Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería informática

Hive: Juego de estrategia para Android con inteligencia artificial

Autora:
Lucía Gil Román

Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería informática

Hive: Juego de estrategia para Android con inteligencia artificial

Autora:
Lucía Gil Román
Tutora:
Mª Aránzazu Simón Hurtado

Agradecimientos

Me gustaría agradecer a Mª Aránzazu Simón Hurtado que apostara por la realización de este trabajo y me ayudara a lo largo de todo su desarrollo.

También agradecer a mis padres y a mi hermana el apoyo recibido durante todos mis años dedicados al estudio de la informática.

Por último, querría dar a las gracias a mi amigos por haber hecho más llevadera esta recta final y no haber dudado de mi en ningún momento.

Resumen

El desarrollo de aplicaciones móviles es un tema que está en pleno auge en el mundo de la informática, y no es para menos: es difícil encontrar a alguien que no disponga de un dispositivo móvil y lo use diariamente para comunicarse con otras personas, consultar información o echar alguna partida. Precisamente, en este último uso se centra este trabajo, en la creación de un juego para dispositivos Android capaz de mantener al usuario entretenido y que a la vez mejore su capacidad de planificar estrategias.

Existen muchas clases de juegos para esta clase de dispositivos, estando muy de moda los de multijugador online y los de tipo puzzle, pero apenas existen juegos en los que puedas echar una partida contra el propio dispositivo, sin necesidad de conexión a Internet o pagar por esta funcionalidad extra. Este trabajo cubre ese hueco, permitiendo jugar innumerables partidas contra la inteligencia artificial diseñada para este propósito.

Hay que tener en cuenta que la ejecución de una inteligencia artificial suficientemente potente conlleva un coste computacional muy alto, por lo que es lógico que para dispositivos que están pensados principalmente para comunicarse apenas existan esta clase de juegos.

Otra funcionalidad que disponen muchas aplicaciones de juegos de estrategia, sobre todo los más clásicos como pueden ser las damas o el ajedrez, que es jugar contra otro jugador usando ambos el mismo dispositivo como si se tratara del tablero de juego real. Poder jugar con otros jugadores de esta forma es una funcionalidad muy atractiva por lo que fue añadida a la aplicación desarrollada en este trabajo.

Palabras clave: Inteligencia Artificial, Juego, Android

Abstract

Mobile application development is an issue at its very peak in the world of computing, and it's no wonder, almost everyone has a mobile device that uses everyday to chat with other people, google information or play simple games. Precisely, this last use is the main theme of this work, the creation of a game for Android devices that can keep the user entertaining and, at once, helps to improve the ability to plan strategies.

There are many categories of games for this type of devices, being in fashion those about online multiplayer and also puzzle types, but there is hardly any game with the option to play against the device itself, without an Internet connection or paying for that extra functionality. This work tries to fill that gap, allowing the user to play countless games against the artificial intelligence designed for this purpose.

We have to keep in mind that the execution of a sufficiently powerful artificial intelligence has a very high computational cost, so is logical that for devices whose main purpose is to communicate there are hardly games of this kind.

Another functionality that many application strategy games have, especially the classic ones like checkers or chess, is the possibility to play a game against another player both using the same device as if it was the actual board game. This method of multiplayer functionality is very attractive so it was added to the developed application.

Key words: Artificial Intelligence, Game, Android

Índice

1. Introducción

Este Trabajo de Fin de Grado titulado “Juego de estrategia para Android con inteligencia artificial” ha sido desarrollado por Lucía Gil Román y tutelado por Mª Aránzazu Simón Hurtado.

Este capítulo servirá para introducir conceptos que permitan entender mejor el tema a desarrollar, los motivos por los que se eligió dicho tema, los objetivos planteados y la estructura de la memoria.

1.1 Motivación y objetivos del proyecto

1.1.1 Motivación

Como tema para el trabajo de fin de grado se tenían inicialmente tres puntos claros, que servirían como base para definir el proyecto:

- Resultaba atractiva la idea de realizar una aplicación Android, pues se quería mejorar el conocimiento que se tenía en ese campo.

- También necesitaba que la aplicación realizara alguna función interesante y llevaba mucho tiempo planteando la posibilidad de realizar algún tipo de juego para este sistema operativo.

- Por último, se querían poner en práctica conceptos impartidos en la asignatura de fundamentos de inteligencia artificial, pues se podían aplicar a juegos.

Una vez se establecieron las bases, se empezó a investigar qué juego de estrategia sería el elegido para ser el tema principal del trabajo de fin de grado. Para ello se indagó mucho en un foro de Internet dedicado a esta clase de juegos [1]. Dicha página web permite buscar juegos ordenados por categorías, de modo que se buscaron juegos abstractos (sin temática) y de estrategia, y se organizaron por ranking, para obtener los juegos mejor valorados. En esa lista ordenada de juegos aparece en quinta posición "Hive" (Colmena en español), un juego formado únicamente por fichas hexagonales que representan insectos.

Una vez leídas las reglas del juego y comprobado que eran realizables, se empezó a investigar posibles aplicaciones o videojuegos que ya existieran sobre él:

- Dispositivos Android: Dispone de una versión [2] que permite al usuario jugar contra otro jugador online y jugar contra otro jugador usando ambos el dispositivo como tablero de juego. Este último modo de juego es el único que funciona correctamente, pues el modo multijugador online necesita mejorar bastante, jugadores que se sabe que están ambos conectados en ese modo son incapaces de iniciar una partida. Su puntuación en Google Play es de 3,4/5.

- Dispositivos iOS: La página web oficial del juego ofrece una versión para este tipo de dispositivos [3], que realiza las mismas funciones que su versión Android. Esta aplicación no es gratuita, y según los pocos usuarios que han hecho comentarios sobre ella en itunes está llena de errores.

- Consola Xbox: Tiene una versión para esta consola de la que se conoce muy poco porque no se pueden ver los comentarios de los usuarios y la página oficial no especifica su contenido [4].

Por lo tanto, se llegó a la conclusión de que a pesar de existir aplicaciones sobre este juego no eran lo suficientemente buenas, los usuarios no estaban satisfechos con su adquisición y además, ofrecían servicios que no funcionaban correctamente.

En resumen, se decidió realizar este proyecto como trabajo de fin de grado por los siguientes motivos:

- El objetivo principal de este tipo de juegos, entretenimiento y mejora de la capacidad de estrategia, es suficientemente motivadores como para permanecer durante varios meses trabajando en el proyecto sin perder ni el interés ni las ganas de mejorarlo día a día.

- El juego elegido para ser implementado, “Hive”, tiene una versión para dispositivos Android que no dispone de una opción para echar partidas contra una inteligencia artificial, de lo cual se quejan varios usuarios de la aplicación en los comentarios de la referencia proporcionada.

- Como motivo extra podemos decir que cubre gran parte de la materia impartida durante los cuatro cursos del grado:

    - Fundamentos de programación
    - Estructura de datos y algoritmos
    - Programación orientada a objetos
    - Interacción persona-computadora
    - Fundamentos de inteligencia artificial
    - Fundamentos de ingeniería del software
    - Modelado de sistemas software
    - Análisis y diseño de algoritmos
    - Diseño de software
    - Tecnologías para el desarrollo software
    - Planificación y gestión de proyectos
    - Sistemas móviles

1.1.2 Objetivos

El objetivo principal de este proyecto es la implementación para dispositivos móviles Android de un juego de mesa de estrategia ya existente y permitir al usuario jugar contra una inteligencia artificial o contra otro usuario real usando el mismo dispositivo.

Para facilitar la consecución de este objetivo se establecieron una serie de metas más simples que se deberían alcanzar en orden sucesivo:

- Crear un programa en Java sin interfaz gráfica para implementar el desarrollo de una partida contra otro jugador. Se deberían cumplir todas las reglas de las instrucciones del juego que se encuentran en el ANEXO I.

- Implementar el objetivo anterior en un proyecto Android y añadirle interfaz gráfica para que el usuario pueda jugar partidas de forma más cómoda.

- Crear una inteligencia artificial contra la que pueda echar partidas un usuario real. No se establece como objetivo que tenga niveles de dificultad, bastará con solo uno que mantenga el desarrollo de la partida interesante, es decir, que la inteligencia artificial no sea ni muy simple ni muy compleja.

Una vez se hayan completado estos objetivos se habrá conseguido el objetivo principal del Trabajo de Fin de Grado.

1.2 Introducción de conceptos teóricos

Esta sección servirá para explicar brevemente algunos conceptos que sirven como base para el desarrollo del proyecto.

1.2.1 Android

Android es un sistema operativo basado en el núcleo de Linux que fue diseñado principalmente para dispositivos móviles con pantalla táctil. [5] Para exponer más claramente qué supone la elección de este sistema operativo se lista a continuación una serie de ventajas y desventajas.

Ventajas:

- Los dispositivos Android ocupan un 82.2% de la cuota de mercado mundial, frente al 14.6% de iOS o el 2.5% de Windows [6] lo cual hace que sea un producto que tiene mucho soporte tanto físico como de software y para el que se desarrollan muchas aplicaciones tanto gratuitas como de pago.

- Es código abierto, lo que permite que cualquiera pueda crear aplicaciones para este sistema operativo sin ningún tipo de licencia.

- Está pensado no solo para dispositivos portables como pueden ser los teléfonos inteligentes, relojes o tabletas, sino que también desarrolla para televisores inteligentes y sistemas de navegación GPS.

- Es un sistema multitarea capaz de gestionar varias aplicaciones abiertas a la vez, suspendiendo aquellas que pasen a un segundo plano.

Desventajas:

- Una de las características nombradas como ventaja es a la vez una desventaja. El hecho de que pueda gestionar varias aplicaciones a la vez ocasiona que el que haya que mantenerlas en ese estado afecte al estado de la batería del dispositivo.

- El usuario debe de estar pendiente de las aplicaciones que tiene en segundo plano y no esté utilizando si quiere alargar la vida de la batería de su dispositivo.

- No es intuitivo para todos los usuarios: una parte de ellos se adapta fácilmente a este tipo de dispositivos, pero otros ven complicado hasta las tareas más sencillas como pueden ser llamar por teléfono o enviar un mensaje.

1.2.2 Juegos de mesa

Los juegos de mesa están compuestos por un tablero y una serie de elementos, pudiendo estos ser fichas, dados, cartas…, pueden ser jugados por un número variable de personas de las que se requiere desde uso de razonamiento estratégico, memoria o coordinación, o simplemente se recurre al azar para su desarrollo

1.2.3 Inteligencia artificial

Según [7], la inteligencia artificial consiste en conseguir que los ordenadores sean capaces de razonar para realizar determinadas tareas del mismo modo que lo hacen los humanos y animales.

Podemos programar ordenadores de modo que adquieran capacidades que les permitan resolver varios problemas de tipo aritmético, búsqueda u ordenación. También podemos conseguir que los ordenadores jueguen determinados juegos de mesa como el tres en raya. Esta clase de problemas fueron considerados inicialmente relacionados con la inteligencia artificial, pero al ser resueltos de formas más variadas y extensas han desaparecido del área del desarrollo de inteligencia artificial.

Pero hay muchas cosas que a los ordenadores no se les da bien y que los humanos ven como triviales: Reconocer caras familiares, hablar un idioma, decidir qué hacer después o ser creativos. Estas acciones son del dominio de la inteligencia artificial: intentar averiguar qué tipo de algoritmos se necesitan para conseguir que los ordenadores puedan realizarlas de la forma más parecida a como lo haría un humano.

1.2.4 Inteligencia Artificial aplicada a juegos de mesa

Las primeras aplicaciones de la inteligencia artificial al mundo de los videojuegos fueron como oponente en populares juegos de mesa. El más popular y que más interés despierta es el ajedrez, pues en los últimos 40 años se ha mejorado notablemente la capacidad de razonamiento de las inteligencias artificiales creadas para jugar a él.

Para enseñar a una inteligencia artificial a jugar deberá aprender a tomar decisiones por sí misma, y para ello se emplean diferentes técnicas. Algunas de ellas son:

- Árboles de decisión: Generan todos los posibles caminos que se pueden tomar. Son fáciles de implementar y de entender.

- Máquinas de estado: Modelan el comportamiento de un sistema con entradas y salidas, haciendo que estas últimas dependan tanto de las señales de entrada como de estados anteriores.

- Lógica difusa: Proporciona un marco matemático que permite modelar la incertidumbre de los procesos cognitivos humanos de forma que pueda ser tratable por un computador [8].

- Sistemas basados en reglas: Técnica que trabaja empleando reglas y comparando resultados obtenidos de estas para aplicar nuevas reglas.

Teoría de juegos

La teoría de juegos es una disciplina de las matemáticas que se encarga de estudiar juegos idealizados. No tiene mucha aplicación para los juegos basados en acciones en tiempo real, pero sí la tiene para juegos basados en turnos como lo son determinados juegos de mesa.

Esta teoría clasifica juegos dependiendo del número de jugadores, el tipo de objetivo que tienen dichos jugadores y la información que cada jugador tiene del juego:

- Número de jugadores: La mayoría de los algoritmos basados en turnos que se han diseñado para juegos de mesa asumen que dicho juego es para dos jugadores. Las optimizaciones de

dichos algoritmos también suponen que solo hay dos jugadores, por lo tanto, aunque estos algoritmos se pueden adaptar para más de dos jugadores no serían tan eficientes.

- Objetivo: Ganar, es el objetivo principal de casi todos los juegos de estrategia. El jugador gana si todos sus oponentes pierden. Por lo que el jugador puede intentar ganar o puede intentar que el resto de jugadores pierdan, el resultado sería el mismo.

- Información: De los juegos se puede tener "información perfecta" o "información imperfecta". Cuando se tiene información perfecta se conocen todos los datos del juego en todo momento, por ejemplo, en el ajedrez se conoce la posición de todas las figuras, amigas o enemigas, en todas las jugadas y se pueden predecir qué posiciones pueden tomar estas. Mientras que en los juegos de información imperfecta hay un elemento aleatorio, dados o cartas, que impiden que se sepan todos los posibles movimientos que el enemigo puede realizar.

Esta clasificación puede ser muy útil a la hora de elegir qué clase de juego de estrategia se quiere implementar para saber si cumple los requisitos necesarios para poder diseñar una inteligencia artificial apropiada para dicho juego. [9]

Las técnicas que se aplican al diseño de inteligencias artificiales para juegos de mesa suelen ser de creación de árboles de decisión y de mejora de los algoritmos de búsqueda en dichos árboles. Para ello se emplea Minimax como creación y búsqueda en el árbol y poda alfa-beta para reducir la búsqueda en dicho árbol.

1.3 Estructura de la memoria

- **Índice**

- **Introducción**: Parte introductoria de la memoria, que contiene una introducción del tema a tratar, los objetivos y motivos del trabajo de fin de grado y una explicación de la estructura de la memoria.

- **Planificación del proyecto**: Estudio previo al comienzo del trabajo en el que se detalla aspectos como las actividades en las que se va a dividir y el tiempo que estas ocuparán, los recursos que se van a necesitar para realizarlo o los planes de contingencia ante posibles riesgos.

- **Análisis**: Capítulo que detalla el análisis realizado para poder determinar aspectos como requisitos funcionales y no funcionales, casos de uso y diagrama de clases a implementar.

- **Diseño**: Tema amplio en el que se muestra el diseño de la aplicación.

- **Implementación**: Explicación intensiva del código.

- **Pruebas**: Sección que muestra las pruebas que se realizaron para comprobar el correcto comportamiento de la aplicación ya acabada.

- **Conclusiones y trabajo futuro**: Apartado dedicado a las conclusiones de trabajo y aspectos en los que la aplicación podría mejorar.

- **Bibliografía**: Lista de las referencias utilizadas a lo largo de todo el documento, junto con su autor y la fecha en que se realizó la consulta.

- **Anexos**: Apéndices que amplían información de ciertos apartados de la memoria:

- ANEXO I: Reglas del Juego. Documento que explica cómo se juega a “Hive”, juego elegido para la realización del trabajo.

- ANEXO II: Manual de instalación. Documento que expone los pasos a seguir para instalar la aplicación en un dispositivo Android.

- ANEXO III: Manual de usuario. Documento que muestra cómo se debe interactuar con la aplicación para su correcto uso.

- ANEXO IV: Pruebas contra la máquina. Documento que contiene una tabla con registros sobre algunas de las partidas jugadas contra la inteligencia artificial.

2. Planificación

2.1 Resumen del proceso

2.1.1 Introducción

Como ya se ha explicado con anterioridad, el propósito de este proyecto es la creación de una aplicación para dispositivos Android que permita a uno o dos usuarios echar partidas a un juego de mesa llamado “Hive”.

El proyecto se empezó a idear sobre los meses de noviembre y diciembre, en los cuales solo se pensaron y apuntaron ideas sobre cuál podría ser el juego elegido y cómo se podría desarrollar.

El inicio real del proyecto fue una semana después del fin de los exámenes del primer cuatrimestre que fue el 18 de enero, por lo tanto se empezó el lunes 25 a trazar un plan.

2.1.2 Suposiciones y limitaciones

Las suposiciones y limitaciones del trabajo que se dedujeron las primeras semanas del inicio del proyecto fueron las siguientes:

- Como no se sabía la fecha de entrega del trabajo se aproximó a la de cursos pasados, suponiendo que esta sería a finales de junio principios de julio. Por lo tanto se estimó que se disponía de un periodo de 5 meses para la realización del trabajo entero.

- Según la guía docente del Trabajo de Fin de Grado esta asignatura tiene 275 horas de trabajo personal y 25 de trabajo con el tutor, por lo que se tendrá que tener en cuenta estas limitaciones.

2.1.3 Entregables del proyecto

El proyecto solo tiene un entregable oficial con fecha límite el 15 de julio de 2016, teniendo que entregar dicho día la memoria junto con el código definitivo del trabajo.

2.1.4 Calendario

El proyecto va a tener más o menos 6 fases distinguibles, que se agrupan de la siguiente forma:

Fase	Fecha de Inicio	Fecha de Fin
Creación del plan de proyecto, del documento de diseño y del documento de análisis	25/01/2016	27/02/2016
Primera fase desarrollo: Modelo del Juego	29/02/2016	26/03/2016
Segunda fase de desarrollo: Interfaz de Usuario	28/03/2016	30/04/2016
Tercera fase de desarrollo: Inteligencia Artificial	2/05/2016	28/05/2016
Pruebas de la aplicación final	30/05/2016	4/06/2016
Memoria	6/06/2016	11/07/2016

Tabla 1. Calendario propuesto para la realización del proyecto.

Como se puede observar en el calendario, se optó por agrupar el plan de proyecto, el análisis y el diseño en una primera fase que abarca más o menos 1 mes, mientras que la fase de desarrollo dispone de tres fases con un mes para cada una de ellas. Esto se ideó así porque la parte que más trabajo y esfuerzo iba a suponer es la fase de desarrollo, dividiendo esta en tres partes se limitaría la complejidad del desarrollo del código y su comprensión.

El calendario propuesto deberá seguir también las siguientes normas:

- Se trabajará de lunes a sábado, pudiendo descansar los domingos.
- Se intentará trabajar de 4 a 5 horas diarias.
- No habrá un horario establecido, puesto que a pesar de que durante el cuatrimestre esta va a ser la única tarea a realizar pueden surgir imprevistos.

Todas estas reglas se siguieron a lo largo de 2 meses, pues a mediados de marzo se tuvo que hacer un cambio drástico en el calendario: Las mañanas de los días laborables (de lunes a viernes) pasaron a estar ocupadas durante un periodo mínimo de 4 o 5 meses, por lo que se redujeron las horas de trabajo en el proyecto a 3 o 4 diarias, dejando de estar libres los domingos y trabajando horas extras los fines de semana. (Párrafo añadido el 15 de marzo de 2016).

2.1.5 Evolución del plan

Esta sección de la memoria es una sección dinámica, lo que quiere decir que se irán añadiendo, nunca modificando, cambios que surjan sobre la planificación propuesta en la sección correspondiente. Cuando se añadan cambios se indicarán en las secciones que los sufran, junto con la fecha en que fueron añadidos.

De todas formas, este documento ha sido elaborado para ayudar en el desarrollo de la aplicación y se intentará cumplir los plazos propuestos y seguir los planes de contingencia propuestos.

2.3 Glosario de Términos

Las definiciones y abreviaturas mostradas a continuación pretenden aclarar conceptos que pueden aparecer a lo largo de la memoria.

Definiciones:

- Usuario: Utilizado para identificar a la persona real que utiliza el dispositivo móvil en el que se encuentra instalada la aplicación.

- Ficha/Pieza: Representación de las piezas físicas/reales del juego.

- Selección de una ficha: Una ficha estará seleccionada cuando el usuario pulse sobre ella y se muestre en pantalla los lugares (si los hay) a los que puede desplazarse dicha ficha.

- Inteligencia artificial/Máquina: Ambos conceptos se emplearan para referirse a lo mismo,

Abreviaturas:

- IA: Inteligencia Artificial.

- TFG: Trabajo de Fin de Grado.

- SO: Sistema Operativo.

2.2 Organización de proyecto

2.2.1 Interfaces externas

La interfaz externa es la conexión entre la persona encargada de planificar y desarrollar el proyecto y el cliente que la encarga. En este caso el cliente es la misma persona encargada del desarrollo del proyecto, por lo tanto no habrá conexiones externas sobre él.

Sin embargo, sí se tendrán conexiones externas con la persona encargada de tutorizar este TFG, Mª Aránzazu Simón Hurtado: para programar tiempos, preguntar cualquier duda que surja, comprobar el estado del trabajo realizado y estructurar y corregir la memoria.

