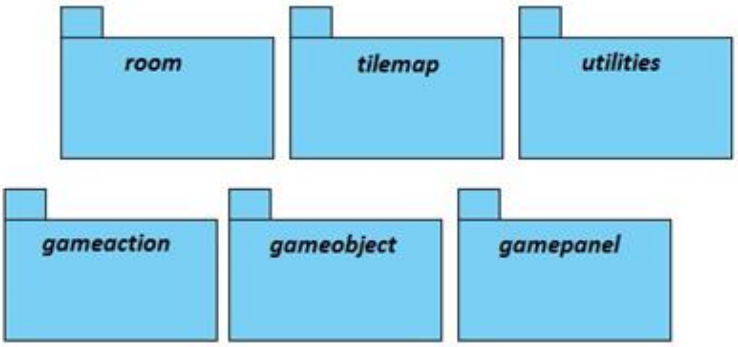

Figura 5.4 – Paquetes de la aplicación

- room: contiene clases que gestionan las conexiones y accesos a la base de datos en SQLite, su ubicación, creación y borrado.
- tilemap: contiene las clases para generar y llenar la mapa del juego.
- utilities: contiene herramientas y clases específicas para el soporte de la aplicación.
- gameaction: este paquete contiene las actividades del juego.
- gameobject: contiene las clases activas del juego.
- gamepanel: contiene clases para la parte interactiva de la pantalla.

5.4.1. Elementos del juego

El personaje del jugador y los "enemigos" tienen las barras de salud.

El juego tiene un tipo especial de moneda, obtenida por la destrucción de "enemigos".

El jugador tiene 3 tipos de hechizos:

1. №1 un hechizo para hacer daño a los enemigos.
2. №2 un hechizo que hace que las paredes sean transitables para el jugador.
3. №3 un hechizo para hacer daño a los enemigos, con más velocidad de ataque pero menos daño que №1.

Cuando el jugador toca un modificador de salud y tiene suficientes puntos, los puntos se gastan y el modificador de salud se destruye, apareciendo un nuevo indicador en la parte superior de la barra de salud del jugador que muestra su salud adicional.

Cuando el jugador toca un modificador de hechizo y tiene suficientes puntos, los puntos se gastan y el modificador de hechizo se destruye, la velocidad de activación del hechizo aumenta.

El juego tiene 4 tipos de enemigos: "normal", "spawner", "ghost" y "boss". Hay un 25% de probabilidad de dejar atrás la curación cuando se destruye.

1. El "normal" se mueve hacia el jugador y, cuando lo toca, le inflige daño y se autodestruye. Puede ser atacado por el jugador.
2. El "spawner" se mueve en una dirección aleatoria y genera una cantidad limitada de enemigos "normales". Puede ser atacado por el jugador.
3. El "ghost" tiene 2 estados, visible e invisible. Cuando "ghost" es invisible, sigue al jugador y no puede ser atacado por el jugador. Cuando "ghost" es visible, puede ser atacado por el jugador y se mueve en una dirección aleatoria y, cuando lo toca un jugador, le inflige daño.
4. El "boss" es un enemigo especial, con un gran cantidad de salud y las habilidades combinadas de otros enemigos como "spawner" y "ghost".

El objeto destructible solo se destruye con el hechizo №1 del jugador. Tiene un 25% de probabilidad de dejar atrás una curación cuando se destruye.

Las paredes son una barrera impenetrable para todos los objetos, en condiciones normales.

La trampa cambia de estado cada 5 segundos - activa / inactiva. Cuando la trampa está inactiva, no interactúa con el jugador. Cuando la trampa está activa, inflige daño al jugador cuando se tocan.

La curación está en el campo de juego por un tiempo limitado. Después de 10 segundos se destruye. Si el jugador toca la curación, restaura completamente su salud.

Cuando se destruyen todos los enemigos, aparece una transición a un nuevo mapa. Cuando el jugador toca este objeto, se abre una nueva mapa.

El juego tiene 3 tipos de mapas: "mercado", "habitual", "especial".

1. El "mercado" es un mapa sin enemigos, pero con modificadores para el jugador.

2. El "habitual" es un mapa con los enemigos estándar como "normal", "spawner" y "ghost".
3. El "especial" es un mapa con un único enemigo del tipo "boss".

5.4.2. Paquete Gameobject

Las clases activas del juego, que se encuentran en el paquete gameobject, representan a los quince nodos de información que muestran los objetos activos del juego. Muchos de estos objetos tienen funciones y variables iguales o similares, por lo que se utilizan tres clases abstractas para simplificar el código.