2.2.2 Roles y responsabilidades

Los diferentes roles que se pueden encontrar en la realización de un proyecto son los siguientes:

- Jefe de proyecto: Responsable de alcanzar los objetivos del proyecto. Para ello debe:
    - Identificar requisitos.
    - Establecer objetivos claros y realizables.
- Analista: Encargado de producir una especificación para un sistema que satisfaga los requisitos del sistema.
- Diseñador: Persona encargada de definir la arquitectura del sistema.
- Programador/desarrollador: Escribe, depura y mantiene el código fuente de la aplicación.
- Probador: Planifica y lleva a cabo pruebas sobre el producto final para asegurar su correcto funcionamiento.

Como solo hay una persona encargada del desarrollo total del proyecto esta abarcará todos los roles, intercambiándolos según sea necesario.

2.4 Planes de procesos de gestión

2.4.1 Plan de trabajo

En esta sección se van a explicar las actividades que conlleva la creación del proyecto, el calendario de estas y los recursos que han sido empleados.

<u>Actividades:</u>

Para encontrar las actividades se ha partido de las fases especificadas en el calendario:

1. Estudio del juego a implementar.
2. Estudio de las tecnologías a emplear.
3. Resumen del proyecto.

4. Organización del proyecto.
5. Identificación de las actividades.
6. Identificación de los recursos.
7. Crear calendario de actividades.
8. Realización del plan de control.
9. Identificación de los riesgos.
10. Plan de control de riesgos.
11. Identificación de las herramientas hardware y software.
12. Planes de procesos técnicos.
13. Revisión del documento.
14. Realización de cambios en el documento.
15. Identificación de requisitos funcionales.
16. Identificación de requisitos no funcionales.
17. Identificación de requisitos de información.
18. Identificación de los casos de uso.
19. Realización del diagrama de los casos de uso.
20. Descripción de los casos de uso.
21. Identificación de las clases del modelo del dominio.
22. Identificación de las relaciones y las multiplicidades.
23. Realización del modelo de clases.
24. Revisión del documento.
25. Realización de cambios en el documento.
26. Investigación sobre diseños aplicados a juegos
27. Investigación sobre interfaces de usuario
28. Investigación sobre IA aplicada a juegos de mesa
29. Elección de tecnologías
30. Realización del modelo arquitectónico
31. Creación de prototipos
32. Desarrollo de la lógica del juego
33. Desarrollo de la Interfaz de usuario
34. Desarrollo de la Inteligencia Artificial
35. Creación del plan de pruebas
36. Realización del plan de pruebas
37. Añadir descripción de las clases al documento de diseño
38. Completar diagrama de clases
39. Completar la memoria
40. Revisión del documento

Calendario:

En esta subsección se mostrarán las horas/hombre que se pretende emplear para cada una de las actividades.

Actividad	Duración (horas/hombre)	Predecesora
1	10	
2	10	1
3	2	2
4	2	3
5	1	4

6	1	5
7	3	6
8	2	7
9	1	8
10	2	9
11	3	10
12	1	11
13	1	12
14	1	13
15	2	14
16	2	15
17	2	16
18	2	17
19	2	18
20	5	19
21	4	20
22	1	21
23	2	22
24	2	23
25	1	24
26	10	25
27	10	26
28	14	27
29	1	28
30	1	29
31	4	30

Tabla 2. Calendario de actividades de la 1º fase.

Actividad	Duración (horas/hombre)	Predecesora
32	96	31

Tabla 3. Calendario de actividades de la 2º fase.

Actividad	Duración (horas/hombre)	Predecesora
33	120	32

Tabla 4. Calendario de actividades de la 3º fase.

Actividad	Duración (horas/hombre)	Predecesora
34	96	33

Tabla 5. Calendario de actividades de la 4º fase.

Actividad	Duración (horas/hombre)	Predecesora
35	5	34
36	5	35

Tabla 6. Calendario de actividades de la 5º fase.

Actividad	Duración (horas/hombre)	Predecesora
37	10	36
38	4	37
39	25	38
40	4	39

Tabla 7. Calendario de actividades de la 6º fase.

El diagrama de Gantt que se generó usando Microsoft Project 2013 está guardado dentro de la carpeta "Diagramas/Diagrama de Gantt" en el CD que contiene esta memoria.

Recursos:

En un principio se iban a dividir los recursos necesarios en categorías utilizando la técnica de Hughes y Cotterell [10], pero basta con dividir los recursos en dos categorías con sus subcategorías para exponer todos los recursos necesarios. Estas dos grandes categorías serán: recursos Hardware y recursos Software, dividiendo estos últimos en Elaboración del código, Elaboración de elementos gráficos, Elaboración de informes y documentos y Seguridad.

- Hardware: Material informático para la realización del proyecto: En él se incluyen los ordenadores utilizados para el desarrollo del código y la memoria y los dispositivos móviles utilizados en las pruebas.

Nombre	MacBook Pro (Retina 13'')
Características técnicas	SO: El Capitán 10.11.4 Procesador: 2.6GHz Intel Core i5 Memoria: 8GB Almacenamiento: 125 GB
Disponibilidad durante el proyecto	100%

Tabla 8. Descripción del recurso hardware MacBook Pro.

Nombre	Sony Vaio
Características técnicas	SO: Windows 8.1 Procesador: 2.5GHz Intel Core i5 Memoria: 6GB Almacenamiento: 750 GB
Disponibilidad durante el proyecto	100%

Tabla 9. Descripción del recurso hardware Sony Vaio.

Nombre	Samsung Galaxy Tab 3
Características técnicas	Número de Modelo: SM-T310 Versión de Android: 4.4.2 Memoria: 1.5 GB Almacenamiento: 16 GB Tamaño de la pantalla: 8''
Disponibilidad durante el proyecto	100%

Tabla 10. Descripción del recurso hardware Samsung Galaxy Tab 3.

Nombre	Samsung Galaxy Tab 4
Características técnicas	Número de Modelo: SM-T230 Versión de Android: 4.4.2 Memoria: 1.8 GB Almacenamiento: 8 GB Tamaño de la pantalla: 7''
Disponibilidad durante el proyecto	No se asegura la total disponibilidad del dispositivo

Tabla 11. Descripción del recurso hardware Samsung Galaxy Tab 4.

Nombre	Dispositivo de almacenamiento externo
Características técnicas	Marca: Kingston Modelo: DTSE9 Capacidad: 16 GB
Descripción del recurso	Utilizado para mantener copias de seguridad de código y documentos
Disponibilidad durante el proyecto	100 %

Tabla 12. Descripción del recurso hardware dispositivo de almacenamiento.

Recurso añadido el 10 de abril de 2016:

Nombre	BQ Aquaris M5
Características técnicas	Número de Modelo: Aquaris M5 Versión de Android: 6.0.1 Memoria: 2 GB Almacenamiento: 16 GB Tamaño de la pantalla: 5''
Disponibilidad durante el proyecto	Total disponibilidad a partir del 10 de abril de 2016

Tabla 13. Descripción del recurso hardware BQ Aquaris M5.

- Software: Programas y servicios utilizados tanto para la elaboración y mantenimiento de la aplicación como para la memoria.

  Elaboración del código:

Nombre	Netbeans
Características técnicas	Versión: 8.0.2 Build 201411181905 Versión Java: 1.8.0_40
Descripción del recurso	Entorno de desarrollo para elaborar la parte Java de la aplicación
Disponibilidad durante el proyecto	100%

Tabla 14. Descripción del recurso software Netbeans.

Nombre	Android Studio
Características técnicas	Versión: 1.5.1 Build #AI-141.2456560 (1 Diciembre 2015)
Descripción del recurso	Entorno de desarrollo para elaborar la parte Android de la aplicación
Disponibilidad durante el proyecto	100%

Tabla 15. Descripción del recurso software Android Studio.

Elaboración de los elementos gráficos de la aplicación:

Nombre	Paint
Características técnicas	
Descripción del recurso	Se ha utilizado para la creación de figuras hexagonales y los prototipos
Disponibilidad durante el proyecto	100%

Tabla 16. Descripción del recurso software Paint.

Nombre	lunapic
Características técnicas	Servicio Web: [11]
Descripción del recurso	Utilizado para crear las fichas del juego
Disponibilidad durante el proyecto	Depende de la estabilidad del servidor en el que se encuentra

Tabla 17. Descripción del recurso software lunapic.

Nombre	makeappicon
Características técnicas	Servicio Web: [12]
Descripción del recurso	Utilizado para crear los diferentes tamaños del icono de la aplicación
Disponibilidad durante el proyecto	Depende de la estabilidad del servidor en el que se encuentra

Tabla 18. Descripción del recurso software makeappicon.

Elaboración de todo tipo de informes y documentos de seguimiento:

Nombre	Google Docs
Características técnicas	Servicio Web: [13]
Descripción del recurso	Para crear documentos en línea y poder acceder a ellos desde cualquier dispositivo
Disponibilidad durante el proyecto	Depende de la estabilidad del servidor en el que se encuentra

Tabla 19. Descripción del recurso software Google Docs.

Nombre	Google Sheets
Características técnicas	Servicio Web: [14]
Descripción del recurso	Para crear hojas de cálculo que servirán para controlar las partidas jugadas contra la IA
Disponibilidad durante el proyecto	Depende de la estabilidad del servidor en el que se encuentra

Tabla 20. Descripción del recurso software Google Sheets.

Nombre	Microsoft Office Word 2007
Características técnicas	Procesador de texto
Descripción del recurso	Utilizado para la creación y formateo de la memoria
Disponibilidad durante el proyecto	100 %

Tabla 21. Descripción del recurso software Microsoft Office Word 2007.

- Seguridad: Para evitar cualquier incidente que pueda provocar la pérdida de una parte o incluso de la totalidad del trabajo realizado se han llevado a cabo las siguientes medidas de seguridad:

Nombre	Github
Características técnicas	Servicio Web: [15]
Descripción del recurso	Se utilizarán repositorios privados para llevar un control de versiones de la aplicación
Disponibilidad durante el proyecto	Depende de la estabilidad del servidor en el que se encuentra

Tabla 22. Descripción del recurso software GitHub.

Nombre	Google Drive
Características técnicas	Servicio Web: [16]
Descripción del recurso	Servicio de alojamiento de archivos para crear copias de seguridad tanto del código como de los documentos
Disponibilidad durante el proyecto	Depende de la estabilidad del servidor en el que se encuentra

Tabla 23. Descripción del recurso software Google Drive.

Nombre	Dropbox
Características técnicas	Servicio Web: [17]
Descripción del recurso	Servicio de alojamiento de archivos al que se suben automáticamente las capturas de pantalla hechas en los dispositivos móviles
Disponibilidad durante el proyecto	Depende de la estabilidad del servidor en el que se encuentra

Tabla 24. Descripción del recurso software Dropbox.

2.4.2 Plan de control

Control de requisitos:

Si una vez acabada y revisada la documentación de análisis, surgiera durante las fases posteriores la necesidad de añadir algún requisito, se estudiará la importancia de este, y si fuera un requisito valioso, se añadirá al análisis y, por lo tanto, al proyecto.

Control del calendario:

Para cumplir las fechas del calendario del proyecto se analizará cada sábado lo conseguido esa semana y se propondrán objetivos para la siguiente. Si en una de esas revisiones se viera que no se va a cumplir la fecha establecida, se añadirán más horas de trabajo a la semana.

Modificación del 15 de marzo de 2016: El día de la semana en que se revisarán los avances y se propondrán objetivos a corto plazo pasará a ser el domingo debido al cambio total del horario.

Control de recursos:

No se prevén muchos problemas en el desgaste de los recursos porque se tienen muchos planes alternativos a estos:

- Si los ordenadores sufrieran algún percance y dejarán de funcionar, se dispone de más tanto en el laboratorio de la universidad como en casa.
- Si todos los dispositivos móviles quedaran inoperativos, se pediría a familiares y amigos el préstamo de estos.
- Todos los servicios online que se utilizan pueden ser sustituidos por programas que no requieren conexión:
    - Google Docs por Microsoft Office
    - Servicios de almacenamiento en la nube por dispositivos físicos de almacenamiento.

2.4.3 Plan de gestión de riesgos

En esta sección se van a exponer los riesgos que se pueden encontrar a lo largo de la elaboración del proyecto y la estrategia para combatirlos en caso de que ocurran.

Título	Baja del trabajador
Fase	Todas las fases
Descripción	Baja indefinida durante un periodo indefinido de tiempo por causas ajenas al proyecto
Probabilidad	Media
Consecuencias	Actividades previstas durante el tiempo que dure la baja del trabajador quedarán sin realizar
Estrategia	Transferir
Plan de acción	Repartir las horas perdidas a otros días laborales

Tabla 25. Descripción del plan de riesgo: Baja del trabajador.

Título	Falta de conexión a Internet
Fase	Todas las fases
Descripción	No disponer de acceso a Internet
Probabilidad	Media
Consecuencias	No se podrá acceder a los servicios web que se utilizan en cada fase
Estrategia	Transferir
Plan de acción	Recurrir a las alternativas Off-line de estos o posponer su utilización hasta que vuelva la conexión

Tabla 26. Descripción del plan de riesgo: Falta de conexión a Internet

Título	Pérdida o mal funcionamiento de algún equipo
Fase	Todas las fases
Descripción	Imposibilidad de seguir trabajando con alguno de los equipos informáticos por algún problema con este o extravío del mismo
Probabilidad	Baja
Consecuencias	Imposibilidad de trabajar con determinado sistema operativo o ciertos programas propios de él
Estrategia	Transferir
Plan de acción	Utilizar otro equipo que se encuentre tanto en el laboratorio de la universidad como en casa (sólo dispositivos)

Tabla 27. Descripción del plan de riesgo: Pérdida de un equipo.

Título	Experiencia insuficiente durante la creación de la aplicación
Fase	Desarrollo
Descripción	Insuficiente conocimiento sobre determinados aspectos de la aplicación. No saber cómo abarcar problemas, sobre todo con la parte gráfica.
Probabilidad	Alta
Consecuencias	Retraso en el calendario establecido
Estrategia	Evitar y Reducir
Plan de acción	Buscar opiniones en sitios web expertos o consultar con personas con más experiencia en el campo que provoca el retraso

Tabla 28. Descripción del plan de riesgo: Experiencia insuficiente.

Título	Retraso en el calendario elaborado
Fase	Todas
Descripción	No haber cumplido los objetivos propuestos en un periodo de fechas
Probabilidad	Alta
Consecuencias	Afecta al resto del calendario
Estrategia	Transferencia
Plan de acción	Replanificar la fase siguiente para no desplazar el desfase de fechas al calendario completo

Tabla 29. Descripción del plan de riesgo: Retraso en el calendario.

2.5 Planes de procesos técnicos

2.5.1 Modelo del proceso

El modelo del proceso a seguir tiene las siguientes partes:

- **Plan de proyecto**: Creación de este tema para tener un guión en el que apoyarse a lo largo de la creación del trabajo.
- **Análisis**: Definición de requisitos y casos de uso.
- **Diseño**: Elaborar y organizar la arquitectura de la aplicación.
- **Implementación**: Desarrollar el código que consiga crear la aplicación propuesta.
- **Pruebas**: Realización de pruebas usando dispositivos móviles para comprobar que la aplicación funciona según lo acordado.

Estas fases pueden tener múltiples iteraciones, pero solo se va a realizar una de cada si todo va según lo planeado. Las últimas dos fases pueden sufrir varias iteraciones, porque al encontrar fallos durante la fase de pruebas se volverá a la fase de implementación/desarrollo para solventarlos.

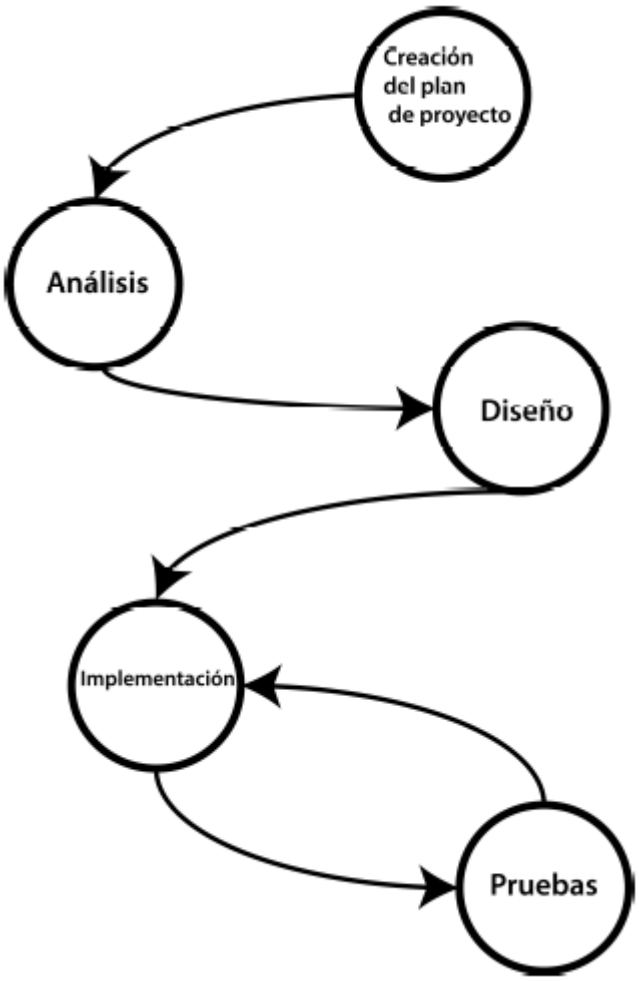

Figura 1. Fases del modelo de proceso.

2.4.2 Plan de despliegue

Al tratarse de un Trabajo de Fin de Grado no se seguirá el proceso normal de despliegue que tiene una aplicación para dispositivos móviles. Cuando se quiere distribuir una aplicación de este tipo se sube a una plataforma de distribución digital de aplicaciones, como puede ser Google Play para aplicaciones Android o App Store para aplicaciones iOS. En este caso se pondrá a disposición de los interesados en este proyecto tanto el código fuente completo con el que, mediante una plataforma de desarrollo, pueden crear un archivo .apk, como el archivo .apk directamente.

3. Análisis

3.1 Introducción

El Sistema a construir deberá seguir una serie de requisitos impuestos tanto por la lógica de las reglas de juego como por el propio autor del Trabajo de Fin de Grado. Se estudiaron minuciosamente las reglas del juego de distintas fuentes [18] - [20] y se analizaron los pasos que debe llevar a cabo un jugador a lo largo de una partida para después transformar estos en Requisitos Funcionales, Requisitos No Funcionales y Casos de Uso.

3.1.1 Visión General del Producto

El producto es una aplicación de entretenimiento para dispositivos Android en el que se ha implementado un juego de mesa llamado Hive (La Colmena) en el que intervienen 2 jugadores (ni más ni menos) y un conjunto de fichas en forma hexagonal que representan diferentes insectos y que compondrán cuando se junten, la colmena, es decir, el tablero de juego estará compuesto por las propias fichas.

El usuario podrá elegir entre jugar una partida contra otro usuario que se encuentre físicamente junto al dispositivo móvil o echar una partida contra la Inteligencia Artificial implementada en la aplicación.

3.1.2 Objetivos del Sistema

Este sistema tiene dos objetivos principales:

1. Permitir al usuario jugar tantas partidas como desee contra otro jugador.

2. Permitir al usuario jugar tantas partidas como desee contra la máquina.

Como objetivo secundario tiene que permitir al usuario leer las reglas del juego, que incluyen como interactuar con el sistema.

3.1.3 Características del Usuario

El juego original solo tiene una restricción respecto a qué personas está orientado el juego y es que deben ser mayores de 9 años. Al implementar este juego en un dispositivo Android ese rango debería mantenerse, pero obviamente se reduce a usuarios que dispongan dicho dispositivo.

3.2 Catálogo de requisitos del Sistema

A continuación, se van a mostrar empleando tablas los requisitos funcionales, no funcionales y de información y descripción de casos de uso que van a ayudar a modelar el sistema a implementar.

3.2.1 Requisitos Funcionales

Los requisitos funcionales son aquellos que especifican que va a ofrecer el sistema. Se muestran a continuación:

FRQ - 0001	Iniciar partida contra otro jugador
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	Ninguna
Descripción	El sistema deberá permitir al usuario iniciar una partida contra otro jugador

Tabla 30. Requisito funcional: Iniciar partida contra otro jugador.

FRQ - 0002	Iniciar partida contra la máquina
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	Ninguna
Descripción	El sistema deberá permitir al usuario jugar una partida contra otro jugador

Tabla 31. Requisito funcional: Iniciar partida contra la máquina.

FRQ - 0003	Partida contra otro jugador
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	Ninguna
Descripción	El sistema deberá permitir al usuario jugar una partida contra otro jugador

Tabla 32. Requisito funcional: Partida contra otro jugador.

FRQ - 0004	Partida contra la máquina
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	Ninguna
Descripción	El sistema deberá permitir al usuario jugar una partida contra la máquina

Tabla 33. Requisito funcional: Partida contra la máquina.

FRQ - 0005	Añadir una ficha
Autor	Lucía Gil Román
Fuente	Lucía Gil Román y Reglas del Juego [ENLACE ANEXO]
Dependencias	FRQ - 0001, FRQ - 0002
Descripción	El sistema deberá permitir al usuario iniciar añadir una ficha al tablero

Tabla 34. Requisito funcional: Añadir una ficha.

FRQ - 0006	Mover una ficha
Autor	Lucía Gil Román
Fuente	Lucía Gil Román y Reglas del Juego [ENLACE ANEXO]
Dependencias	FRQ - 0001, FRQ - 0002
Descripción	El sistema deberá permitir al usuario mover una ficha que ya está en juego

Tabla 35. Requisito funcional: Mover una ficha.

FRQ - 0007	Cancelar la selección de una ficha
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	FRQ - 0001, FRQ - 0002, FRQ - 0004
Descripción	El sistema deberá permitir al usuario cancelar la selección de una ficha (las fichas se seleccionan para ver los posibles movimientos de esta)

Tabla 36. Requisito funcional: Cancelar selección de una ficha.

FRQ - 0008	Cancelar la acción de añadir una ficha
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	FRQ - 0001, FRQ - 0002, FRQ - 0003
Descripción	El sistema deberá permitir al usuario cancelar la acción de añadir una ficha a la partida

Tabla 37. Requisito funcional: Cancelar acción de añadir una ficha.

FRQ - 0009	Ver los posibles movimientos de una ficha
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	FRQ - 0001, FRQ - 0002
Descripción	El sistema deberá permitir al usuario ver los posibles movimientos de una ficha al seleccionarla

Tabla 38. Requisito funcional: Ver posibles movimientos de una ficha.

FRQ - 0010	Ver las reglas del juego
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	
Descripción	El sistema deberá permitir al usuario ver las reglas del juego

Tabla 39. Requisito funcional: Ver las reglas del juego.

FRQ - 0011	Acceder al menú principal
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	FRQ - 0001, FRQ - 0002, FRQ - 0008
Descripción	El sistema deberá permitir al usuario acceder al menú principal de la aplicación

Tabla 40. Requisito funcional: Acceder al menú principal.

FRQ - 0012	Abandonar una partida
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	FRQ - 0001, FRQ - 0002
Descripción	El sistema deberá permitir al usuario abandonar una partida

Tabla 41. Requisito funcional: Abandonar una partida.

3.2.2 Definición de Actores

Los actores son aquellas personas o sistemas que van a interactuar con el sistema desarrollado. En este caso se han identificado dos: El usuario y la inteligencia artificial:

ACT-001	Usuario Principal
Autor	Lucía Gil Román
Fuentes	Instrucciones del juego [18]
Descripción	Este actor representa al usuario que interactúa con la aplicación a través de un dispositivo móvil
Comentarios	

Tabla 42. Definición de actor: Usuario.

ACT-002	Usuario Secundario
Autor	Lucía Gil Román
Fuentes	Instrucciones del juego [18]
Descripción	Este actor representa al usuario que no inicia la partida contra otro jugador pero es uno de los dos jugadores
Comentarios	

Tabla 43. Definición de actor: Inteligencia artificial.

3.2.3 Casos de Uso del Sistema

Los casos de uso sirven para especificar una serie de pasos que permiten realizar alguna acción disponible por el sistema. En ellos se relata tanto el flujo normal que permite realizar la acción, como los pasos alternativos en caso de que no se siga ese flujo. A continuación se muestran los casos de uso:

UC-0001	Iniciar una partida contra otro jugador	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias		
Actores	ACT-001	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera iniciar una partida contra otro jugador	
Precondición	El sistema se ha iniciado y el actor se encuentra en el menú principal	
Secuencia Normal	Paso	Acción
	1	El actor solicita al sistema iniciar una partida contra otro jugador
	2	El sistema prepara el tablero de juego y elige qué jugador empieza primero
Postcondición	El actor ha iniciado una partida contra otro jugador	
Excepciones	Paso	Acción

Tabla 44. Definición del caso de uso: Iniciar partida contra un jugador.

UC-0002	Iniciar una partida contra la máquina	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias		
Actores	ACT-001	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera iniciar una partida contra la máquina	
Precondición	El sistema se ha iniciado y el actor se encuentra en el menú principal	
Secuencia Normal	Paso	Acción
	1	El actor solicita al sistema iniciar una partida contra la máquina
	2	El sistema prepara el tablero de juego y elige qué jugador empieza primero
Postcondición	El actor ha iniciado una partida contra la máquina	
Excepciones	Paso	Acción

Tabla 45. Definición del caso de uso: Iniciar partida contra la máquina.

UC-0003	Jugar una partida contra otro jugador	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias	UC-0001	
Actores	ACT-001, ACT-002	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario juegue una partida contra otro jugador	
Precondición	El usuario inició una partida contra otro jugador	
Secuencia Normal	Paso	Acción
	1	El actor (ACT-001) realiza una acción de las permitidas
	2	El sistema pinta de nuevo el tablero de juego
	3	El actor (ACT-002) realiza una acción de las permitidas
	4	El sistema pinta de nuevo el tablero de juego y el caso de uso vuelve al Paso 1.
Postcondición	Se ha completado una ronda de la partida	
Excepciones	Paso	Acción
	1,2	Si el usuario solicita añadir una ficha, se activa el caso de uso (UC-005) "Añadir una ficha al tablero"
	1,2	Si el usuario solicita mover una ficha, se activa el caso de uso (UC-006) "Cambiar una ficha de posición"
	1,2	Si el usuario solicita abandonar la partida, se activa el caso de uso (UC-007) "Abandonar una partida"
	2,4	Si la partida ha terminado, el sistema muestra el ganador y pide confirmación al usuario. El caso de uso queda sin efecto.

Tabla 46. Definición del caso de uso: Jugar una partida contra otro jugador.

UC-0004	Jugar una partida contra la máquina	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias	UC-0002	
Actores	ACT-001	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario juegue una partida contra la máquina	
Precondición	El usuario inició una partida contra la máquina	
Secuencia Normal	Paso	Acción
	1	El actor (ACT-001) realiza una acción de las permitidas
	2	El sistema realiza sus cálculos y pinta el tablero con las jugadas del actor (ACT-001) y la máquina. El caso de uso vuelve al Paso 1.
Postcondición	Se ha completado una ronda de la partida	
Excepciones	Paso	Acción
	1,2	Si el usuario solicita añadir una ficha, se activa el caso de uso (UC-005) "Añadir una ficha al tablero"
	1,2	Si el usuario solicita mover una ficha, se activa el caso de uso (UC-006) "Cambiar una ficha de posición"
	1,2	Si el usuario solicita abandonar la partida, se activa el caso de uso (UC-007) "Abandonar una partida"
	2,4	Si la partida ha terminado, el sistema muestra el ganador y pide confirmación al usuario. El caso de uso queda sin efecto.

Tabla 47. Definición del caso de uso: Jugar una partida contra la máquina.

UC-0005	Añadir una ficha al tablero	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias	UC-0003, UC-0004	
Actores	ACT-001, ACT-002	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera poner una ficha en juego	
Precondición	El usuario está jugando una partida contra otro jugador o contra la máquina	
Secuencia Normal	Paso	Acción
	1	El actor selecciona una posición del tablero
	2	El sistema muestra una lista de fichas que puede añadir al juego
	3	El actor selecciona qué ficha quiere poner en juego
	4	El sistema añade la ficha en la posición seleccionada en el Paso 1
Postcondición	El actor ha añadido una ficha al juego	
Excepciones	Paso	Acción
	1,3	Si el usuario solicita cancelar la operación, el sistema la cancela, a continuación este caso de uso queda sin efecto.
	1	Si el usuario selecciona una posición ocupada por una ficha enemiga, el sistema sigue mostrando las posiciones donde posicionar una ficha, a continuación este caso de uso continúa en el Paso 1.
	1	Si el usuario selecciona una posición ocupada por una ficha amiga, si su la abeja está en juego y la ficha seleccionada se puede mover el sistema muestra posiciones en las que se puede mover esa ficha, a continuación el caso de uso queda sin efecto.
	1	Si el usuario selecciona una posición ocupada por una ficha amiga, si su abeja no está en juego o la ficha no se puede mover el sistema muestra posiciones en las que se puede mover esa ficha, a continuación el caso de uso queda sin efecto.
	1	Si el usuario selecciona cualquier zona que no tenga forma de hexágono, el sistema sigue mostrando las posiciones donde posicionar una ficha, a continuación este caso de uso continúa en el Paso 1.

Tabla 48. Definición del caso de uso: Añadir una ficha al juego.

UC-0006	Cambiar una ficha de posición	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias	UC-0003, UC-0004	
Actores	ACT-001, ACT-002	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera mover una pieza en juego	
Precondición	El usuario está jugando una partida contra otro jugador o contra la máquina	
Secuencia Normal	Paso	Acción
	1	El actor (ACT-001) selecciona una ficha del tablero
	2	El sistema muestra espacios a los que se puede mover esa ficha
	3	El actor (ACT-001) selecciona un hueco al que mover la ficha
	4	El sistema mueve la ficha a la posición seleccionada en el Paso 3
Postcondición	El actor (ACT-001) ha movido una ficha del juego	
Excepciones	Paso	Acción
	1,3	Si el usuario solicita cancelar la operación, el sistema la cancela, a continuación este caso de uso queda sin efecto.
	1	Si el usuario selecciona una ficha que no se puede mover, el sistema sigue mostrando (si hay) posiciones en la que se puede poner una ficha en juego, a continuación el caso de uso continúa en el Paso 1.
	1	Si el usuario selecciona una posición ocupada por una ficha enemiga, el sistema sigue mostrando las posiciones donde posicionar una ficha, a continuación este caso de uso continúa en el Paso 1.
	1	Si el usuario selecciona una posición en la que se puede añadir una ficha en vez de seleccionar una ficha, el sistema muestra qué fichas puede añadir (si las hay), a continuación el caso de uso queda sin efecto.

Tabla 49. Definición del caso de uso: Cambiar una ficha de posición.

UC-0007	Abandonar una partida	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias	UC-0001, UC-0002, UC-0005	
Actores	ACT-001	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera abandonar una partida en curso	
Precondición	El sistema inició una partida contra otro jugador o contra la máquina	
Secuencia Normal	Paso	Acción
	1	El actor solicita al sistema abandonar la partida actual
	2	El sistema solicita confirmación de la acción
	3	El actor confirma que quiere abandonar la partida
	4	El sistema devuelve al usuario al menú principal de la aplicación
Postcondición	El sistema inició el menú principal	
Excepciones	Paso	Acción
	2	Si el usuario solicita cancelar la operación, el sistema la cancela, a continuación este caso de uso queda sin efecto.

Tabla 50. Definición del caso de uso: Abandonar una partida.

UC-0008	Ver las reglas del juego	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias		
Actores	ACT-001	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera ver las reglas de juego	
Precondición	El sistema se ha iniciado y el actor se encuentra en el menú principal	
Secuencia Normal	Paso	Acción
	1	El actor solicita al sistema ver las reglas de juego
	2	El sistema mostrará por pantalla las reglas de juego
Postcondición	El actor (ACT-001) ha visto las reglas de juego	
Excepciones	Paso	Acción

Tabla 51. Definición del caso de uso: Ver reglas del juego.

UC-0009	Acceder al menú principal	
Autor	Lucía Gil Román	
Fuentes	Lucía Gil Román	
Dependencias		
Actores	ACT-001	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario quiera acceder al menú principal	
Precondición	El usuario debe de estar viendo las reglas del juego	
Secuencia Normal	Paso	Acción
	1	El actor solicita volver al menú principal
Postcondición	El sistema inició el menú principal	

Tabla 52. Definición del caso de uso: Acceder al menú principal.

3.2.3 Diagrama de Casos de Uso

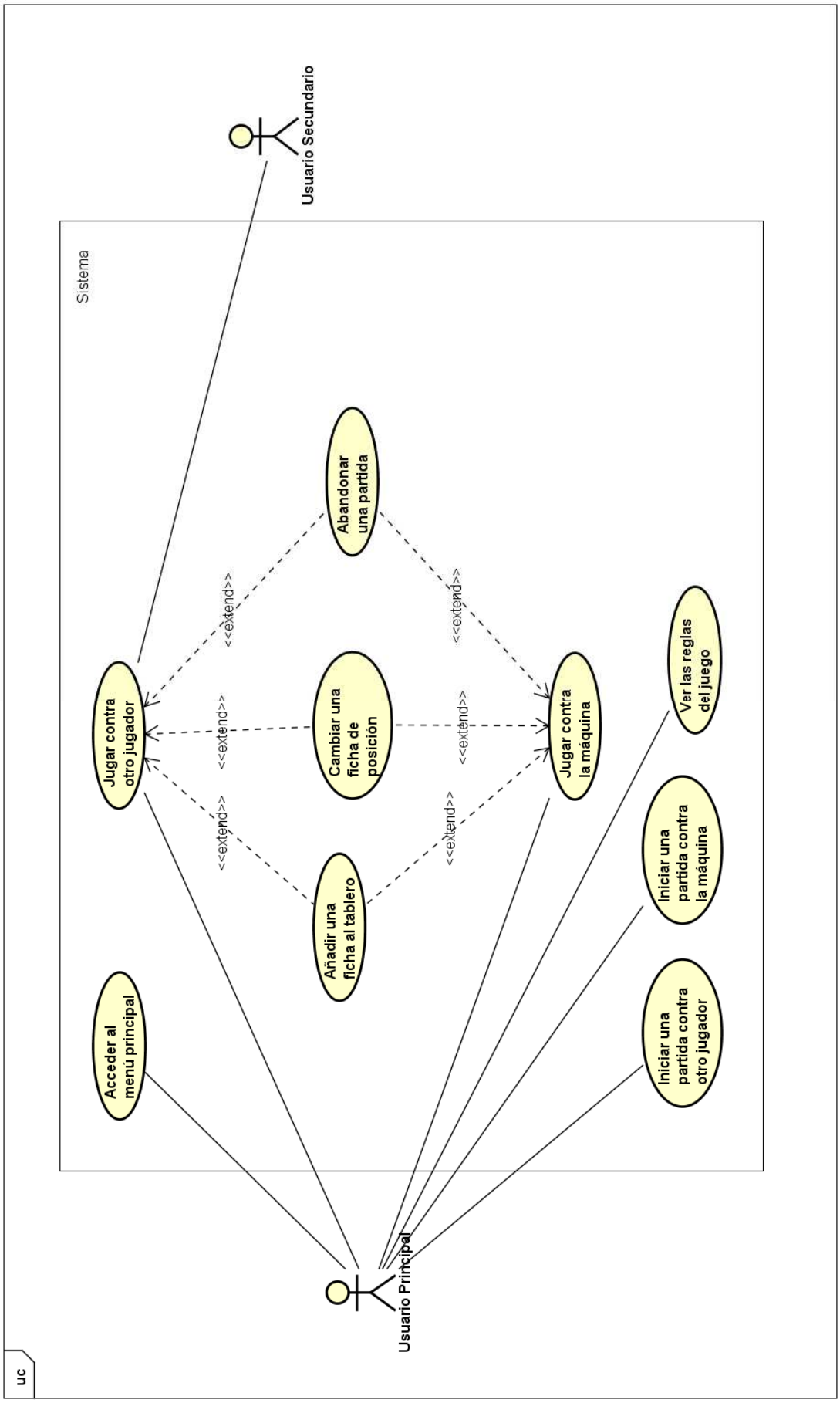

Figura 2. Diagrama de casos de uso.

3.2.4 Requisitos No Funcionales

Los requisitos no funcionales especifica criterios que se usan para calificar la operación del sistema.

NFR-0001	Tecnología
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	Ninguna
Descripción	El sistema deberá diseñarse para dispositivos Android

Tabla 53. Definición del requisito no funcional: Tecnología.

NFR-0002	Versión Android
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	Ninguna
Descripción	El dispositivo deberá tener como mínimo la versión de Android 4.1.2 (API 16) y como mucho Android 6.0 (API 23)

Tabla 54. Definición del requisito no funcional: Versión Android.

NFR-003	Idioma
Autor	Lucía Gil Román
Fuente	Lucía Gil Román
Dependencias	Ninguna
Descripción	El sistema deberá mostrar información en inglés o en español en función del idioma del dispositivo

Tabla 55. Definición del requisito no funcional: Idioma.

3.2.4 Requisitos de Información

Los requisitos de información son formas especializadas de requisitos que indican qué información deberán tener determinados aspectos del sistema. En este caso sólo se necesita información de los usuarios que juegan partidas y de las fichas que estos tienen:

IRQ-0001	Características de las fichas
Autor	Lucía Gil Román
Fuente	Reglas del Juego y Reglas del Juego [ENLACE ANEXO]
Dependencias	Ninguna
Descripción	El sistema deberá almacenar información correspondiente a las fichas
Datos específicos	• Color • Tipo de insecto • Estado: Si se encuentra o no en juego • Valor: Para ayudar a la inteligencia artificial a tomar decisiones. • Coordenadas: Para situar la ficha en el tablero imaginario.

Tabla 56. Definición del requisito de información: Ficha.

IRQ-0002	Características de los jugadores
Autor	Lucía Gil Román
Fuente	Reglas del Juego
Dependencias	Ninguna
Descripción	El sistema deberá almacenar información correspondiente a los jugadores
Datos específicos	• Color • Fichas: Un jugador tiene asignado 11 fichas del mismo color: 1 Abeja 3 Hormigas 3 Saltamontes 2 Arañas 2 Escarabajos

Tabla 57. Definición del requisito de información: Jugador.

3.3 Modelo del dominio

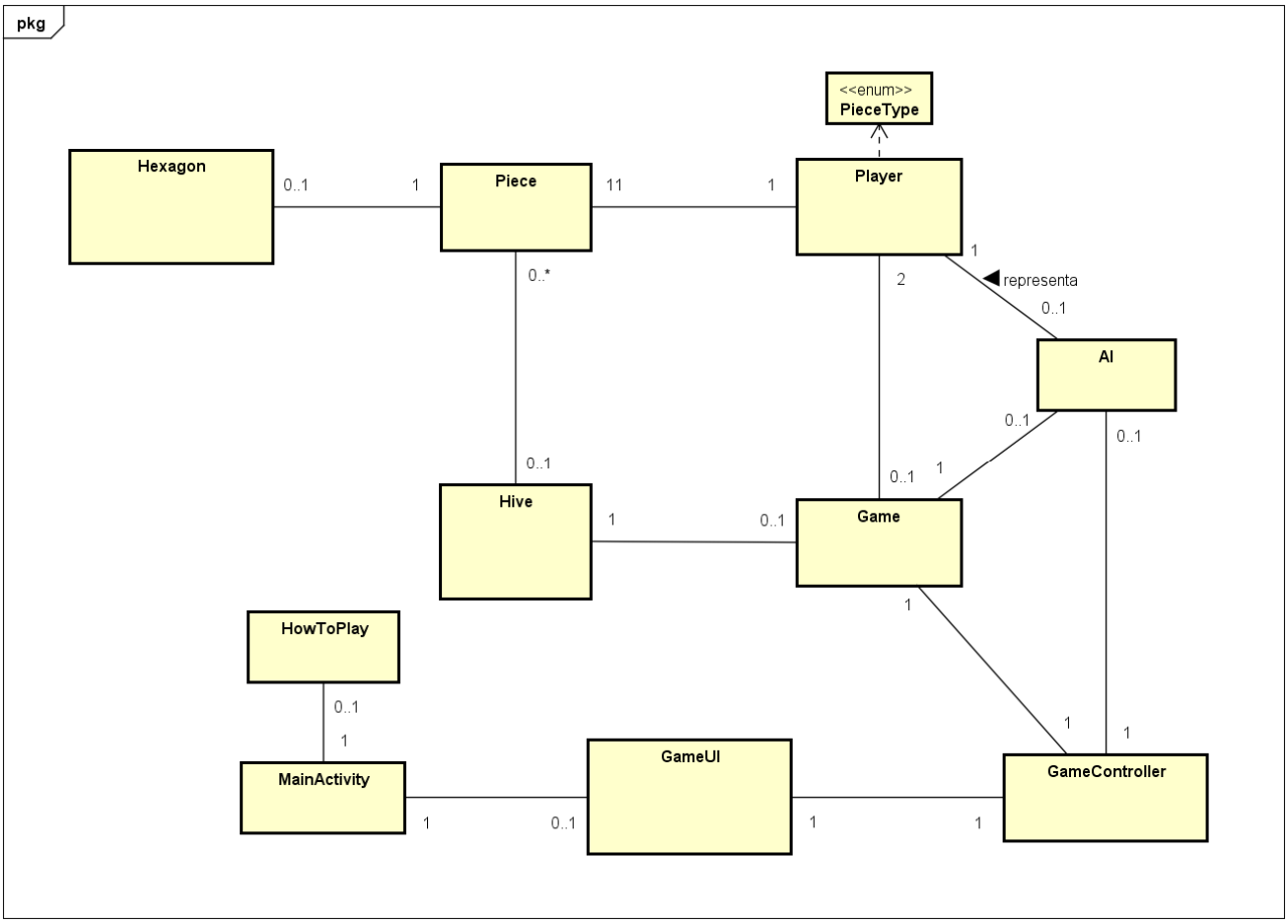

Figura 3. Diagrama del modelo del dominio.

4. Diseño

4.1 Introducción

Este capítulo ampliará lo expuesto en el tema dedicado al análisis del sistema para poder especificar más a fondo el contenido de los paquetes y las clases que forman la aplicación.

El tema se ha dividido principalmente en 3 partes, pues corresponden a los 3 pasos que se han seguido para crear la aplicación, empezando por la lógica del Juego lo que llega a ser equivalente al modelo, siguiendo por la Interfaz de Usuario y acabando por la Inteligencia Artificial.

Para conseguir una visión más detallada se utilizarán a lo largo de este tema prototipos de la aplicación, descripción a fondo de clases, diagramas de paquetes y diagrama completo de clases.