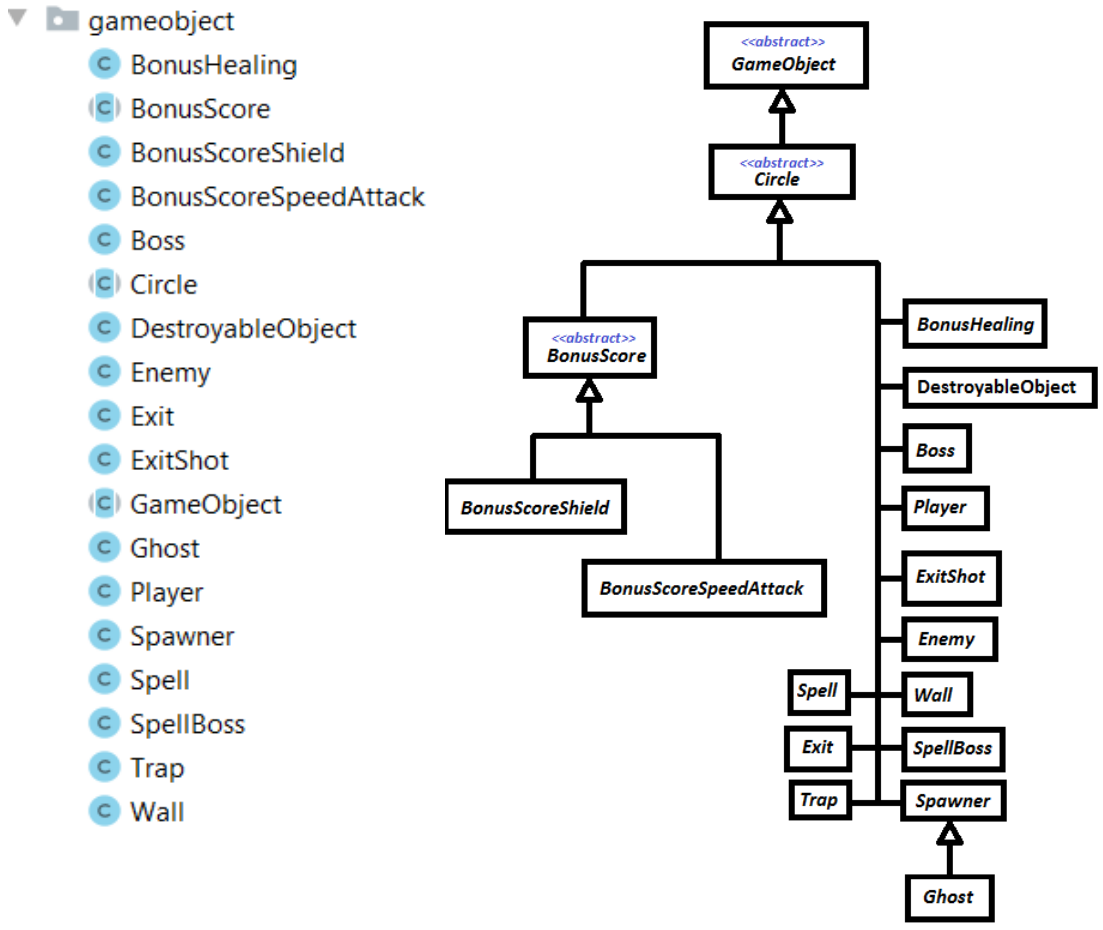

Figura 5.4.2 – Clases de “gameobject”

En esta etapa de desarrollo, todos los elementos del juego tienen la forma de un círculo, ya que es más simple para los cálculos de distancia y colisión.

Gameobject – Representa la clase principal para todos los elementos del juego, define el ID del objeto y las coordenadas del objeto en general. También incluye una función que determina la distancia en píxeles entre dos GameObjects.

Circle – Almacena datos sobre la posición del sprite, el radio del objeto y la frecuencia con la que se actualiza el objeto. También incluye una función que determina la colisión de dos objetos de la clase Circle.

BonusScore – Almacene el precio y el color del texto para todos los bonos que se pueden comprar en el juego.

BonusScoreShield – Incluye tres subtipos definidos por el constructor, los subtipos difieren entre sí en precio, en utilidad para el usuario y en código del sprite. La utilidad agrega un margen de salud extra además de la salud básica.

BonusScoreSpeedAttack – Incluye dos subtipos definidos por el constructor, los subtipos difieren entre sí en precio, en utilidad para el usuario y en código del sprite. La utilidad modifica el ataque del personaje del jugador.

BonusHealing – Es un objeto de existencia temporal, que se elimina automáticamente después de la expiración del tiempo o después del contacto con el jugador. Restaura la salud primaria del jugador en caso de colisión.

DestroyableObject – Es un objeto inamovible con dos subtipos. El subtipo cambia después de recibir el primer daño del jugador, luego de recibir el segundo daño, el objeto se remueve del mapa, dejando atrás un BonusHealing con la cierta probabilidad. Es un objeto infranqueable que detiene el movimiento de otros objetos a través de sí mismo.

Boss – Incluye dos subtipos definidos por el constructor, los subtipos difieren entre sí en el radio del objeto, en la velocidad de movimiento, en la salud, en la frecuencia de transición a un estado invulnerable y en código del sprite. Tiene movimiento en una dirección aleatoria y puede atacar al jugador invocando al SpellBoss.

SpellBoss – Incluye dos subtipos definidos por el constructor, los subtipos difieren entre sí el daño infligido, en la velocidad de movimiento, y en código del sprite. El subtipo depende del subtipo del Boss que le convoca. Tiene un movimiento rectilíneo simple.

Player – Es un objeto del juego sobre el cual el usuario tiene control total. El movimiento y la dirección del ataque están determinados por dos joysticks interactivos que se crean automáticamente en el punto de contacto del usuario con la pantalla.

Spell – Incluye tres subtipos definidos por el constructor. Cada subtipo tiene su propia función única. El primer subtipo es necesario para interactuar con el DestroyableObject e

influir daño a los enemigos, el segundo subtipo es necesario para interactuar con el Wall, el tercer subtipo también inflige daño a los enemigos, pero no puede interactuar con el DestroyableObject.

Enemy – Incluye tres subtipos definidos por el constructor, los subtipos difieren entre sí el daño infligido, en la velocidad de movimiento, y en código del sprite. Tiene un movimiento decidido hacia el Player.

Wall – Es un objeto inamovible con dos subtipos. El subtipo cambia después de recibir el ataque especial del jugado. Es un objeto infranqueable que detiene el movimiento de otros objetos a través de sí mismo mientras está en el primer subtipo, pero después de pasar al segundo subtipo, permite que el jugador lo atraviese.

Exit – Es un objeto especial e inamovible que permite que el usuario sea transferido a un nuevo mapa del juego. Este objeto aparece después de que se destruyen todos los enemigos en el mapa actual.

ExitShot – Este elemento del juego sirve para ayudar al usuario señalando la dirección del objeto Exit. Tiene un movimiento decidido hacia el Player.

Trap – Es un objeto inamovible con dos subtipos. El subtipo cambia de forma cíclica y automática después de la expiración del tiempo predeterminado. Cuando se activa el primer subtipo, el objeto está inactivo y no interactúa con el jugador de ninguna manera. Cuando se activa el segundo subtipo, el objeto se activa y causa daño al jugador en caso de colisión.

Spawner – Incluye cinco subtipos definidos por el constructor, los subtipos difieren entre sí en el radio del objeto, en la velocidad de movimiento, en la salud, en la frecuencia de generación del objeto Enemy y en código del sprite. Tiene movimiento en una dirección aleatoria.

Ghost – Es un objeto con dos subtipos. El subtipo cambia de forma cíclica y automática después de la expiración del tiempo predeterminado. Cuando se activa el primer subtipo, el objeto es visible y se mueve en una dirección aleatoria, y puede recibir daño del jugador. Cuando se activa el segundo subtipo, el objeto se vuelve invisible y se mueve decidido hacia el Player, y no puede recibir daño del jugador.

5.4.3. Paquete Gamepanel

Las clases para la parte interactiva de la pantalla, que se encuentran en el paquete gamepanel, representan a los cuatro nodos de información que muestran los objetos para la interacción con el jugador.

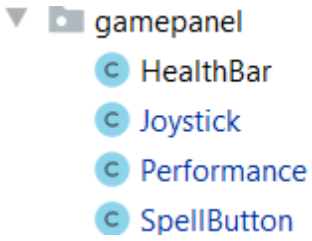

Figura 5.4.3 – Clases de “gamepanel”

HealthBar – Este elemento muestra el salud actual del objeto enlazado. Se muestra en el mapa del juego como dos pequeños rectángulos ubicados ligeramente más arriba que el objeto anclado.

Joystick – Este elemento finge la acción del mando analógico. Se situa en el lugar donde el jugador hace clic en la pantalla táctil, y luego define el movimiento, la dirección y la profundidad del toque del usuario en la pantalla.

SpellButton - Este elemento permite cambiar el subtipo del Spell, también muestra el subtipo del Spell actual y el tiempo hasta que se activa.

Performance - Muestra el rendimiento de la aplicación mostrando FPS y UPS actuales.

5.4.4. Paquete Room

Room proporciona una capa de abstracción en SQLite para permitir un acceso más fluido a la base de datos.[25] Por el Room, se puede crear rápidamente bases de datos SQLite y realizar operaciones como crear, leer, actualizar y eliminar.

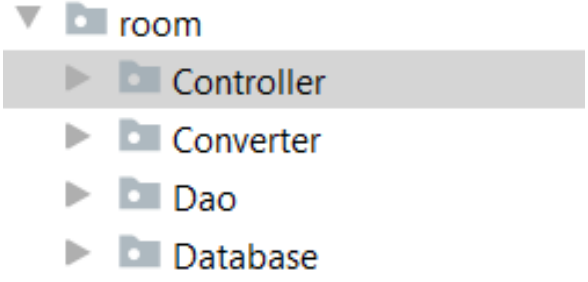

Figura 5.4.4.1 – Paquetes de “room”

Entity: en lugar de crear la tabla SQLite, crearemos la Entidad. La entidad no es más que una clase modelo anotada con @Entity. Las variables de esta clase son nuestras columnas, y la clase es nuestra tabla.

Controller: proporciona una llamada segura entre el DAO y la Entity.

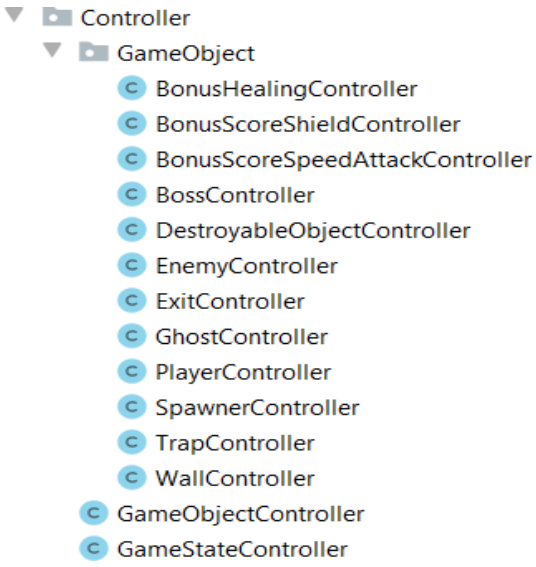

Figura 5.4.4.2 – Controller de “room”

Converter: en los casos en que los objetos de Entity contienen campos que no son primitivos y no se pueden almacenar en la base de datos, se utiliza TypeConverter que simplifica y adapta los datos a un formato aceptable para SQLite.

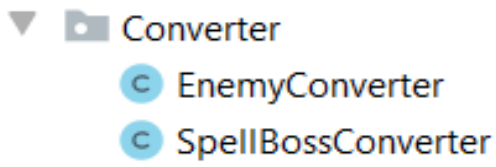

Figura 5.4.4.3 – Converter de “room”

Database: Es una clase abstracta donde definimos todas nuestras entidades.

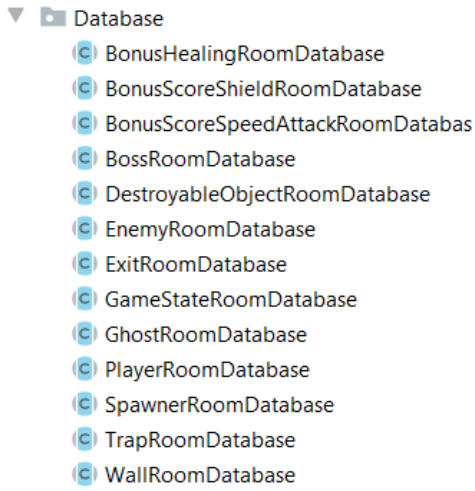

Figura 5.4.4.4 – Database de “room”

DAO: Objeto de acceso a datos. Es una interfaz que define todas las operaciones que necesitamos realizar en nuestra base de datos.

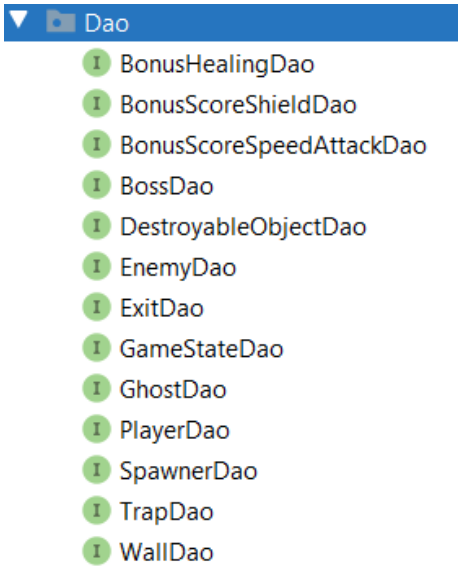

Figura 5.4.4.5 – DAO de “room”

5.4.5. Paquete GameAction

Todas las clases responsables de acciones, cálculos complejos, visualizaciones y movimientos en el juego están en el paquete GameAction. Este paquete contiene tres grandes clases responsables de tres acciones principales.