4.2 Tecnologías propuestas

Para elegir el lenguaje en que se iba a programar la parte lógica de la aplicación no hubo que deliberar mucho, sería en Java, por ser el lenguaje sobre el que más conocimiento se tenía y, por lo tanto, no se añadiría a la complejidad del proyecto la falta de aptitudes en el lenguaje elegido.

Sin embargo, a la hora de elegir el lenguaje para desarrollar la interfaz de usuario para las partidas surgieron muchas dudas. Para empezar es un campo sobre el que se tenía muy poca experiencia, como mucho el adquirido en la creación de páginas web o en aplicaciones que recibían valores de entrada por medio de formularios, por lo tanto, se buscaría en repositorios proyectos con características similares y se adaptarían según la necesidad. El lenguaje para desarrollar el tablero de juego no se decidió en la fase de diseño, pero se priorizó la búsqueda de proyectos Android, creados usando Java y el conjunto de herramientas de desarrollo SDK, por ser la combinación más común a la hora de desarrollar proyectos Android y, a la vez, poseer experiencia en esa clase de proyectos. .

Por último, cabe destacar que se optó por programar todo el código únicamente en inglés para facilitar la comprensión del mismo a un mayor número de desarrolladores.

4.3 Modelo arquitectónico

Para mantener organizadas y aisladas las partes en las que se divide la aplicación, se ha utilizado el patrón modelo-vista-controlador (MVC), y cada una de ellas realizará las siguientes funciones:

- **Modelo**: Guardará la lógica del juego. Por lo tanto, tiene la información de los elementos del juego y cómo deben actuar estos.

- **Vista**: Muestra al usuario datos en respuesta a la información que éste solicite por pantalla.

- **Controlador**: Pasa información entre la vista y el modelo.

Para identificar estas partes se han utilizado paquetes. El modelo abarcará tanto el paquete que contiene la lógica del juego como el que contiene la inteligencia artificial, y la vista el paquete que tiene exclusivamente la parte de interfaz de usuario y el paquete que realiza ajustes en la red que forma el tablero de juego. El controlador solo tendrá un paquete con una clase que controlará, como se ha mencionado antes, el paso entre las clases de la vista y las del modelo.

4.4 Prototipos

Para diseñar el prototipo de la aplicación se intentó tener en cuenta aplicaciones del mismo tipo dadas las limitaciones que suponía no tener conocimientos de diseñador gráfico. . Se optó por un diseño muy básico y simple, suficiente para poder interactuar a través de él de una forma agradable e intuitiva.

Por ello se decidió que para la funcionalidad principal de la aplicación, que es jugar, se necesitaba simplemente una malla formada por hexágonos regulares que hicieran de tablero de juego (Figura 4).

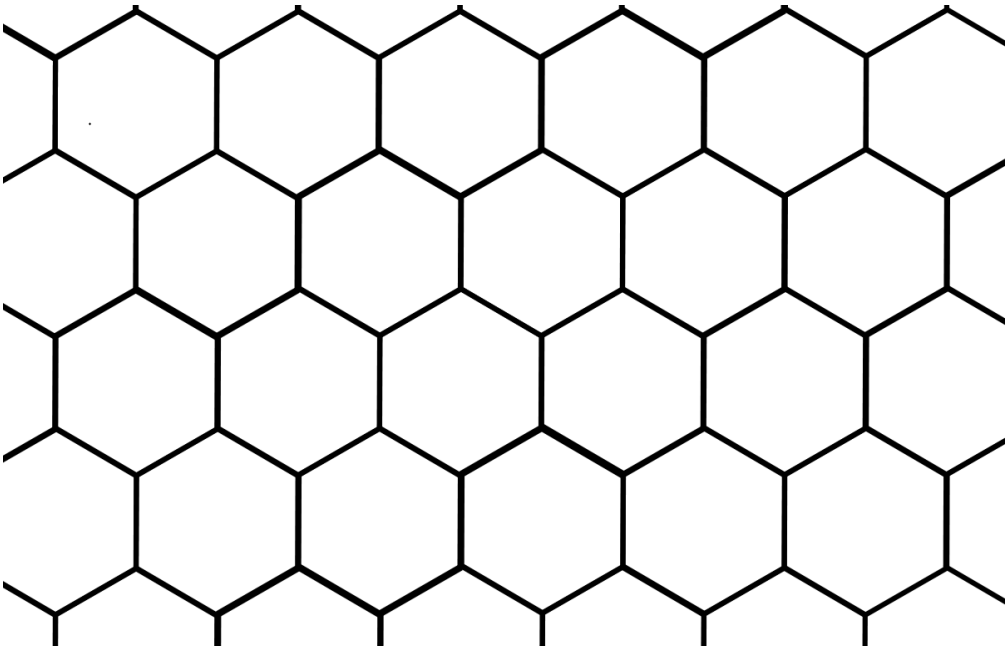

Figura 4. Malla de hexágonos regulares. [21]

Sobre dicho panel de hexágonos se irían poniendo o moviendo las piezas del juego, por lo que se necesitaba una forma de conseguir esto. Se optó por sombrear hexágonos en los que al pulsar se desplegara una lista con los nombres de las fichas disponibles para ser añadidas al juego. Para mover una ficha se decidió que al ser pulsada se sombrearan hexágonos a los que esta podría desplazarse.

El siguiente paso era diseñar las fichas del juego. La forma más simple de realizar esto era colorear hexágonos de blanco o negro y escribir el nombre de los insectos en ellas. Pero esto creaba un problema, la aplicación va a estar en dos idiomas, español e inglés, por lo que habría que dibujar todas las fichas dos veces traduciendo el nombre del insecto para que se mostrara la ficha adecuada dependiendo del idioma del dispositivo. Lo más eficiente y gráficamente agradable habría sido dibujar los insectos como vienen en las fichas originales (Figura 5), pero se eligió la primera opción porque el tiempo que llevó crearlas fue mínimo. La creación de fichas con los insectos dibujados se pospondría, si hubiera tiempo, una vez estuviera funcionando la aplicación entera correctamente o para una supuesta segunda versión.

Se intentó seguir la estética original del juego poniendo el nombre del insecto del mismo color que viene el insecto dibujado (Figura 6).

Figura 5. Fichas originales que representan una hormiga.

Figura 6. Representación de la hormiga negra en inglés y en español.

Una vez se había decidido cómo se iba a diseñar la interfaz había que decidir el diseño del resto de la aplicación. Para ello se decidió que al iniciar la aplicación el usuario pudiera ver un menú con el que acceder a las diferentes opciones de juego (contra otro jugador o contra la máquina) y ver las reglas del juego. También se debería visualizar el logo de la aplicación y el nombre de la aplicación. La organización de esos tres elementos (menú, logo y título) se muestra en la Figura 7.

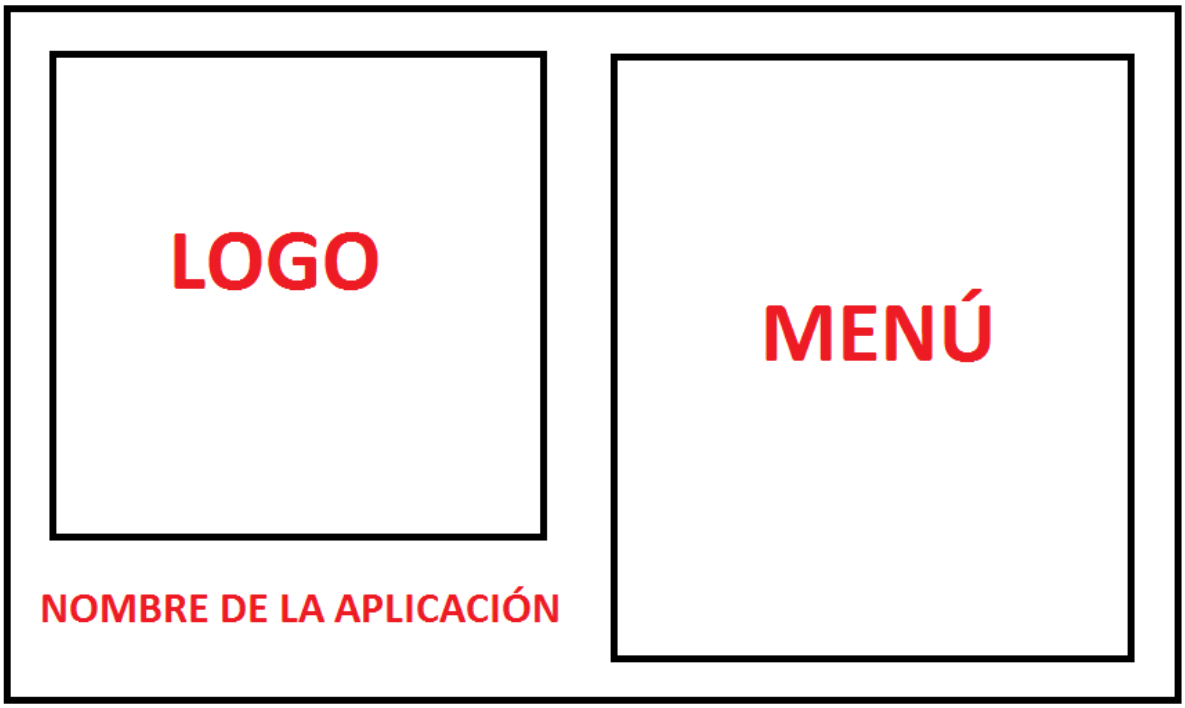

Figura 7. Prototipo para la ventana principal.

Para el logo de la aplicación se utilizó una imagen que se encontró accidentalmente mientras se buscaba dibujos de colmenas o mallas hexagonales. Ver Figura 8.

Figura 8. Logo para la aplicación.[22]

Como el logo estaba pintado utilizando únicamente dos colores (blanco y amarillo) se optó por utilizar dichos colores como el tema visual de la aplicación, siendo el fondo de las partidas en amarillo y el fondo del menú y las reglas del juego en blanco.

Para obtener el nombre de la aplicación se optó por el nombre original del juego "Hive".

Por último, había que mostrar las complejas reglas del juego al usuario de una forma lo más sencilla posible de manera que este pudiera entender tanto las reglas del juego como la forma en que había que añadir y mover fichas usando esta aplicación. Para ello se ideó mostrar las reglas del juego como pares imagen-texto, de forma que la imagen mostrara lo que el texto estaba explicando, así ante un repaso rápido de las reglas solo habría que ver las imágenes. Ver Figura 9.

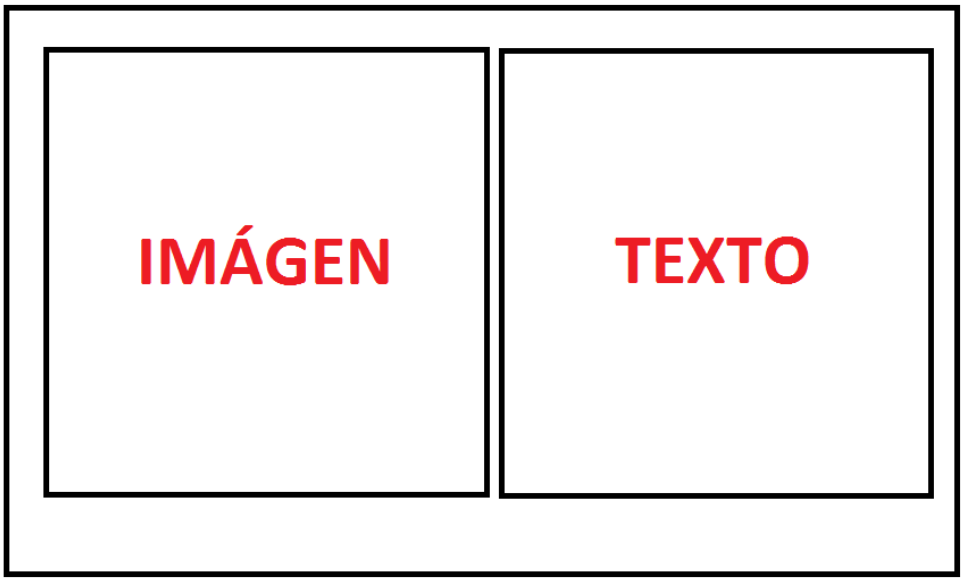

Figura 9. Prototipo para la pantalla de las reglas del juego.

Para pasar a ver la siguiente regla el usuario solo tendría que tocar el texto, mientras que para ver la regla anterior tendría que tocar la imagen.

4.5 Diseño detallado

Este apartado fue elaborado una vez se disponía de la aplicación completamente acabada y funcionando. Por lo tanto, está pensado para explicar en profundidad las clases que componen cada paquete y cómo se idearon determinadas partes.

4.5.1 Lógica del Juego

Observando las reglas del juego [ANEXO I] y echando unas cuantas partidas se llega a la conclusión de que existen principalmente los siguientes elementos:

- Juego
- Tablero (1) , inexistente al principio de la partida pero que crece según van pasando las rondas, por lo que merece el nombre de colmena (Hive)
- Fichas (22)
- Jugadores (2)

Luego fue fácil deducir que dichos elementos se convertirían en clases para poder así añadir atributos y métodos para formar la lógica del juego.

Piece:

Cuando abres cualquier bolsa de “Hive” para empezar una partida lo único que viene en su interior son 22 fichas blancas y negras con diferentes insectos dibujados en ellas, por lo tanto es la parte más importante del juego y por la que se empieza a idear la aplicación.

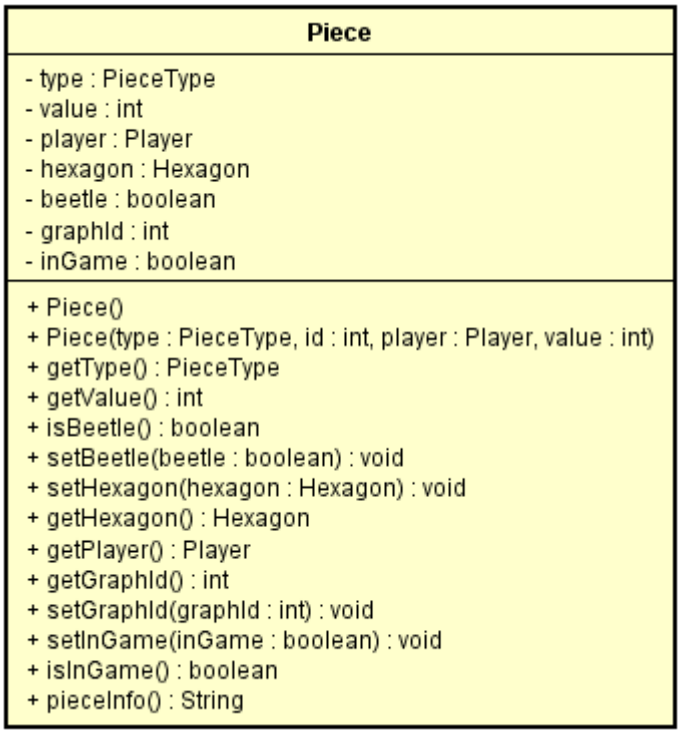

Figura 10. Clase Piece.

La clase Piece (Ficha) tiene los siguientes atributos:

- **Type** (Tipo): De tipo PieceType (Tipo de Ficha), que sirve para identificar el tipo de ficha que es, es decir, qué clase de insecto representa. En la clase PieceType se puede ver cómo se han definido como constantes los cinco tipos de fichas que existen: BEE (Abeja), GRASSHOPPER (Saltamontes), BEETLE (Escarabajo), SPIDER (Araña) y ANT (Hormiga).

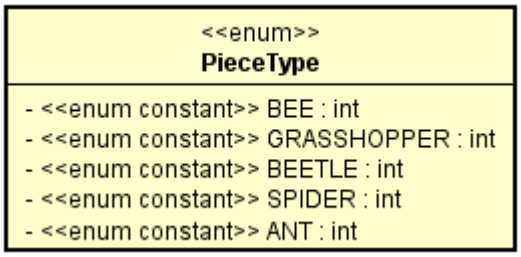

Figura 11. Clase PieceType.

- **Value** (Valor): De tipo entero, utilizado para asignar un valor a la ficha que será utilizado por la Inteligencia Artificial para determinar cuál puede ser su siguiente jugada.

- **Player** (Jugador): De tipo Player (Jugador). Las fichas puede ser blancas o negras, esto quiere decir que pueden pertenecer a un jugador o a otro por lo que en realidad las fichas están asignadas a un jugador.

- **Hexagon** (Hexágono): Cuando la ficha se ponga en juego estará asignada a una posición dada por (Columna, fila, nivel). Dicho trío está representado por la clase Hexagon (Hexágono) que explicaremos más adelante.

- **Beetle** (Escarabajo): De tipo booleano, utilizado para indicar si esa ficha tiene o no un escarabajo encima de ella. Obviamente, esto sólo puede pasar si la ficha está en juego.

- **graphId** (Id en el grafo): De tipo entero y que sirve para indicar el número del nodo al que pertenece la ficha en el grafo que forma el tablero de juego (Esta parte se ampliará en la explicación de la clase Hive (Colmena)).

- **inGame** (En Juego); De tipo booleano y que sirve para especificar si la ficha está o no en juego.

La clase Piece(Ficha) tiene los siguientes métodos:

- **Public Piece()**: Constructor para crear una ficha sobre la que se desconocen todos los datos.

- **Public Piece(type : PieceType, id : int, player : Player, value : int)**: Constructor para crear una ficha sobre la que se conoce su tipo, a qué jugador pertenece y cuál es su valía.

- **Setters & Getters** de los distintos atributos.

Player:

Cada partida de “Hive” se compone única y exclusivamente de 2 jugadores que se turnan para colocar fichas sobre una superficie preferiblemente lisa.

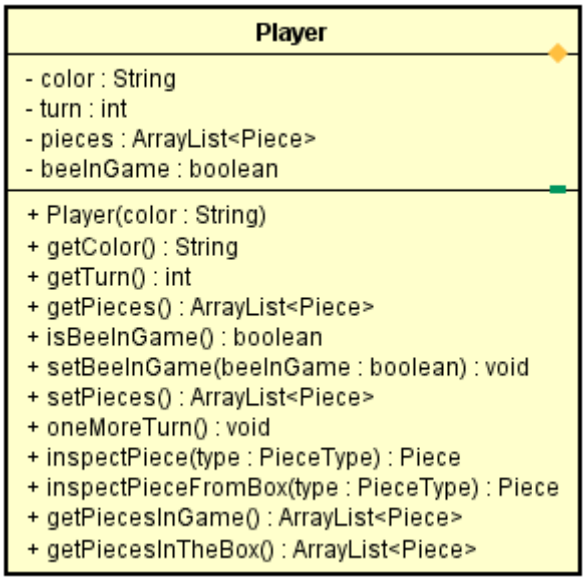

Figura 12. Clase Player.

La clase Player (Jugador) tiene los siguientes atributos:

- **Color**: De tipo String, para identificar al color que representa al jugador.

- **Turn** (Turno): De tipo entero, para especificar durante una partida cuántos turnos ha jugado.

- **Pieces** (Fichas/Piezas): De tipo ArrayList<Piece>, que almacena las 11 fichas que tiene el jugador.

- **beeInGame** (Abeja en Juego): De tipo booleano para indicar si el jugador ha colocado ya o no la abeja en el tablero de juego.

  La clase Player (Jugador) tiene los siguientes métodos:

- **Public Player(color : String)**: Constructor para crear un jugador asignándole un color.

- **Public oneMoreTurn() : void**: Para incrementar en 1 el contador de turnos.

- **Public inspectPiece(type : PieceType) : Piece**: Devuelve una ficha del jugador dado su tipo.

- **Public inspectPieceFromBox(type : PieceType)** : Piece: Devuelve una ficha del jugador que no está en juego dado su tipo.

- **Public getPiecesInGame() : ArrayList<Piece>**: Devuelve la lista de fichas que el jugador tiene en juego.

- **Public getPiecesInTheBox() : ArrayList<Piece>**: Devuelve la lista de fichas que el jugador no ha puesto en juego, por lo tanto, permanecen en lo que he denominado caja (box)

Hive:

La colmena o tablero no existe al iniciar una partida, esta se va construyendo según van los jugadores añadiendo fichas al juego. La colmena imaginaria está compuesta por hexágonos, identificados por columna, fila y nivel como se indica en la Figura 13.

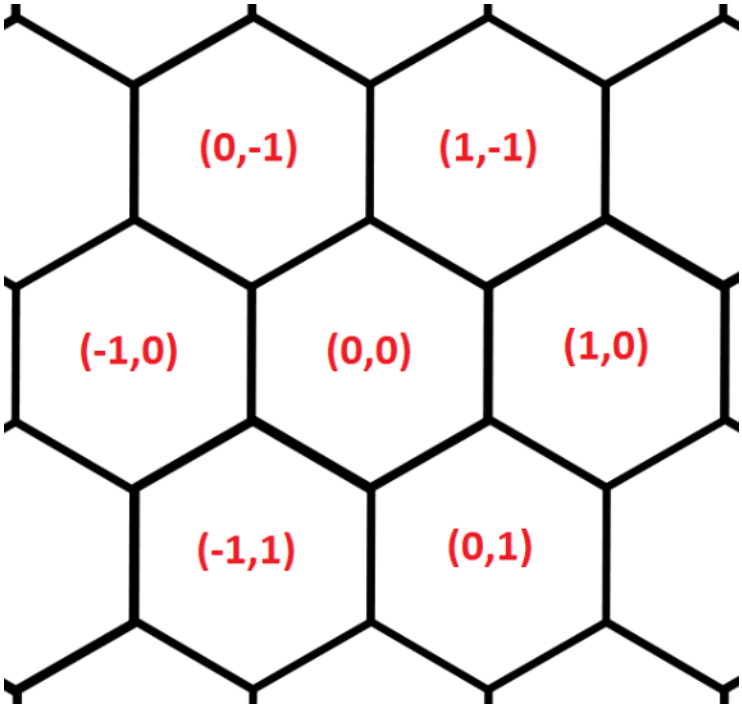

Figura 13. Representación de una ficha en la colmena.

De esta forma el centro de la colmena está indicado en la coordenada (0,0), y el resto de hexágonos se referencia con respecto a esta de la siguiente forma:

- Hexágono superior izquierda: misma columna, una fila menos.

- Hexágono superior derecha: una columna más y una fila menos.
- Hexágono inmediatamente a su izquierda: una columna menos, misma fila.
- Hexágono inmediatamente a su derecha: una columna más, misma fila.
- Hexágono inferior izquierda: una columna menos, una fila de más.
- Hexágono inferior derecha: misma columna, una fila de más.

Así podremos identificar las fichas en el tablero y conocer sus hexágonos vecinos. La tercera coordenada que indica el nivel en el que está la ficha no aparece representado en la imagen, habría que ver dicha imagen en tres dimensiones para poder visualizar los niveles.

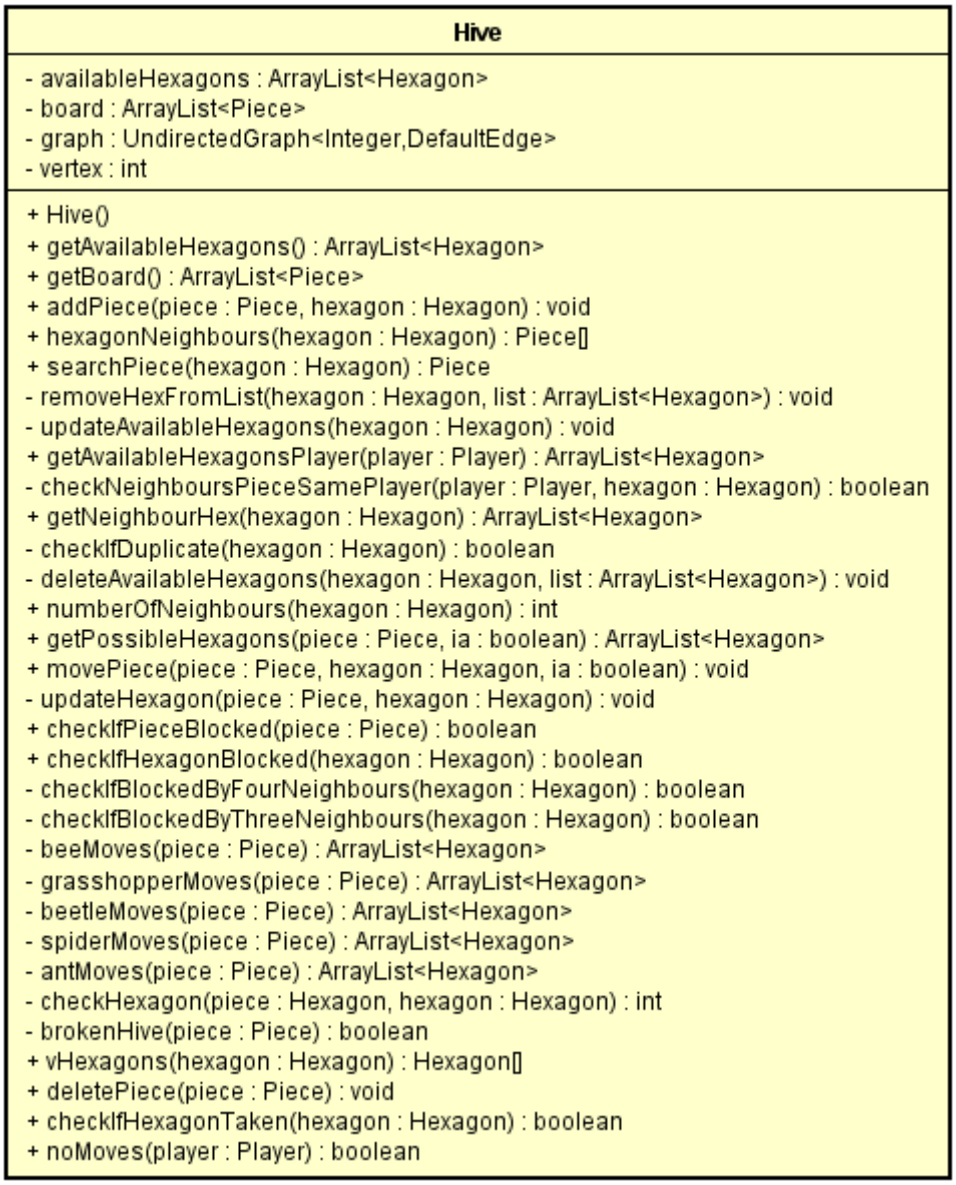

Figura 14. Clase Hive.

La clase Hive(Colmena) tiene los siguientes atributos:

- **availableHexagons** (Hexágonos disponibles): De tipo ArrayList<Hexagon> que servirá como lista para almacenar los hexágonos que estarán disponibles según qué jugador tenga el turno y qué movimientos desea hacer.

- **Board** (Tablero): De tipo ArrayList<Piece> que guardará las fichas que se van añadiendo al tablero.

- **Graph** (Grafo): De tipo UndirectedGraph<Integer,DefaultEdge> obtenida de [23] y que sirve para representar el tablero en forma de grafo.

- **Vertex** (Vértice): De tipo entero, actuará como contador de vértices del grafo.

La clase Hive (Colmena) tiene los siguientes métodos:

- **Public Hive()**: Constructor que crea una colmena con el contador de vértices a 0 y con un único hexágono disponible, el del medio (0,0,0).

- **Public addPiece(piece : Piece, hexagon : Hexagon) : void**: Añade una ficha al tablero en la posición indicada por la variable hexagon. Entre los pasos por los que pasa una ficha para ser añadida al tablero está cambiarle el atributo "hexagon" por la nueva posición, cambiar su estado de "inGame" a verdadero, obtener sus vecinos para añadirlos a la lista de hexágonos disponibles, asignarle un vértice y añadirla al grafo y, por último, eliminar el hexágono que ocupa de la lista de hexágonos disponibles.

- **Public hexagonNeighbours(hexagon : Hexagon) : Piece[]**: Devuelve un Array de tipo Piece (Ficha) con los vecinos que tiene el hexágono pasado como parámetro.

- **Public searchPiece(hexagon : Hexagon) : Piece**: Busca una ficha del tablero dadas las coordenadas del hexágono al que pertenece.

- **Private removeHexFromList(hexagon : Hexagon, list : ArrayList<Hexagon>) : void**: Borra un hexágono dado de la lista proporcionada.

- **Private updateAvailableHexagons(hexagon : Hexagon) : void**: Añade un nuevo hexágono a la lista de hexágonos disponibles si este no se encuentra ya en ella.

- **Public getAvailableHexagonsPlayer(player : Player) : ArrayList<Hexagon>**: Devuelve la lista de hexágonos disponibles en los que el jugador dado puede poner una ficha por primera vez en el tablero.

- **Private checkNeighboursPieceSamePlayer(player : Player, hexagon : Hexagon) : boolean**: Comprueba si los vecinos de un hexágono dado son todos del jugador que se pasa como parámetro.

- **Private getNeighbourHex(hexagon : Hexagon) : ArrayList<Hexagon>**: Devuelve los hexágonos vecinos vacíos del hexágono dado, es decir, si en el hexágono del nivel 0 encuentra una ficha seguirá subiendo niveles hasta encontrar uno vacío.

- **Private checkIfDuplicate(hexagon : Hexagon) : boolean**: Comprueba si el hexágono ya se encuentra en la lista de hexágonos disponibles.

- **Private deleteAvailableHexagons(hexagon : Hexagon, list : ArrayList<Hexagon>) : void**: Borra un hexágono de la lista que se pasa al método si este no tiene vecinos.

- **Public numberOfNeighbours(hexagon : Hexagon) : int**: Devuelve el número de fichas que rodea un hexágono.

- **Public getPossibleHexagons(piece : Piece, ia : boolean) : ArrayList<Hexagon>**: Devuelve los hexágonos a los que se podría mover la ficha según el tipo de esta.

- **Public movePiece(piece : Piece, hexagon : Hexagon, ia : boolean) : void**: Mueve una ficha a la posición especificada por el hexágono.

- **Private updateHexagon(piece : Piece, hexagon : Hexagon) : void**: Asigna a una ficha del tablero una nueva posición.

- **Public checkIfPieceBlocked(piece : Piece) : boolean**: Comprueba si la ficha está bloqueada.

- **Private checkIfHexagonBlocked(hexagon : Hexagon) : boolean**: Comprueba si un hexágono está bloqueado. El hexágono dado puede estar ocupado o no.

- **Private checkIfBlockedByFourNeighbours(hexagon : Hexagon) : boolean**: Comprueba si un hexágono que tiene 4 vecinos estos le están bloqueando.

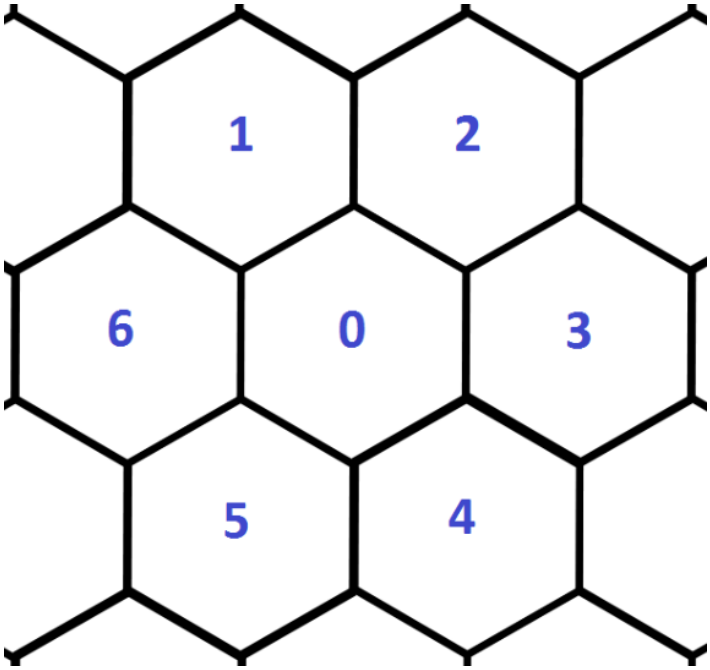

Figura 15. Representación de la colmena con hexágonos numerados.

Si suponemos que en la Figura 15 hay una ficha en el hexágono 0 y está rodeada por cuatro hexágonos, podemos saber si esta está bloqueada si dos hexágonos seguidos están vacíos. Es decir, comprueba la ocupación de los siguientes pares de hexágonos: (1-2), (2-3), (3-4), (4-5), (5-6), (6-1).

- **Private checkIfBlockedByThreeNeighbours(hexagon : Hexagon) : boolean**: Comprueba si un hexágono que tiene 3 vecinos estos le están bloqueando. Observando la Figura 15 podemos suponer que en el hexágono 0 hay una ficha rodeada por otras tres. Esta ficha

estará bloqueada si las fichas que la rodean están todas distanciadas unas de otras por un solo hexágono. Comprobará los siguientes tríos: (1-3-5) y (2-4-6). Si alguno de esos tríos tiene esos tres huecos ocupados, la ficha estará bloqueada.

- **Private beeMoves(piece : Piece) : ArrayList<Hexagon>**: Devuelve los huecos a los que se puede mover una abeja.

- **Private grasshopperMoves(piece : Piece) : ArrayList<Hexagon>**: Devuelve los huecos a los que se puede mover un saltamontes.

- **Private beetleMoves(piece : Piece) : ArrayList<Hexagon>**: Devuelve los huecos a los que se puede mover un escarabajo.

- **Private spiderMoves(piece : Piece) : ArrayList<Hexagon>**: Devuelve los huecos a los que se puede mover una araña.

- **Private antMoves(piece : Piece) : ArrayList<Hexagon>**: Devuelve los huecos a los que se puede mover una hormiga.

- **Private checkHexagon(piece : Hexagon, hexagon : Hexagon) : int**: Devuelve un 1 si la pieza puede deslizarse hacia el hexágono dado. Supongamos que en la Figura 15 hay una ficha en el hexágono 0 Si quiere desplazarse hacia el hexágono 2, deberá comprobar primero que en 2 no hay ya una ficha y luego si hay en los hexágonos 1 y 3. Si hay en ambos, no podrá deslizarse hacía 2, y tampoco podrá hacerlo si no hay fichas en ninguno de los dos pues se estaría separando de la colmena. Por lo tanto, sólo podrá desplazarse hacia el hexágono 2 si hay una ficha en 1 o en 3, pero no en los dos a la vez.

- **Private brokenHive(piece : Piece)** : boolean: Método en el que se utiliza la clase isCutpoint() del paquete JgraphT [] para comprobar si al mover esa ficha se rompería la colmena, es decir, comprueba que el vértice que representa a la ficha en el grafo no parta en dos dicho grafo si ese vértice desaparece.

- **Public vHexagons(hexagon : Hexagon) : Hexagon[]**: Devuelve dos hexágonos que formarían una V con el hexágono dado. Esta función se ideó para trazar los tres primeros movimientos de la inteligencia artificial. Para explicar cómo funciona vamos a suponer, utilizando la Figura 13, que hay una ficha de la inteligencia artificial en el hexágono 0 y una del usuario en 5. Esta función devolverá los hexágonos 1 y 3 como hexágonos en los que la inteligencia artificial colocará sus fichas, formando así una “V” con sus fichas.

- **Public deletePiece(piece : Piece) : void**: Quita una ficha del tablero. Esto no está permitido en las reglas del juego, las fichas que se ponen en juego no se pueden quitar, pero en esta aplicación lo usa la IA para tomar decisiones, es decir, la inteligencia artificial va a poner fichas temporalmente en el tablero para ver las consecuencias de esa acción, pero luego las quita para que el tablero no sufra cambios.

- **Public checkIfHexagonTaken(hexagon : Hexagon) : boolean**: Comprueba si un hexágono dado está ocupado.

- **Public noMoves(player:Player) : boolean**: Comprueba si el jugador que se pasa como parámetro no puede mover ninguna de las fichas que tiene puestas en juego.

Game:

Una partida está compuesta principalmente por dos jugadores y 22 fichas, es todo lo que se necesita en el juego original para poder jugar.

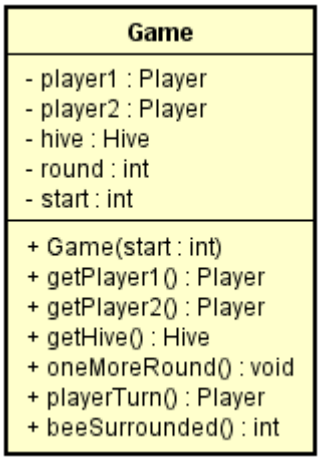

Figura 16. Clase Game.

La clase Game(Juego) tiene los siguientes atributos:

- **Player1** (Jugador 1): De tipo Player, representa al jugador 1.
- **Player2** (Jugador 2): De tipo Player, representa al jugador 2.
- **Hive** (Colmena): De tipo Hive, representa el tablero de juego/colmena.
- **Round** (Ronda): De tipo entero es un contador de rondas.
- **Start** (Inicia): De tipo entero sirve para indicar qué jugador empieza primero.

La clase Game (Juego) tiene los siguientes métodos:

- **Public Game(start : int)**: Constructor que crea un nuevo juego dado un entero que indica qué jugador empieza primero. Por defecto se asigna al jugador 1 el color blanco y al 2 el negro.
- **Public oneMoreRound() : void**: Incrementa la ronda en 1.
- **Public playerTurn() : Player**: Devuelve a qué jugador le toca realizar una acción sobre el juego.
- **Public beeSurrounded() : int**: Devuelve un 3 si las abejas de los 2 jugadores están completamente rodeadas, un 2 si lo está la del jugador 2 y un 1 si lo está la del jugador 1.

Hexagon:

Cuando una ficha se pone en juego debe tener un atributo que la identifique de forma única en el tablero de juego y así poder ser identificada para tenerla en cuenta a la hora de realizar cualquier

movimiento. Por eso se creó la clase Hexágono que ayudará a identificar a una ficha dada su posición en el juego mediante su número de fila, columna y nivel.

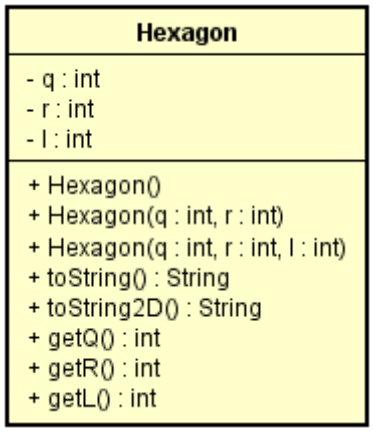

Figura 17. Clase Hexagon.

La clase Hexagon (Hexágono) tiene los siguientes atributos:

- **Q**: De tipo entero, para identificar una columna.
- **R**: De tipo entero, para identificar una fila.
- **L**: De tipo entero, para identificar un nivel.

La clase Hexagon (Hexágono) tiene los siguientes métodos:

- **Public Hexagon()**: Constructor que inicializa las coordenadas a una posición imposible (en principio): (-100,-100,-100)
- **Public Hexagon(q : int, r : int)**: Constructor que inicializa el hexágono a las coordenadas dadas por q y r, y el nivel lo inicializa a 0.
- **Public Hexagon(q : int, r : int, l : int)**: Constructor que crea un hexágono en las coordenadas dadas.
- **Public toString()** : String: Devuelve un cadena con los valores del hexágono.
- **Public toString2D()** : String: Devuelve una cadena con los datos de la columna y fila del hexágono.

4.5.2 Interfaz de Usuario

Una vez se tenía la lógica del juego implementada, se pasó a idear una Interfaz de usuario. Se eligió crear un menú que tuviera tres opciones:

- Echar una partida contra otro jugador.
- Echar una partida contra la máquina.
- Ver las reglas del juego

Para la creación del menú se optó por un código encontrado en un repositorio github [24] que se había utilizado con anterioridad para trabajos propios y había dado muy buenos resultados. El código tenía todo lo que necesitaba: Un simple menú para mostrar las posibles acciones que podía realizar el usuario.

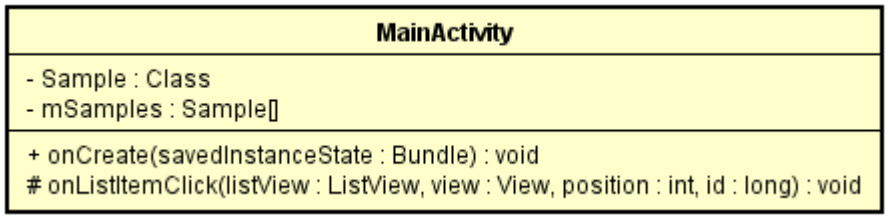

Figura 18. Clase MainActivity.

La clase que implementa el menú principal se llama MainActivity (Actividad Principal) y tiene los siguientes atributos:

- Una clase privada llamada **Sample**: Que tiene un constructor que asocia una actividad con un título y un método que devuelve el título de dicha actividad.

- Un array de tipo Sample llamado **mSample**: Que se utilizará para guardar la lista de actividades.

Esta clase dispone de dos métodos:

- **public onCreate(savedInstanceState : Bundle) : void**: Que inicializa el array con los tres elementos que va a tener el menú principal: Partida contra otro jugador, partida contra la máquina y ver reglas del juego. También inicializa dicha lista con su vista correspondiente.

- **protected onListItemClick(listView : ListView, view : View, position : int, id : long) : void**: Que es llamada cuando se pulsa un elemento de la lista e inicializa dicha actividad. Si la actividad seleccionada es la de echar una partida contra la máquina, añadirá a la llamada de dicha actividad un elemento que indique que la partida es contra la máquina:

```
@Override
protected void onListItemClick(ListView listView, View view, int position, long id) {
    // Launch the sample associated with this list position.
    Intent intent = new Intent(MainActivity.this, mSamples[position].activityClass);
    //If the user clicked the second option --> Game against the AI
    if(position==1) {
        Bundle b = new Bundle();
        b.putBoolean("AI", true);
        intent.putExtras(b);
    }
    startActivity(intent);
}
```

Figura 19. Trozo de código perteneciente a la clase MainActivity.

La siguiente parte consistía en crear una interfaz de usuario para la parte lógica del juego que se explicó en la sección anterior. Como se disponía de muy poco conocimiento en este campo, se optó por buscar en repositorios github trabajos que presentaran las siguientes características:

- El proyecto debería estar escrito en Android, pues se tenían conocimientos suficientes sobre este lenguaje de programación como para entender el código y así poder adaptarlo a nuestras necesidades.

- Debería consistir en una malla de hexágonos, que representaran la colmena a construir en cada partida.

- Cada uno de los hexágonos debería estar identificado de forma única por una columna y una fila.

Tras muchas búsquedas infructuosas, se encontró el siguiente repositorio: [25], que contenía todas las características que se buscaban. El código encontrado se encargaba de crear una malla en forma hexagonal formada por hexágonos en los que había introducido o no una imagen circular. Al pulsar en dicha imagen, se hacía momentáneamente más pequeña, y mostraba por pantalla las coordenadas correspondientes.