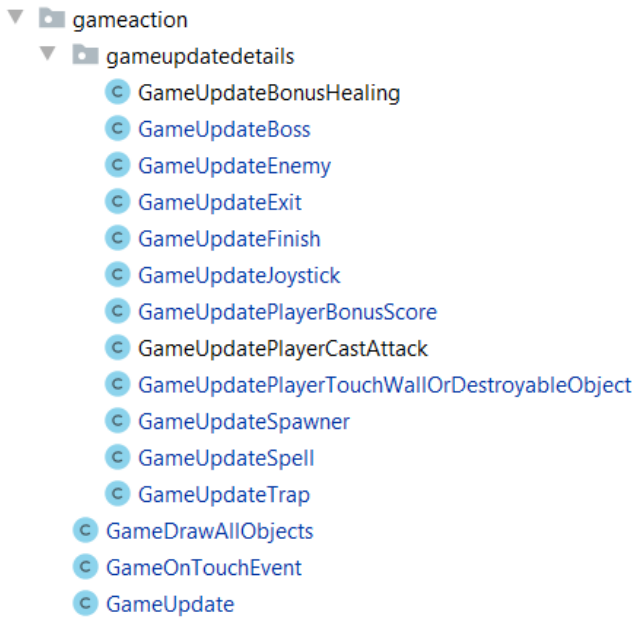

Figura 5.4.5 – Clases de “gameaction”

GameDrawAllObjects - Esta clase es responsable de la parte gráfica del juego, llamando y visualizando todo lo que se necesita.

GameOnTouchEvent – Esta clase es responsable de recibir comandos y toques del usuario en la pantalla, se interactúa con Joysticks y SpellButton.

GameUpdate –Esta clase es responsable de todos los cálculos y acciones del juego, se divide en subclases que se encuentran en el paquete GameUpdateDetails.

5.4.6. Paquete Utilities

El paquete Utilities contiene elementos auxiliares que facilitan la interacción con la aplicación, y también contiene funciones que no requieren un uso constante y son llamados en casos especiales y específicos.

Figura 5.4.6 – Clases de “utilities”

GameState – Esta clase es responsable del estado técnico actual de la actividad del juego, almacena los resultados del juego del jugador y datos sobre el idioma actual.

StaticVar - Esta clase almacena variables estáticas que deben usarse y llamarse desde cualquier clase.

Utils - Esta clase almacena funciones estáticas que deben usarse y llamarse desde cualquier clase.

5.4.7. Paquete TileMap

El paquete TileMap contiene todos los elementos y objetos principales del juego.

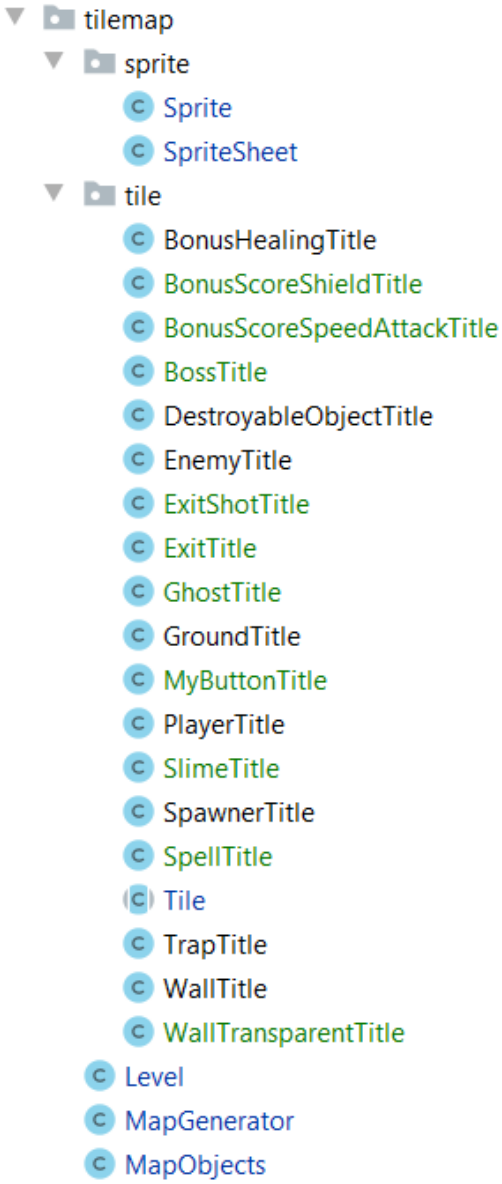

Figura 5.4.7.1 – Clases de “tilemap”

Level – Esta clase es un depósito para crear y especificar un mapa de juego. Cada símbolo representa la ubicación de un objeto en particular.

Código del objeto	Tipo de objeto
0	Player
1	Wall
2	-
3	Spawner o Trap
4	Spawner
5	Enemy o Wall
6	DestroyableObject
7	Trap
8	Exit
9	BonusScore
#	Boss

Tabla 5.4.7 – Códigos de objetos en el mapa.

```
public static final char[][] RANDOM_LAYOUT_10 = new char[][] {
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
        {1, 2, 2, 2, 2, 2, 2, 2, 2, 1},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
};
```

Figura 5.4.7.2 – Ejemplo de un mapa de juego.

MapGenerator – Esta clase es un creador de mapas, recibe un boceto de la clase Level y lo llena con elementos aleatorios del juego.

MapObjects – Esta clase es el depósito de todos los elementos y objetos del juego, y también es responsable de su creación y eliminación.

Sprite – Esta clase es un controlador y está asociada con la clase SpriteSheet.

SpriteSheet – Esta clase es un almacenamiento activo para imágenes de juegos, cuando se llama al constructor, recibe una llamada en formato String, encuentra la imagen requerida

en la aplicación y la corta en mosaicos del tamaño deseado y la guarda en una lista. Por lo tanto, cuando se llama a dibujar un objeto, encuentra el elemento deseado en la lista y lo muestra, sin tener que cargar una imagen grande cada vez, lo que reduce la carga en el procesador.

Tile – Es una clase proxy abstracta que asocia un Tile particular con un Sprite.

```
public Tile() {
    sprite= new Sprite(this.getClass().getSimpleName().toLowerCase());
}
```

Por lo tanto, al agregar un xxxTile nuevo al juego, no es necesario realizar ninguna llamada más que un solo “super()” en su constructor.

```
public class PlayerTitle extends Tile {
    public PlayerTitle() { super(); }
}
```

Figura 5.4.7.3 – Ejemplo de contacto entre una clase y su imagen.