En la Figura 20 se puede ver el aspecto que tiene dicha aplicación:

Figura 20. Captura de pantalla de la interfaz elegida [26]

El siguiente paso fue iniciar un nuevo proyecto Android en Android Studio con el código de ese repositorio y empezar a modificarle.

Lo primero que se hizo fue cambiar las imágenes que se mostraban por hexágonos en vez de círculos. Inicialmente se optó por llenar la pantalla de hexágonos color naranja para simular la colmena, hexágonos color gris que indicasen huecos en los que se podía colocar o mover una ficha y hexágonos de color negro o blanco, según el jugador, con los nombres de los insectos escritos en ellos. Ver Figura 21.

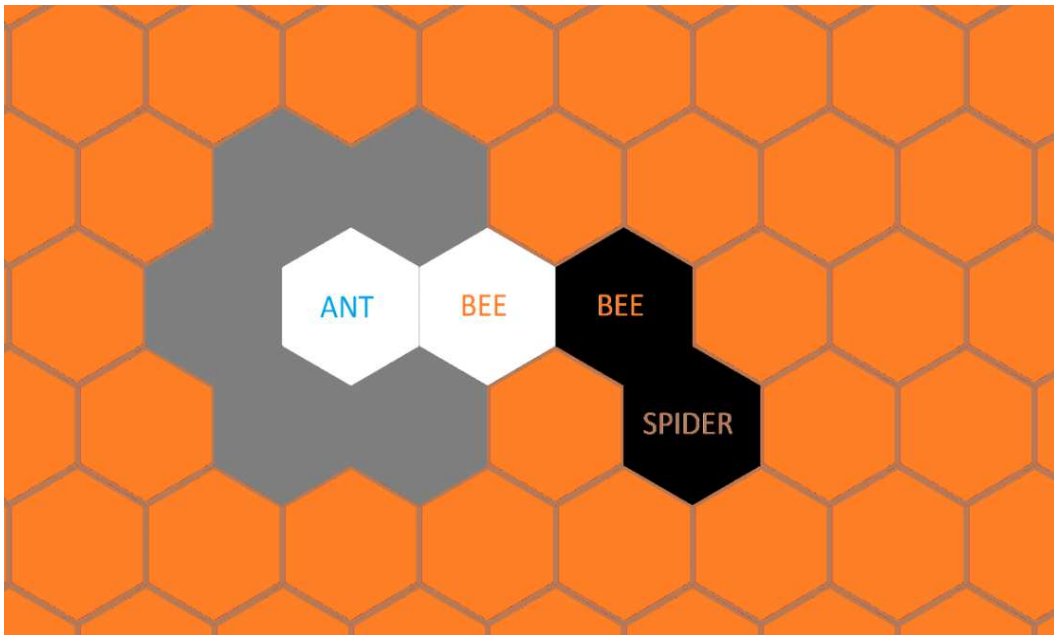

Figura 21. Captura de pantalla del primero diseño de la interfaz.

Esta fue sólo la idea inicial, pues se sabía que el tablero imaginario no tenía un tamaño fijo, podría crecer infinitamente, y que lo más eficiente era hacer que este fuera creciendo según se colocaran fichas como en el juego real.

Por tanto, se optó por mostrar al inicio de cada partida un hexágono gris sobre un fondo amarillento, de modo que el primer jugador colocaría su primera ficha en dicho hexágono y en el siguiente turno se ampliase el número de hexágonos grises para que el siguiente jugador pudiera elegir dónde colocar su primera ficha.

Este nuevo tablero tampoco es infinito, pero tienen un radio de 22 hexágonos por lo que es muy difícil llegar a tocar un extremo, pues formar una línea recta no ayuda a conseguir el objetivo del juego y además llevaría mucho tiempo. En una partida contra la máquina sería imposible alcanzar una esquina por esta misma razón: La máquina juega para ganar, no para buscar los límites del tablero.

A pesar de no solucionar la cuestión del tablero infinito se optó por programar esta última opción porque se gastaban sólo los recursos necesarios, es decir, en vez de tener que pintar innecesariamente un tablero entero como en la idea inicial sólo se van a pintar los hexágonos de las fichas puestas en juego y los hexágonos sombreados según a qué jugador le toque ese turno y qué acciones va a realizar. La Figura 22 muestra la interfaz final de una partida.

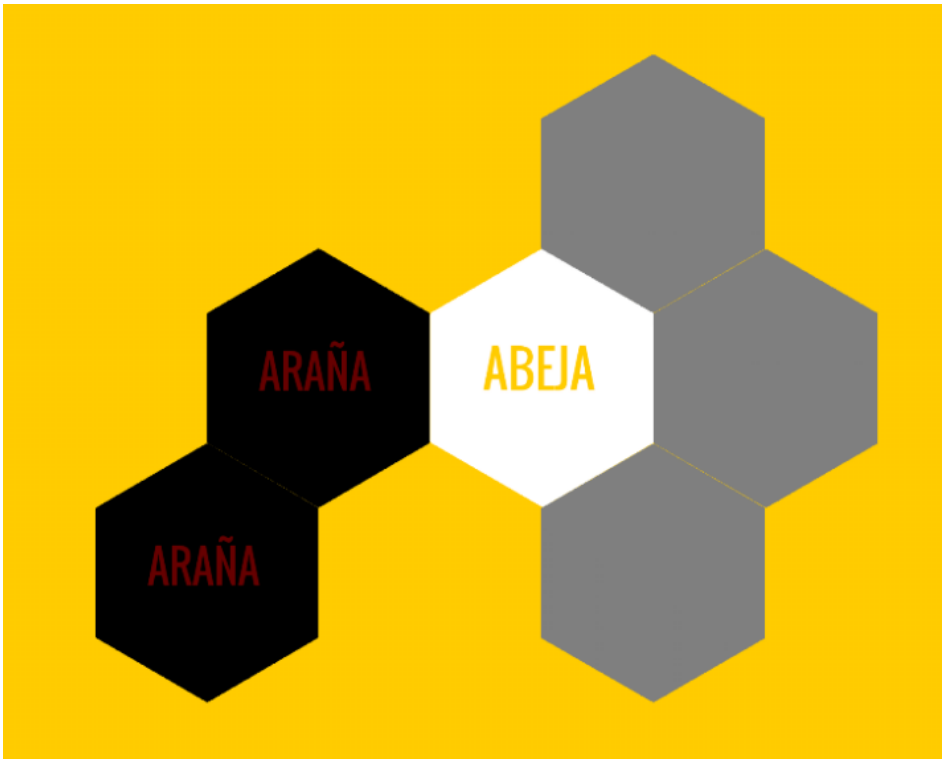

Figura 22. Captura de pantalla del diseño definitivo de la interfaz.

Las clases que se cogieron del repositorio fueron las siguientes:

- Del modelo:
  - Cube
  - Grid
  - Hex
- De la Interfaz de usuario:
  - CircleImageView
  - MainActivity

Para que fueran de utilidad se adaptaron realizando los siguientes cambios:

- **Hex**: Se observó que esta clase tenía casi los mismos atributos y las mismas características que la clase Hexagon creada en el modelo de juego, por lo que se refactorizaron las llamadas a dicha clase para que en vez de llamar a Hex llamara a Hexagon.

- **Cube**: Que es utilizada por la clase Grid (red) para crear la malla de hexágonos. De esta clase se eliminaron métodos que no eran utilizados nunca.

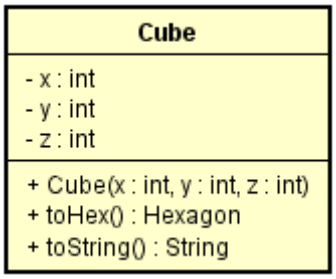

Figura 23. Clase Cube.

- **CircleImageView**: Esta fue la única clase que no sufrió ningun cambio. El código de la aplicación tiene esta clase en su estado original. Es utilizada para dibujar los hexágonos en la red.

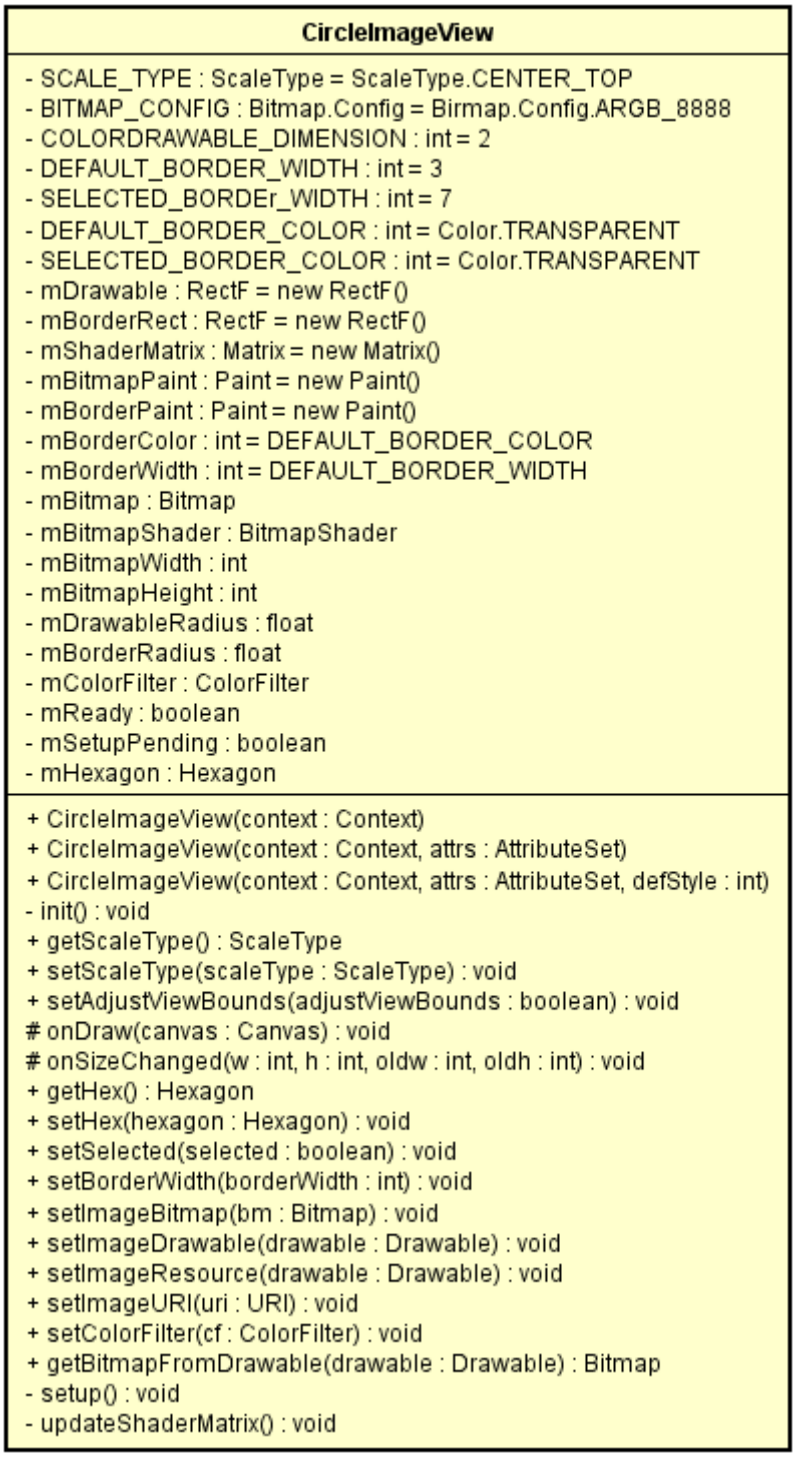

Figura 24. Clase CircleImageView.

Las otras dos clases deben ser explicadas de una forma más amplia y precisa, pues sufrieron muchos cambios para cumplir todas las especificaciones del sistema.

Empezaremos por la clase MainActivity que fue renombrada como GameUI (Interfaz de Usuario del Juego).

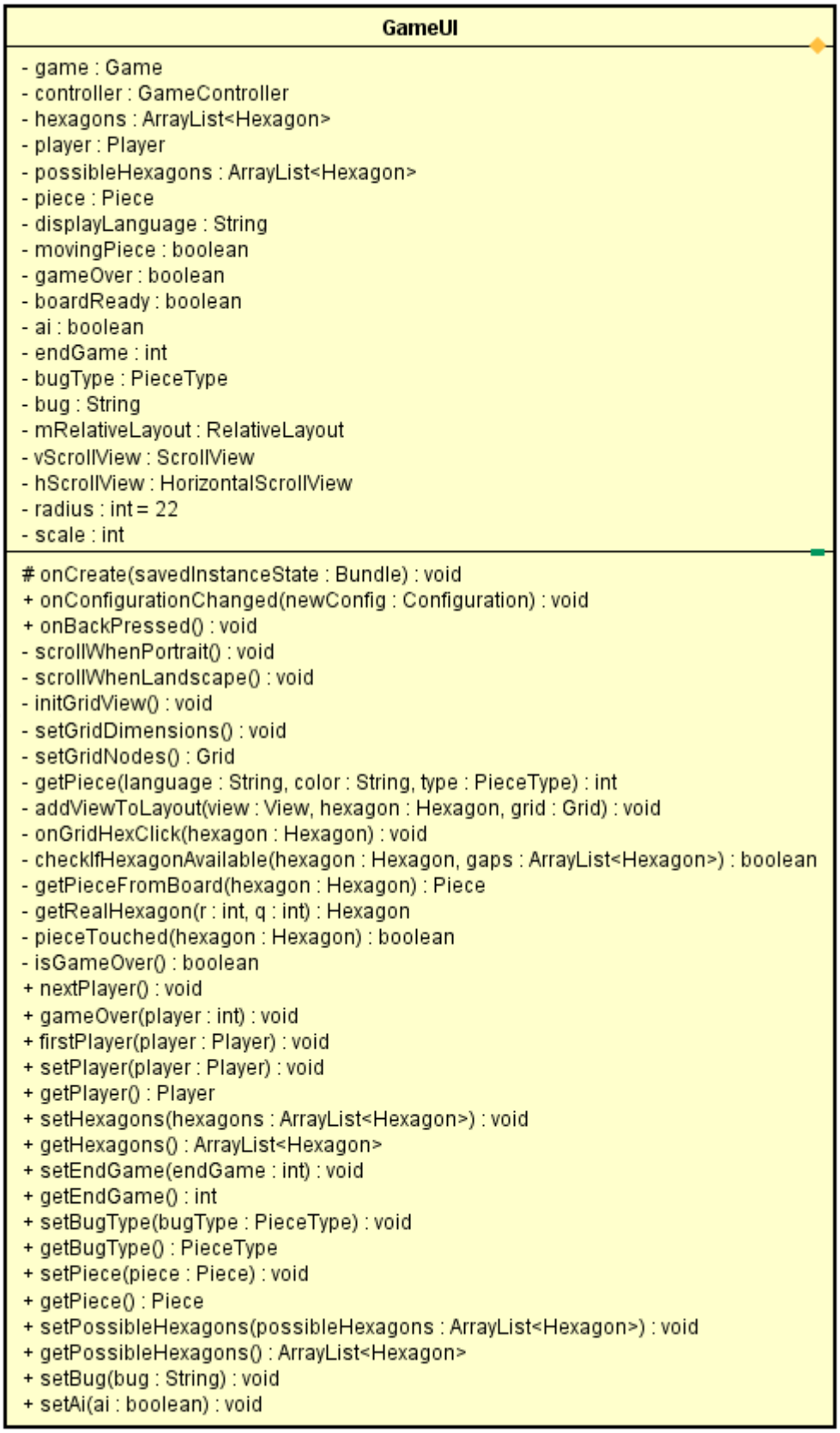

Figura 25. Clase GameUI.

Como la cantidad de atributos y métodos que hay en esta clase es ingente, se van a explicar únicamente los más importantes. Para tener más información sobre todos los atributos y métodos se puede ver el código de la aplicación que está completamente comentado.

- **private setGridNodes() : Grid**: Es el método encargado de dibujar la malla de hexágonos. En él se realizan las siguientes acciones en orden:

    1. Se borra el tablero de juego anterior.
    2. Comprueba si la partida ha acabado: 1 o las 2 abejas fueron totalmente rodeadas en el turno anterior.
    3. Obtiene el jugador al que le toca jugar este turno.
    4. Si no se está realizando la acción de mover una ficha del juego se obtienen los posibles huecos en los que ese jugador puede colocar una ficha por primera vez.
    5. Se realizan distintas comprobaciones de casos excepcionales:
        a. El jugador ya ha puesto todas las fichas en juego.
        b. No tiene hexágonos donde poner fichas nuevas y no tiene la abeja en juego.
        c. Tiene todas las fichas en juego y no puede mover ninguna.
        d. No tiene hexágonos donde poner fichas nuevas y no puede mover las que tiene en juego.
    6. Se obtiene la red a dibujar: Para ello se llama al constructor de la clase Grid que se explicará más adelante.
    7. Se añaden las acciones que se van a realizar cuando se toque la red. Al pulsar un hexágono se asegurará de que toca dentro de él y al levantar el dedo se realizará una acción dependiendo del hexágono que se tocó. Esta última parte se ampliará en el segundo método que se va a explicar sobre esta clase: onGridHexClick(hexagon : Hexagon) : void
    8. Se dibujan los hexágonos.

- **private onGridHexClick(hexagon : Hexagon) : void**: Método al que se llama cuando se pulsa un hexágono de la red. Hace 4 comprobaciones:

    1. Si el booleano que indica que se está moviendo una ficha está como true, y el hexágono al que se quiere mover es un hexágono válido: Mueve la ficha a esa posición, incrementa el turno y la ronda, devuelve el booleano el valor false y llama a setGridNodes() para que dibuje la nueva situación de juego.
    2. Si el booleano que indica que se está moviendo una ficha está a false, y el hexágono que se pulsó estaba en gris: Se añade una ficha al juego, para ello despliega un diálogo en el que el usuario selecciona la pieza que quiere añadir. Añade la ficha, incrementa el turno y la ronda y llama a setGridNodes() para que dibuje la nueva situación de juego.
    3. Si el usuario toca una de sus fichas: Se obtienen los posibles movimientos de esa ficha, se asigna el valor true al booleano que indica si se está moviendo una ficha y se llama a setGridNodes() para que dibuje la nueva situación de juego.
    4. Si se estaba moviendo una ficha, pero se tocó un hexágono del oponente: Se cancela la acción de mover una ficha, se asigna el valor false al booleano que indica si se está moviendo una ficha y se llama a setGridNodes() para que dibuje la nueva situación de juego.

La otra clase modificada fue Grid (Red), pues esta creaba únicamente redes compuestas por hexágonos y en forma hexagonal, mientras que no se necesitaba que la red tuviera ninguna forma específica y, además, sólo se necesitaban dibujar ciertos hexágonos, no toda la red.

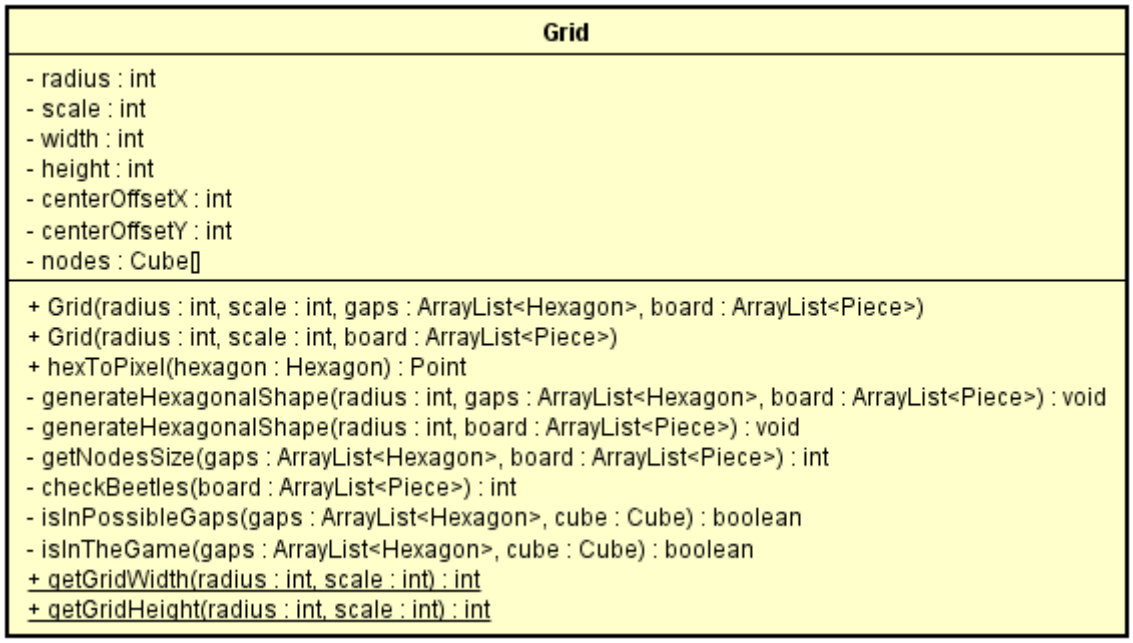

Figura 26. Clase Grid.

Los cambios que se realizaron fueron los siguientes:

- Añadir el atributo board de tipo ArrayList<Hexagon> para guardar los hexágonos que se tenían que dibujar.

- Se modificó el constructor y se crearon dos: Uno para cuando no hay que mostrar hexágonos sombreados y sólo hay que mostrar las fichas que están en juego y otro para cuando hay que mostrar ambos.

- La función que genera el número de hexágonos que hay que dibujar (getNodesSize) generaba hexágonos de acuerdo a una fórmula para crear una red en forma hexagonal. Por lo tanto, se borró su contenido y se añadió código para calcular el número de hexágonos a dibujar en función de las listas que se pasaron al constructor.

- Otra función que se modificó fue (generateHexagonalShape), que era la función que generaba la red hexagonal. Ahora, durante el recorrido en el que genera la red hexagonal, solo añadirá el hexágono a la lista de hexágonos para dibujar, si este se encuentra en alguna de las listas que se pasó al constructor.

Con todos estos cambios y más que se pueden observar en el código, se creó la interfaz de usuario para que este pueda interactuar con el dispositivo y jugar partidas.

La siguiente clase a explicar es HowToPlay (Cómo jugar) que muestra de una forma dinámica cuáles son las reglas del juego y qué se debe hacer para añadir fichas al juego y mover las que ya están sobre el tablero.

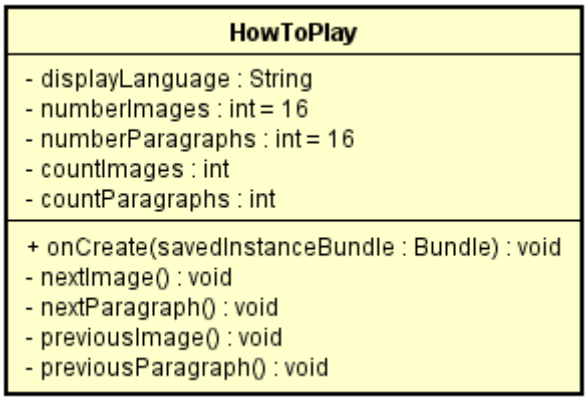

Figura 27. Clase HowToPlay.

Al iniciar la actividad se crean tres mapas:

- Uno para guardar los pares nombre-recurso de los párrafos a mostrar.
- Otro para guardar los pares nombre-recurso de las imágenes en español a mostrar.
- Y por último, otro para guardar los pares nombre-recurso de las imágenes en inglés.

La vista que utiliza esta clase solo tiene dos elementos: Una imagen y un Texto. En dichos elementos se irán cargando los contenidos de los mapas de la siguiente forma:

- Al cargar la actividad, se asignará al elemento imagen el primero del mapa de imágenes según el idioma del dispositivo y el primer párrafo del mapa de párrafos.

- La siguiente “página” de las instrucciones se mostrará cuando se pulse sobre el texto. Esto generará que se cargue en los elementos de la vista la imagen y el párrafo que se encuentran inmediatamente después en los respectivos mapas.

- Para acceder a la “página” anterior se deberá pulsar en la imagen. Al hacer esto se provoca el efecto contrario que al pulsar el texto, se mostrarán la imagen y el texto que se encuentren almacenados en el mapa justo antes de los que se están mostrando actualmente.

Una última clase a explicar es Language (Idioma). Esta clase surgió para solucionar un problema que apareció a la hora de cumplir el requisito de que la aplicación debería estar en inglés y en español. Como el código está escrito en inglés, la clase enum que añade un tipo a la ficha la identifica con su nombre en inglés, entonces al recuperar las fichas del usuario para mostrarlas a este por pantalla solo aparecían en inglés, por lo que se creó esta clase para traducir dicha salida al idioma en que se encontrase el dispositivo.

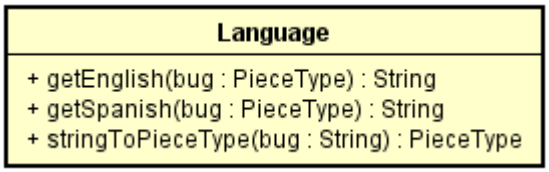

Figura 28. Clase Language.

La clase Language (Idioma) tiene los siguientes métodos:

- **Public getEnglish(bug : PieceType) : String**: Devuelve la traducción en inglés del tipo dado.

- **Public getSpanish(bug : PieceType) : String**: Devuelve la traducción en español del tipo dado.

- **Public stringToPieceType(bug : String) : PieceType**: Devuelve el tipo PieceType que corresponde al tipo pasado como String.

Las clases que se han explicado en esta sección se pueden agrupar en dos grupos: Las que muestran algo al usuario y las que ajustan y preparan el tablero de juego. Por lo tanto se han creado dos paquetes para almacenarlos:

- UI (Interfaz de Usuario)
    - MainActivity
    - GameUI
    - HowToPlay
- BoardSettings (Ajustes del tablero)
    - Cube
    - Grid
    - CircleImageView
    - Language

4.5.3 Inteligencia Artificial

Una vez estaba acabada la aplicación para poder echar partidas contra otros jugadores usando el mismo dispositivo, hubo que empezar a plantearse cómo diseñar la inteligencia artificial para que pudiera jugar con un mínimo nivel de dificultad.

Para preparar esta parte del proyecto se leyó [9] y se estableció contacto con un ex-alumno (Francisco José Presencio) que presentó como trabajo de fin de carrera un proyecto similar: una aplicación Android con inteligencia artificial para jugar partidas de ajedrez con distintos niveles de dificultad. También se habló con la tutora del proyecto sobre cómo abarcar esta parte.

De esta fase de preparación se plantearon dos posibles formas de aplicar la inteligencia artificial:

1. Como un conjunto de reglas que se aplicarían en un determinado orden en función del estado del tablero de juego.

2. Generando un árbol con todas las posibles jugadas a lo largo de determinados turnos y, empleando un algoritmo para encontrar el mejor movimiento posible. Esto sería posible empleando:

    - Minimax: Método que permite obtener la mejor jugada teniendo en cuenta que el contrincante elegirá la jugada que peor nos venga.

    - Poda alfa-beta: Técnica que reduce las ramas a evaluar del árbol, disminuyendo así el tiempo de búsqueda.

De las dos formas se eligió la opción 1: Utilizar un conjunto de reglas. Las razones por las que se eligió esta opción y no la otra fueron las siguientes:

- El tiempo empleado en seguir las reglas no sería estable, es decir, podría tardar muy poco o mucho dependiendo de la jugada, mientras que en la generación del árbol con todas las

jugadas sería siempre estable pero, a la vez, demasiado largo aunque los movimientos a realizar fueran muy obvios.

- La dificultad de la inteligencia artificial usando un árbol variaría con el número de niveles de dicho árbol. Como la aplicación está pensada para dispositivos móviles de distintas características, el árbol a generar no podría ser muy profundo o consumiría mucha batería. Un dispositivo muy potente toleraría bien la creación y búsqueda del árbol pero no uno con poca memoria RAM como suelen ser los teléfonos inteligentes de gama media.

- Se podría trazar una estrategia clara y razonable empleando el conocimiento adquirido de múltiples horas de juego con diferentes tipos de jugadores empleando simplemente reglas.

También se disponía del siguiente artículo para apoyar esta decisión [27]

El siguiente paso fue trazar un plan para que la inteligencia artificial eligiera el mejor movimiento siguiendo las reglas:

- El primer paso fue elegir cómo debería empezar las partidas. Se decidió, por simplicidad y porque era una forma de abrir el juego muy eficiente, seguir las pistas sobre cómo empezar una partida de la página oficial del juego [28]. En dichas pistas se muestran dos formas de realizar las primeras tres jugadas que permiten defender de forma eficiente a la abeja. Por lo tanto, se implementaron esas dos formas de empezar el juego y una forma aleatoria de elegirlas.

- Cuando ya han pasado los tres primeros turnos de la inteligencia artificial deberá empezar a seguir una serie de reglas:

    1. Comprobar el estado de su abeja:

        a. Si la abeja no se puede mover, deberá comprobar si puede mover alguna de las fichas amigas que la rodean.

        b. Si la abeja se puede mover, comprobará el número de vecinos que tiene ella y la abeja enemiga, y en función de esos números tomará una decisión: Si la abeja enemiga tiene menos de cuatro vecinos y la abeja de la inteligencia artificial tiene más de dos, moverá esta para ponerla a salvo.

    2. Si cuando comprobó el estado de la abeja no realizó ningún movimiento ni añadió ninguna ficha, atacará al oponente. Para atacar al contrario seguirá la siguiente lógica:

        a. Si alguna de las fichas que ya está en juego puede colocarse en una posición mejor, moverá esa ficha.

        b. Si ninguna de las fichas del juego puede mejorar su posición, añadirá una ficha nueva.

        c. Si no puede añadir una ficha al juego, moverá una de las que ya están aunque empeore su posición actual. Los jugadores no pueden pasar cuando llega su turno, están obligados a realizar una acción, por lo tanto, aunque empeore su situación actual deberá mover la ficha.

d. Si no ha podido realizar ninguna de las acciones anteriores, significa que no puede ni mover las fichas que están en juego ni añadir nuevas porque ya no le quedan. Es una situación a la que es muy difícil llegar, pero llegado el momento deberá pasar el turno al otro jugador.

La lógica seguida fue esta porque más importante que atacar la abeja enemiga es comprobar que la tuya está lo suficientemente a salvo, de esta forma se puede atacar al enemigo sin preocuparse de la propia abeja. Para atacar al enemigo se eligió mover primero las que están ya en el tablero porque al añadir una ficha no se ataca al enemigo directamente, habría que esperar al siguiente turno, por lo tanto es mejor mover fichas que ya están en juego que pueden atacar al enemigo en un solo turno.

Para comprobar cuál es la mejor posición para mover una ficha se evalúan los posibles destinos. Para ello se comprueba cada una de las fichas que la rodean y según su valor, color y situación de bloqueo se realiza la siguiente operación:

- Las fichas tienen un valor asignado según su importancia:

    - Abeja: 30
    - Saltamontes: 2
    - Araña: 6
    - Escarabajo: 4
    - Hormiga: 8

    - Para deducir qué valores asignar a las fichas se ha utilizado la experiencia en el juego, la abeja tiene la puntuación más alta por ser el objetivo principal del juego atraparla y, por ejemplo, la hormiga tiene la siguiente puntuación más alta porque es una ficha que se puede colocar casi en cualquier posición con un solo movimiento, por lo tanto es más peligrosa que el resto de fichas que no son la abeja y por eso tiene más valor que ellas.

- Si la ficha coincide con el color de las fichas de la inteligencia artificial el valor que tiene asignado dicha ficha pasaría a ser negativo y si fuera del otro jugador se mantendría positivo.

- Por último si al mover la ficha a la nueva posición bloquea otras, el valor de esas fichas se multiplica por dos.

Para aclarar esta parte de cómo evaluar una posición se va a utilizar un ejemplo: supongamos que al mover una ficha negra a un hueco tiene como vecinos a una abeja blanca y dos hormigas negras, una que queda bloqueada y otra no. Al mover la ficha a ese hueco provoca que la abeja blanca quede bloqueada. La función de evaluación sería:

Una abeja de distinto color y bloqueada: su valor en positivo multiplicado por dos = 60
Una hormiga del mismo color y que no queda bloqueada: su valor en negativo = -8
Otra hormiga del mismo color y que queda bloqueada: su valor en negativo por dos = -16
Por último, para obtener el valor de ese movimiento habrá que sumar dichos valores: 60 - 8 - 16 = 36.

Una vez se tienen todos los valores de todos los posibles destinos de la ficha se elegirá aquel que tiene la puntuación más alta y la ficha se moverá a ese lugar.

Una táctica parecida se sigue para comprobar cuál es la mejor posición a la que colocar una ficha por primera vez en juego. Para elegir dónde ponerla se simula que se coloca la ficha en el juego y

entonces se evalúan los posibles movimientos en el siguiente turno de esa ficha de la misma forma que se evalúan las que ya están en juego. Esta parte llevaría mucho tiempo de computación si se tienen que colocar todas las fichas que no están puestas en juego y en todas las posiciones posibles, por lo tanto la mayoría de las veces se elige un tipo de ficha de forma aleatoria para evaluar. Existen casos especiales que se han ido añadiendo según se observaba el comportamiento de la inteligencia artificial en las partidas, como por ejemplo, si tiene que añadir una ficha al juego elegirá un saltamontes si la abeja enemiga tiene algún hueco que la rodea bloqueado, pues dicha pieza puede acceder a huecos bloqueados y otras no.

La clase AI tiene el siguiente aspecto:

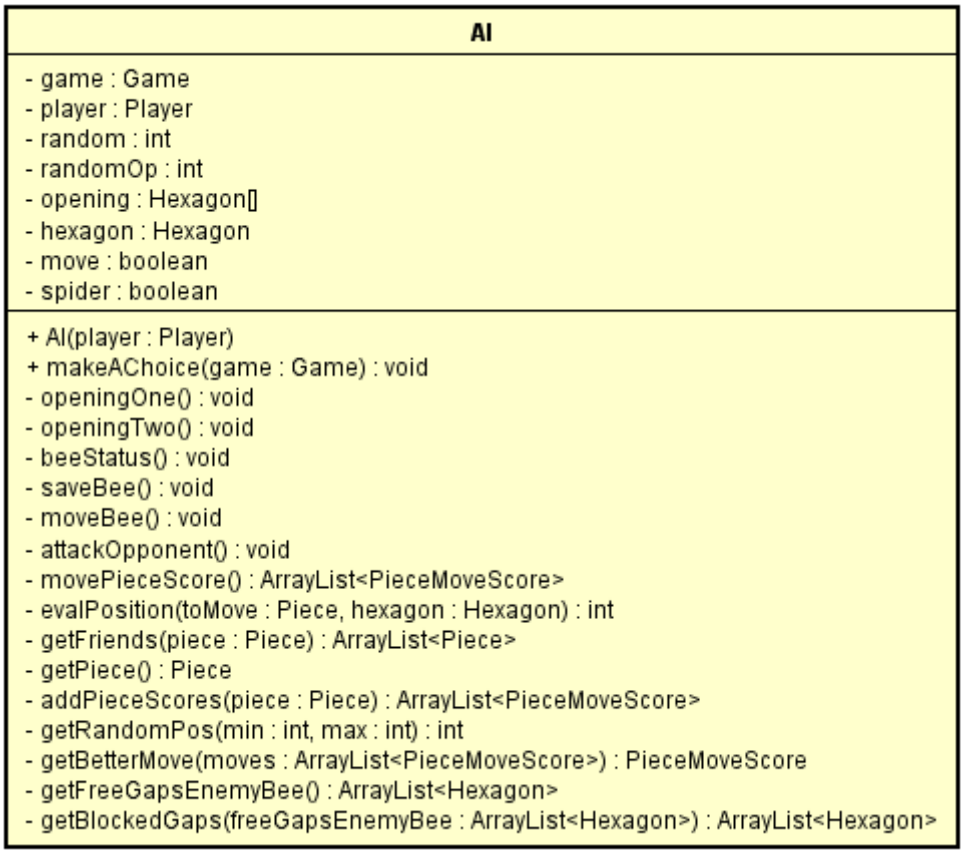

Figura 29. Clase AI.

Tiene una serie de atributos y métodos para llevar a cabo la lógica de la inteligencia artificial propuesta.

Se creó otra clase para almacenar los resultados de la evaluación de posiciones junto con la ficha que se movía y la posición destino. La clase a la que se llamó PieceMoveScore (Ficha-Movimiento-Puntuación) es muy simple:

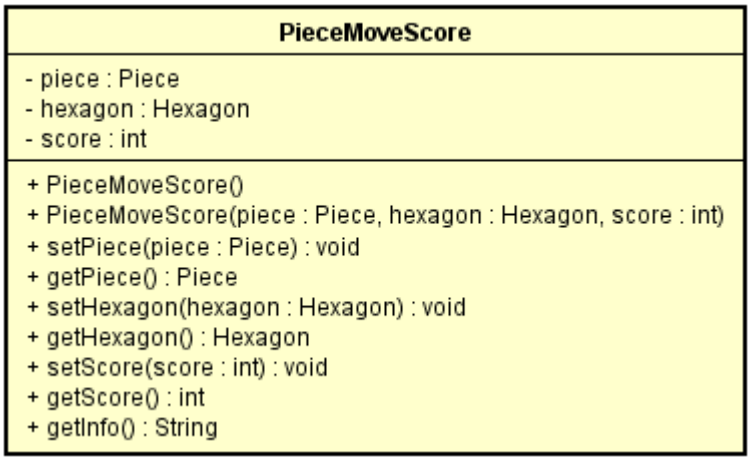

Figura 30. Clase PieceMoveScore.

Todos sus métodos son públicos pues son accedidos desde la clase principal de la inteligencia artificial.

La inteligencia artificial desarrollada es muy simple, pero lo suficientemente compleja como para realizar partidas interesantes y no predecir su siguiente movimiento. Es cierto que algunas veces se puede adivinar cuál va a ser su siguiente jugada, pero jugando contra otra persona pasa lo mismo, hay jugadas que son muy obvias y si sigues estrictamente la lógica del juego se sabe qué va a suceder.

4.5.4 Controlador

Para actuar de intermediario entre la vista y el modelo se creó la clase gameController (Controlador del Juego) que es la única clase del paquete Controller(Controlador). Su única función es pasar información desde el modelo a la vista cuando esta se lo pida. La clase tiene el siguiente aspecto:

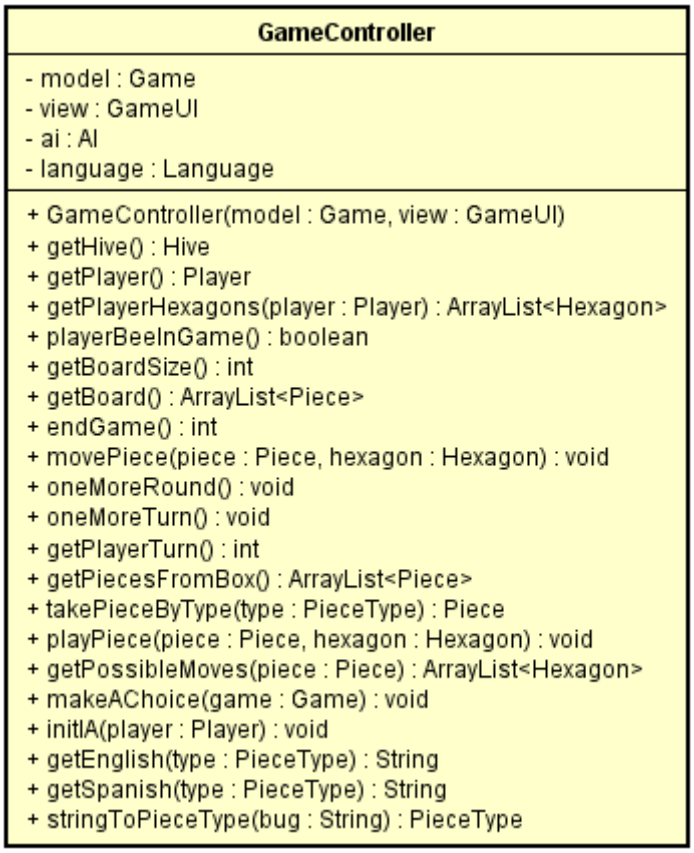

Figura 31. Clase GameController.

Está compuesta de métodos públicos que pueden ser accedidos desde la vista y que a su vez estos acceden al modelo. Algunos de esos métodos acceden directamente a métodos de la vista para mostrar la información mientras que otros simplemente devuelven los datos obtenidos del modelo a la vista.

4.6 Diagramas de secuencia

A continuación se van a mostrar diagramas de secuencia que explican cómo realizar algunos de los casos de uso nombrados en el análisis del sistema.

Todos los diagramas se guardaran en la carpeta "Diagramas/Diagramas de secuencia" para poder apreciarlos con más detalle.