Esta aplicación utiliza Sprites del formato png creados manualmente con Adobe Photoshop y Adobe Illustrator con pruebas gratuitas limitadas. El elemento visual recortado del juego tiene un tamaño múltiplo de 128x128 píxeles. Si el objeto tiene más de 128 píxeles, se utiliza el multiplicador de tamaño. Entonces, por ejemplo, el objeto Boss que tiene un tamaño de 240 píxeles usa un multiplicador de dos y un mosaico de 256x256 píxeles.

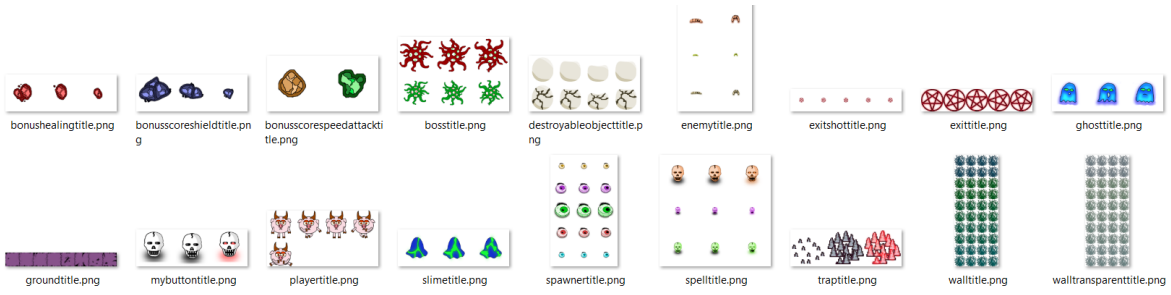

Figura 5.4.7.4 – Sprites del juego.

5.4.8. Multilenguaje

Esta aplicación ofrece acceso a una selección de diferentes idiomas, en esta etapa de desarrollo solo se proporcionan dos idiomas. Inglés como idioma predeterminado y español como idioma secundario. El idioma seleccionado se guarda y se selecciona automáticamente incluso después de cerrar y volver a cargar la aplicación.

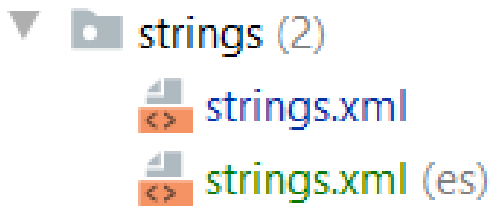

Figura 5.4.8.1 – Paquete de accesorios de idioma.

Para agregar nuevos idiomas, debe crear un nuevo archivo en la carpeta Strings y completarlo.

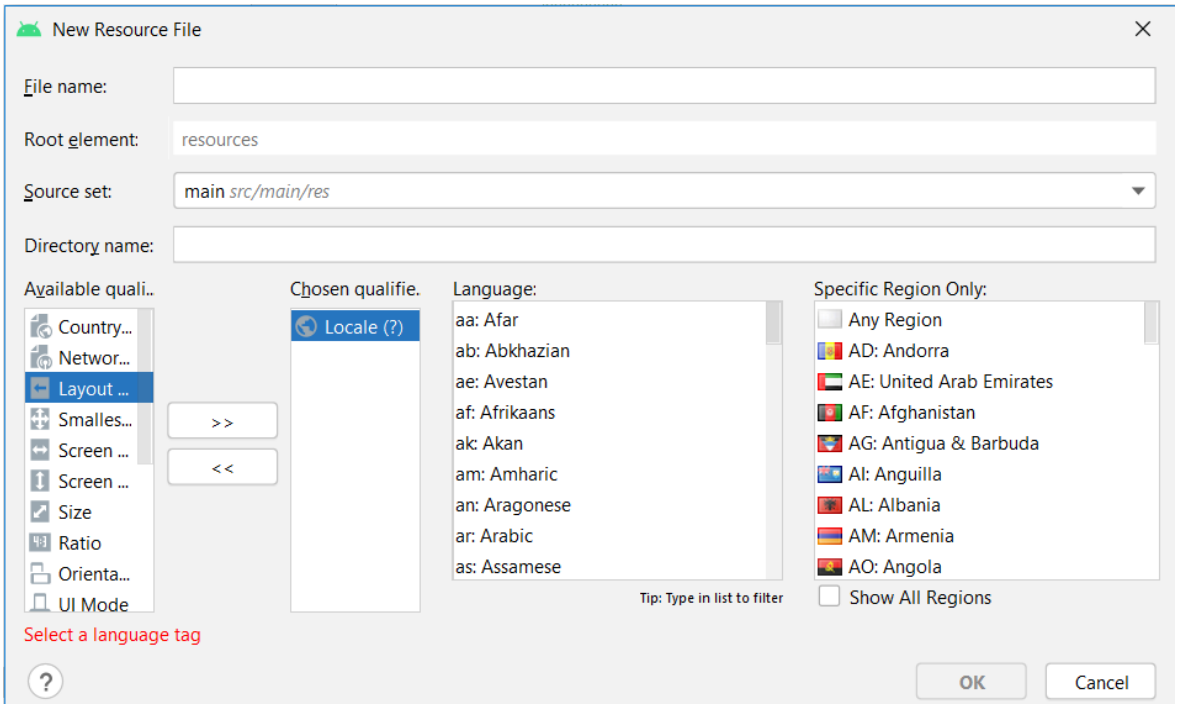

Figura 5.4.8.2 – Creación de un nuevo idioma.

Al crear una nueva actividad, establece el idioma activo como el idioma del sistema, por lo que para evitar esto, hay que verificar el idioma seleccionado por el usuario y el idioma actual de la aplicación, y si no coinciden, se llama a una función para cambiar el idioma y recargar la aplicación.

5.4.9. Activity

Este proyecto se basa en el uso de tres actividades que se abren con la expectativa de un resultado.

La primera actividad, que es la pantalla principal, ofrece acceso a la elección de diferentes idiomas, ofrece acceso a la transición al menú principal y también permite cerrar la aplicación.

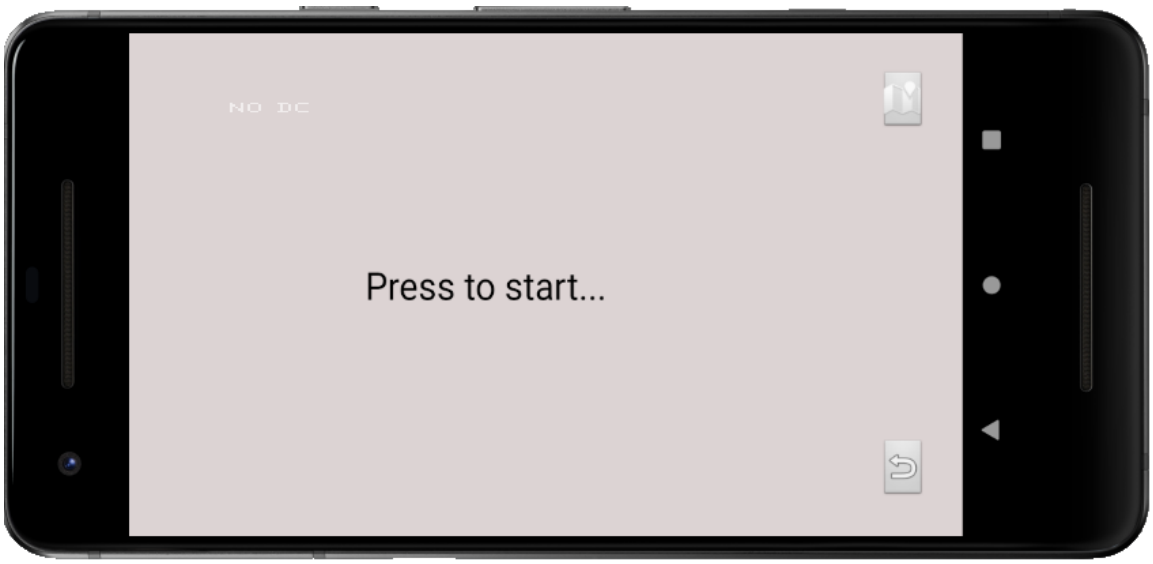

Figura 5.4.9.1 – La primera actividad, ventana inicial.

Figura 5.4.9.2 – La primera actividad, selección de idioma.

Figura 5.4.9.3 – La primera actividad, desactivación de la aplicación.

La segunda actividad, se encarga de la información sobre los resultados actuales del jugador, la configuración de elementos visuales, la limpieza de la base de datos, el manual del juego, el inicio de una nueva partida y la descarga de la última partida.

Figura 5.4.9.4 – La segunda actividad, ventana inicial.

Figura 5.4.9.5 – La segunda actividad, ventana de la configuración.

Figura 5.4.9.6 – La segunda actividad, ventana de la limpieza de la base de datos.

Figura 5.4.9.7 – La segunda actividad, ventana del manual del juego.

La tercera actividad es la principal de este proyecto y es responsable del componente activo del juego. Dentro de esta parte, se crea y se lleva a cabo toda la actividad del juego. Al finalizar el partido, se cierra la actividad y se vuelve a la segunda actividad con la transferencia del resultado a ella y se guarda todos los datos del juego en el Room, también todos los datos se guardan antes de cerrar la aplicación, lo que permite interrumpir el juego en cualquier momento y continuarlo sin perder el progreso.

5.4.10. Pruebas con el emulador

Device: Pixel 2 (5”, 1920 × 1080)

Target: Android 11.0 (Google Play)

CPU/ABI: x86

API Level: 30

Figura 5.4.10 – Dispositivo

5.4.10.1. Versión técnica

Simulación del juego con objetos representados como simples formas circulares.

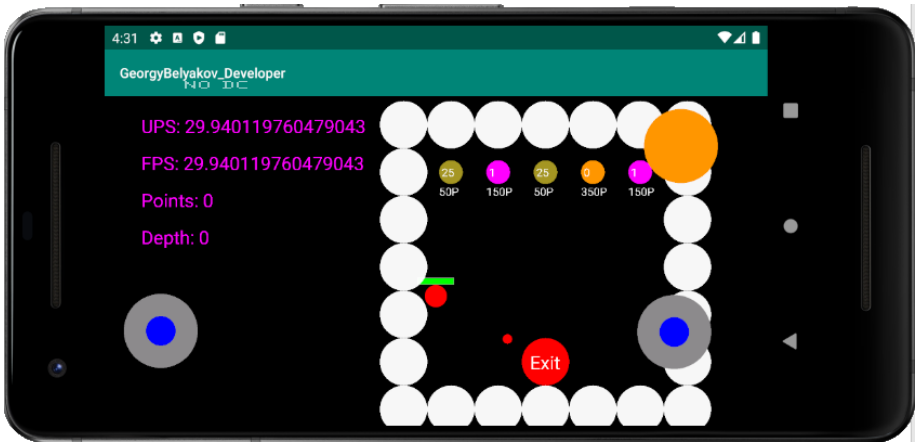

Figura 5.4.10.1.1 – Versión técnica, mapa “mercado”

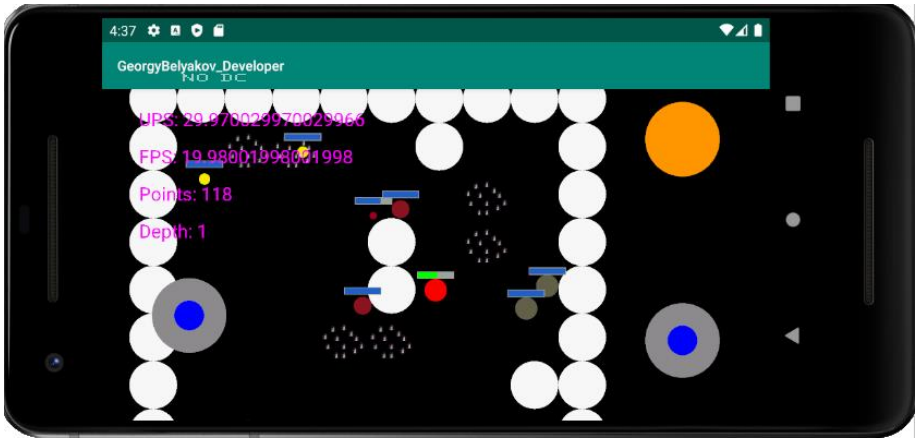

Figura 5.4.10.1.2 – Versión técnica, mapa “habitual”

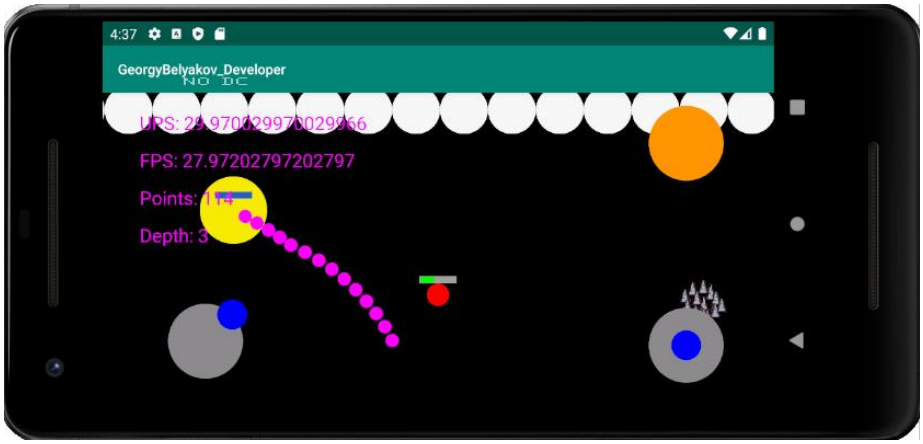

Figura 5.4.10.1.3 – Versión técnica, mapa ”especial”

5.4.10.2. Versión gráfica

Simulación del juego con objetos representados mediante elementos Sprite.

Figura 5.4.10.2.1 – Versión gráfica, mapa “mercado”

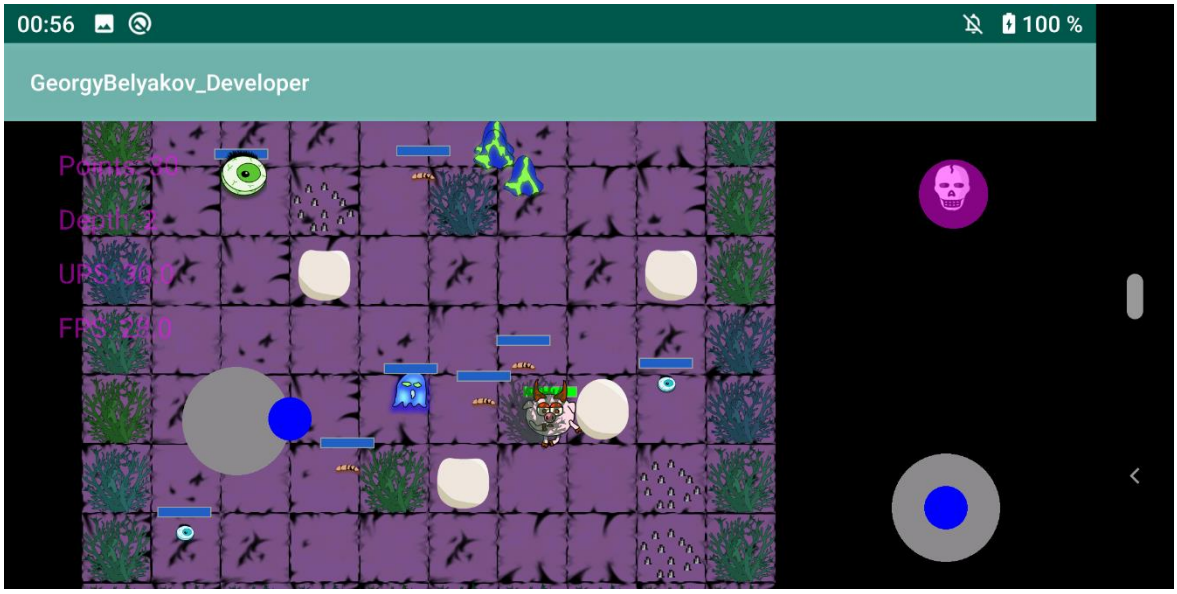

Figura 5.4.10.2.2 – Versión gráfica, mapa “habitual”

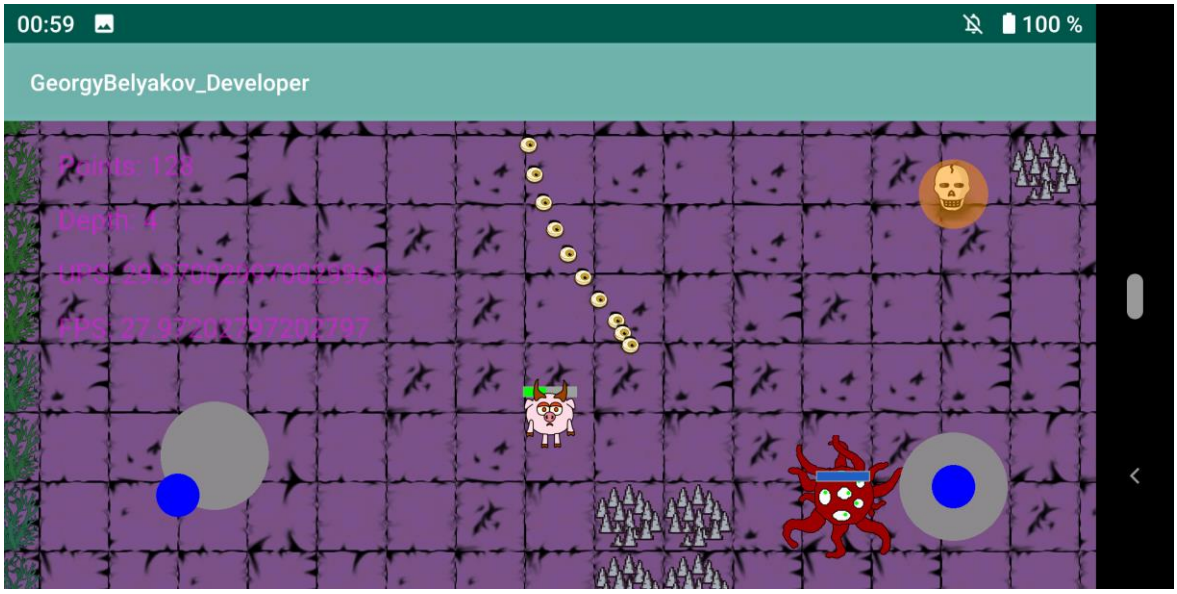

Figura 5.4.10.2.3 – Versión gráfica, mapa ”especial”

6. Conclusiones

Respondiendo a las tareas planteadas al inicio del trabajo y extrayendo conclusiones generales, cabe destacar lo siguiente. Si bien el objetivo principal era desarrollar y crear una aplicación móvil para una gran cantidad de usuarios, hay puntos que se consideraron importantes, pero que no se implementaron por completo en el tiempo planificado.

En primer lugar, debe tenerse en cuenta que el objetivo original de este proyecto es crear un backend completo y óptimo para una aplicación creada con un alto nivel técnico. Esto incluye la investigación sobre cómo las bases de datos se crean y se administran correctamente en las aplicaciones de Android, la investigación sobre cómo se aplica y se utiliza el Sprite en los dispositivos móviles y la aplicación del multilingüismo. A nivel técnico la aplicación utiliza factorías, interfaces y clases abstractas, clases proxy, patrones singleton, convertidoras y controladores, el código evita duplicidades o cálculos innecesarios, todos los elementos complejos y grandes se dividen por partes, lo que simplifica mucho la lectura del código y su modificación. Gracias a la optimización del código, la introducción de un nuevo idioma no requiere cambios importantes, solo la adición de un archivo con la traducción y la indicación de este archivo en la lista de selección. Agregar nuevos efectos visuales o cambiar texturas visuales tampoco requiere ningún esfuerzo o cambios en el código del programa, lo que simplifica enormemente el trabajo al mejorar el componente gráfico del juego. La adición de nuevos objetos de juego también está sistematizada y simplificada al máximo, por lo que se puedes mejorar el contenido del juego sin ningún problema. Todos estos puntos fueron alcanzados y fueron de vital importancia en el desarrollo de este trabajo.

Sin embargo, en términos de jugabilidad, la aplicación sigue siendo demasiado simple y monótona. Los enemigos con los que luchan los jugadores tienen pocas especies y tipos, son extremadamente similares entre sí y no son de particular interés durante el juego. La generación aleatoria de mapas existente se crea áreas de juego de plantilla que no pueden diferir mucho entre sí. La parte visual de este proyecto es de un nivel extremadamente bajo, los elementos del juego están dibujados a mano, tienen muchos defectos y mala calidad. Esta aplicación no cuenta con elementos especiales y específicos que no se encontrarían en otros juegos similares, y por lo tanto no es capaz de suscitar emoción e interés entre el usuario.

El sistema operativo Android es extremadamente conveniente y amigable para el desarrollador, tiene funciones y comandos predefinidos muy útiles que permiten ahorrar mucho tiempo durante el desarrollo. Aunque, uno de los puntos a mejorar de cara al futuro es la adaptación de la aplicación para iOS, que ocupa un porcentaje importante del mercado mundial. Los gráficos son lo más relevante de un videojuego, obviamente una de las principales mejoras necesarias para este proyecto es mejorar el componente visual del juego. El segundo punto importante es aumentar la cantidad de elementos únicos del juego con los que el usuario podría interactuar.

Finalmente, teniendo en cuenta la valoración de los resultados, se concluye que, a pesar de las carencias y sugerencias de mejora de algunos de los puntos y elementos desarrollados anteriormente, la resolución del trabajo es positiva, aunque se consigue el objetivo principal. Se ha desarrollado una aplicación móvil con la que el usuario puede distraerse fácilmente y pasar su tiempo jugando y disfrutando, pero la aplicación no es capaz de mantener a un usuario por mucho tiempo.

7. Bibliografía

[1] «Videojuego de rol de acción - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Videojuego_de_rol_de_acci%C3%B3n (accedido 09.06.2022).

[2] «Videojuego de disparos - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Videojuego_de_disparos (accedido 09.06.2022).

[3] «Videojuego de mazmorras - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Videojuego_de_mazmorras (accedido 09.06.2022).

[4] «Sprite (videojuegos) - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Sprite_(videojuegos) (accedido 09.06.2022).

[5] «Mobile & Tablet Operating System Market Share Worldwide | Statcounter Global Stats». https://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/ (accedido 21.01.2022).

[6] «Android - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Android (accedido 21.01.2022).

[7] «Lenguajes de programación», Java. https://webtematica.com/lenguajes-de-programacion (accedido 21.01.2022).

[8] «Top Programming Languages 2020», IEEE Spectrum. https://spectrum.ieee.org/top-programming-language-2020 (accedido 21.01.2022).

[9] «index | TIOBE - The Software Quality Company», TIOBE. https://www.tiobe.com/tiobe-index/ (accedido 21.01.2022).

[10] «Native vs. Hybrid App | Umbrella IT», consultoría IT. https://umbrellait.com/blog/native-vs-hybrid-app/ (accedido 26.01.2022).

[11] «Archero – Apps en Google Play». https://play.google.com/store/apps/details?id=com.habby.archero&hl=es_419&gl=US (accedido 26.01.2022).

[12] «Archero Advertising Strategy: Everything You Need to Know - Udonis». https://www.blog.udonis.co/mobile-marketing/mobile-games/archero-advertising (accedido 26.01.2022).

[13] «Ingenieria en Software - Tema 3 Modelos Desarrollo: Modelos Evolutivos». http://tema3isoftware.blogspot.com/p/modelos-de-desarrollo-tecnicas-y.html (accedido 24.04.2022)

[14] «A Spiral Model of Software Development and Enhancement», pp. 24. ACM SIGSOFT Software Engineering Notes.

[15] «Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing», pp. 121. Ken W. Collier.

[16] «7: Lifecycle Planning», pp. 140. Steve McConnell.

[17] «Desarrollo en espiral - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Desarrollo_en_espiral (accedido 21.01.2022).

[18] «Desarrolladores de Android | Android Developers». https://developer.android.com/ (accedido 21.01.2022).

[19] «Stack Overflow - Where Developers Learn, Share, Build Careers». https://stackoverflow.com/ (accedido 21.01.2022).

[20] «GitHub - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/GitHub (accedido 21.01.2022).

[21] «GitHub: Where the world builds software · GitHub». https://github.com/ (accedido 21.01.2022).

[22] «Base de datos: ¿qué tipos hay y cómo funciona conectada a un software?». https://www.ticportal.es/glosario-tic/base-datos-database (accedido 07.06.2022).

[23] «Bases de datos: historia, funciones y modelos - IONOS». https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos/ (accedido 07.06.2022).

[24] «Room, la librería de Base de datos de Android - DevExperto». https://devexperto.com/room-la-libreria-de-base-de-datos-de-android/ (accedido 07.06.2022).

[25] «Skeleton (computer programming) - Wikipedia». https://en.wikipedia.org/wiki/Skeleton_(computer_programming) (accedido 26.01.2022).