Iniciar una partida contra otro jugador

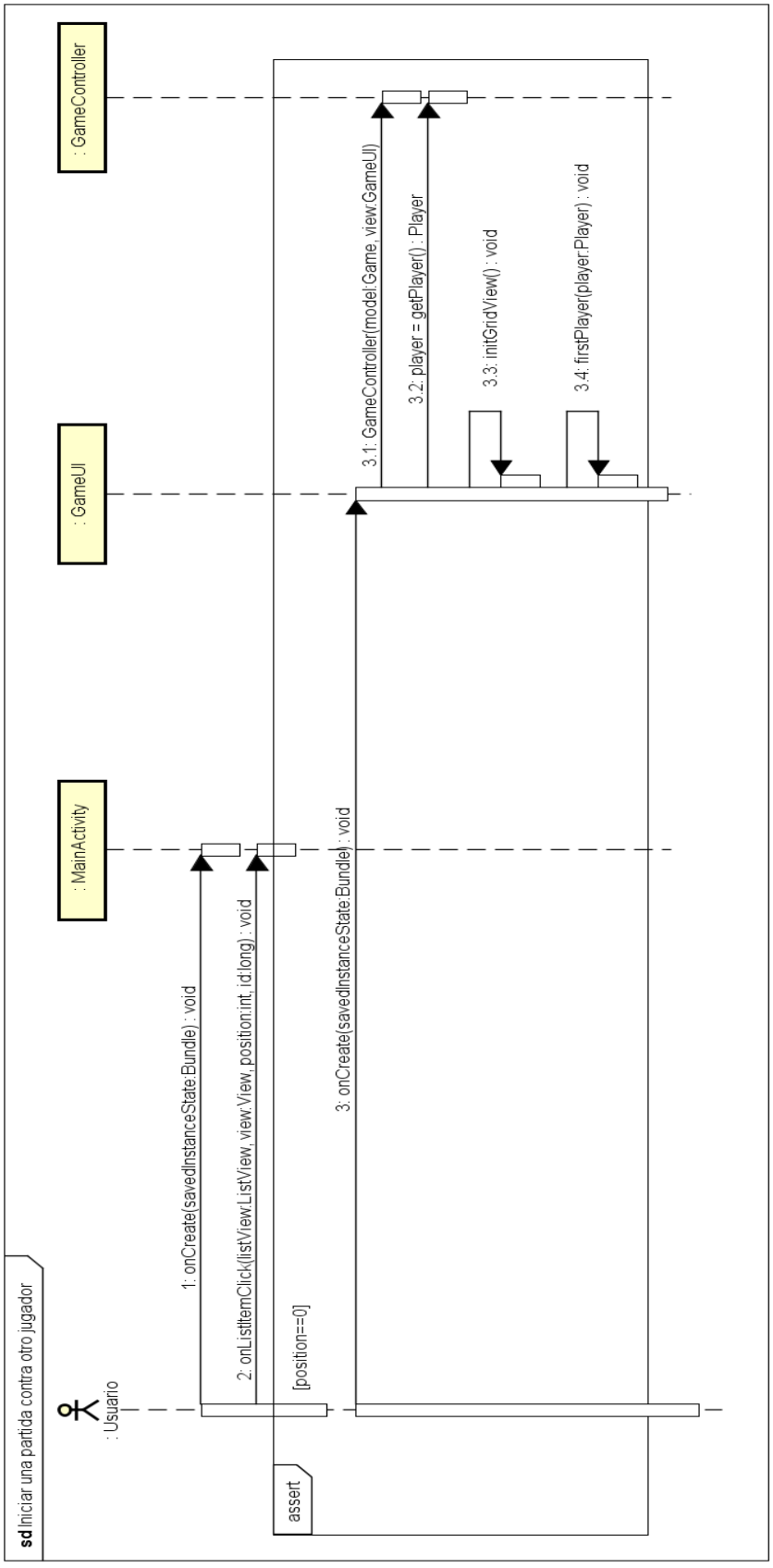

Figura 32. Diagrama de secuencia: Iniciar una partida contra otro jugador.

Iniciar una partida contra la máquina

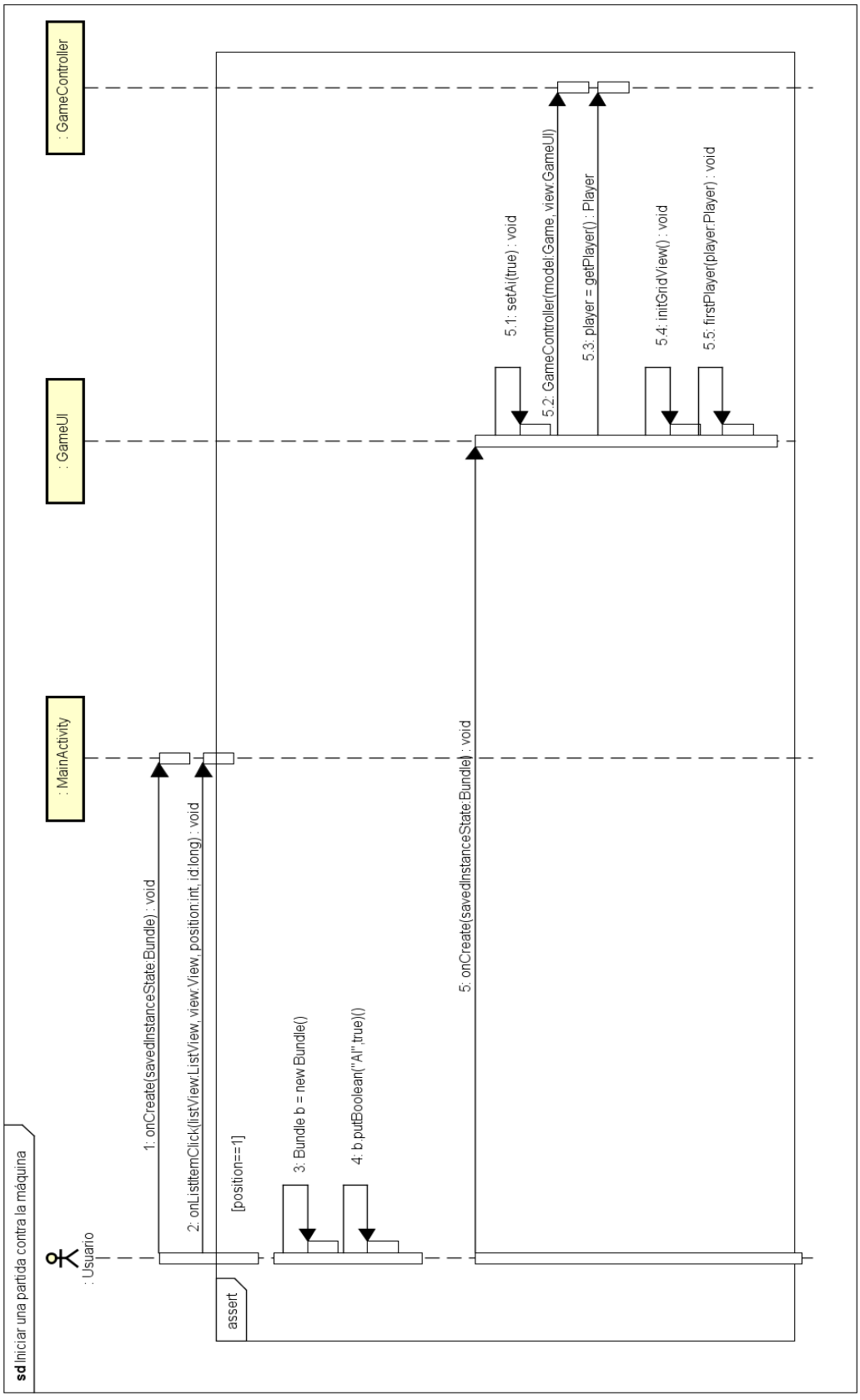

Figura 33. Diagrama de secuencia: Iniciar una partida contra la máquina.

Añadir una ficha al tablero

Este caso de uso se ha dividido en dos diagramas: Mostrar fichas de la caja del jugador, como preludio a poder colocar una ficha, y añadir una ficha al tablero.

Mostrar fichas de la caja del jugador

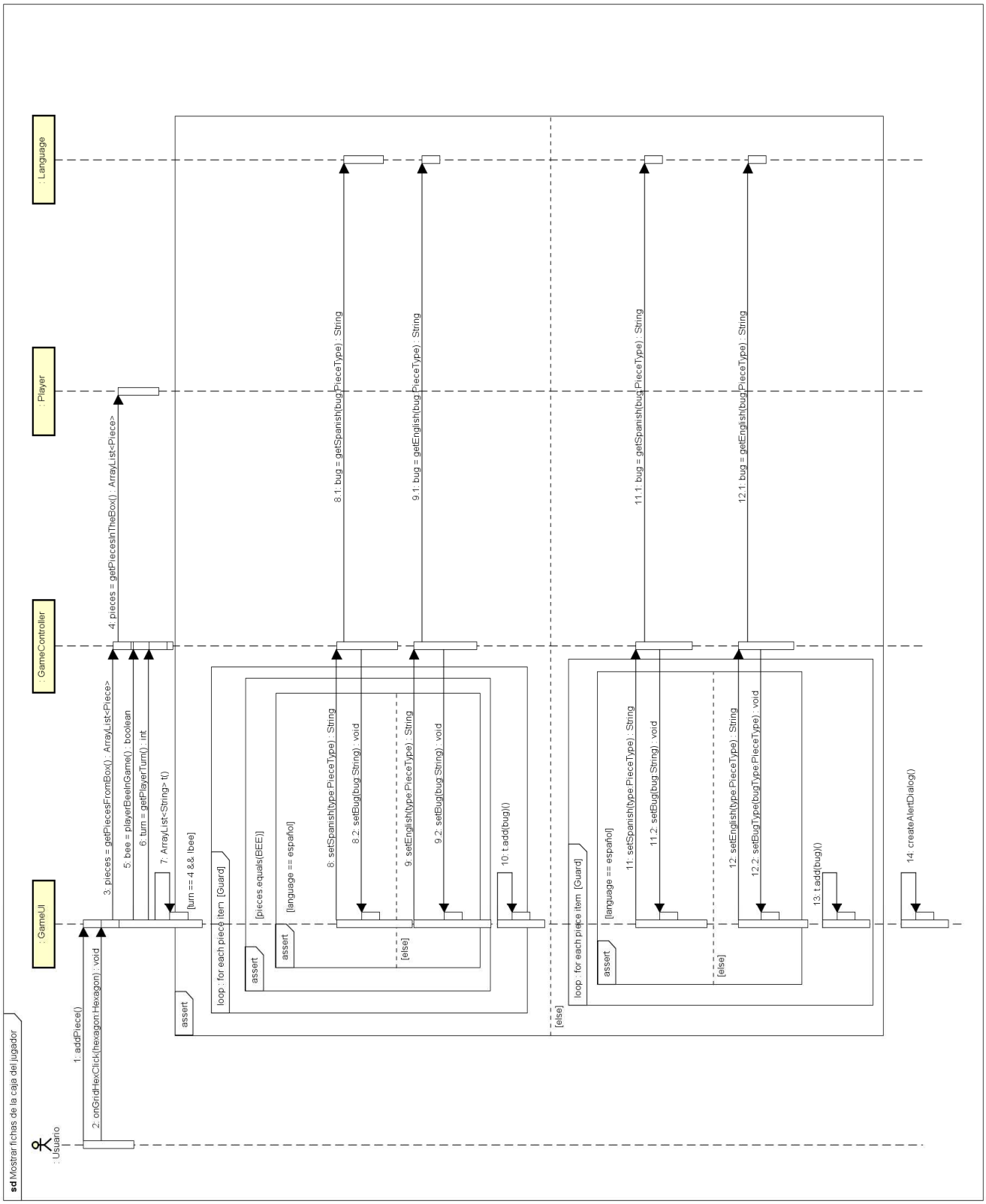

Figura 34. Diagrama de secuencia: Mostrar fichas de la caja del jugador.

Añadir una ficha al tablero

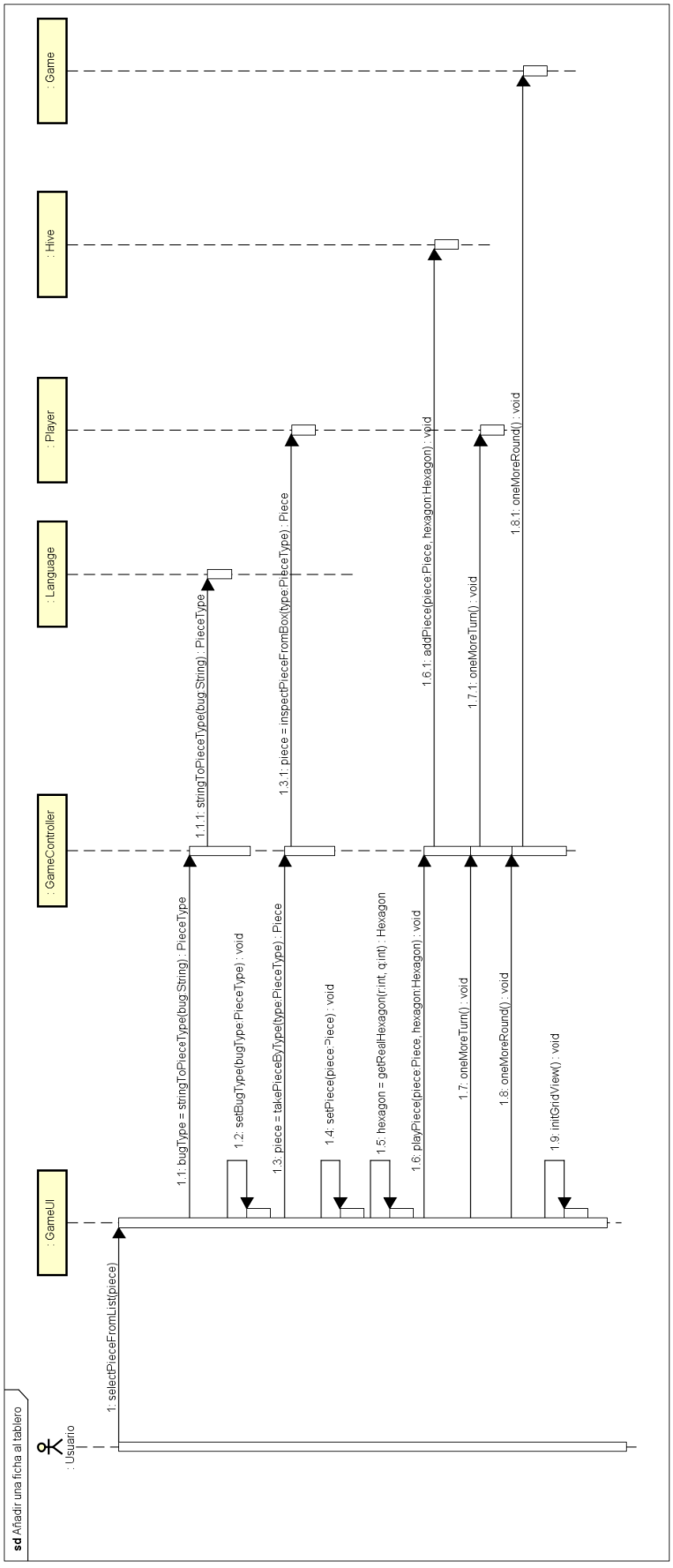

Figura 35. Diagrama de secuencia: Añadir una ficha al tablero.

Cambiar una ficha de posición

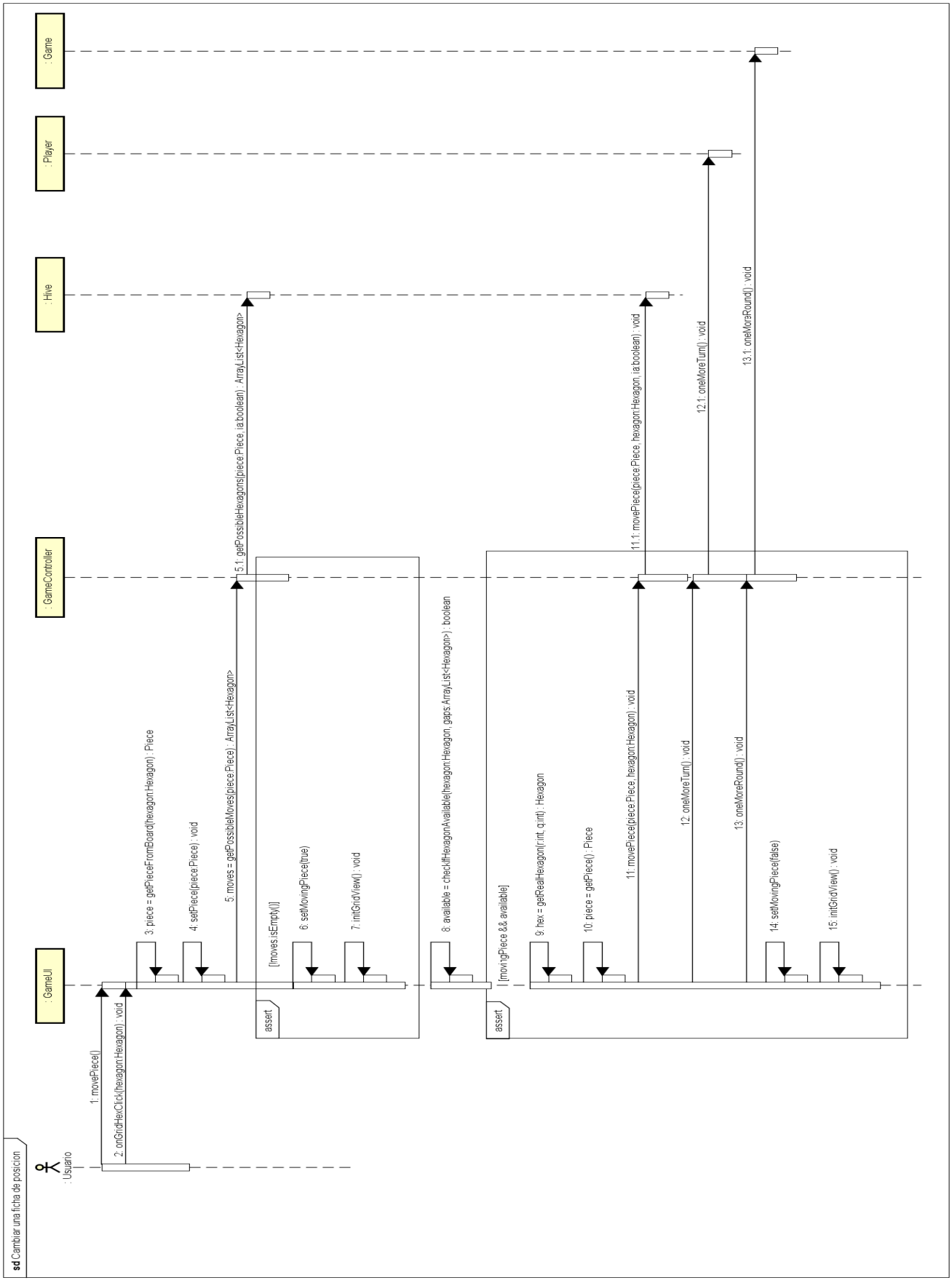

Figura 36. Diagrama de secuencia: Cambiar una ficha de posición.

4.7 Diagrama de Paquetes

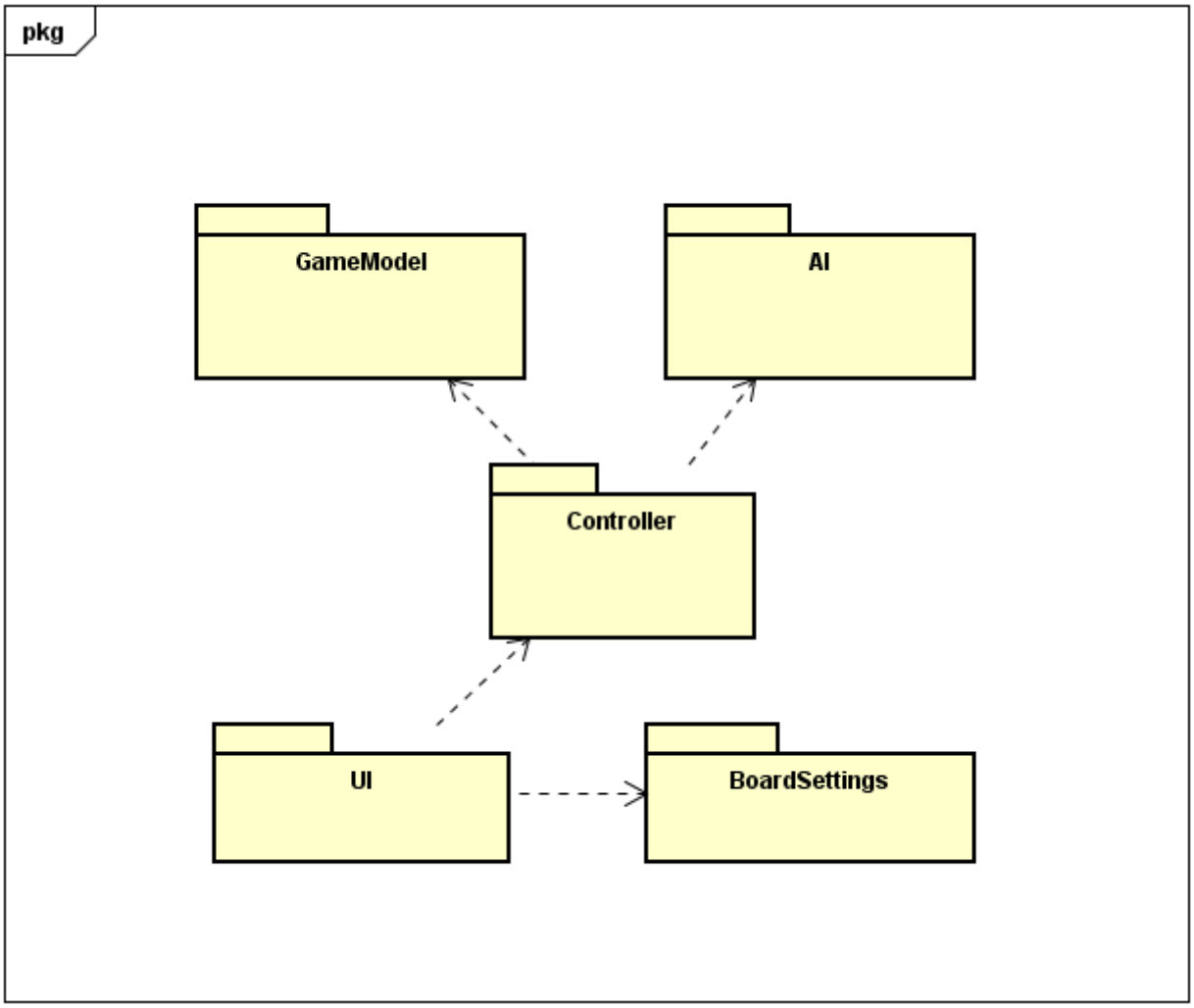

Figura 37. Diagrama de paquetes.

4.8 Diagrama de Clases Completo

El diagrama de clases se muestra en tres colores: Verde para las clases originales, azul para las obtenidas de repositorios públicos y que han sido modificadas y rosa para las obtenidas de repositorios públicos y que no han sufrido ninguna alteración en sus líneas de código.

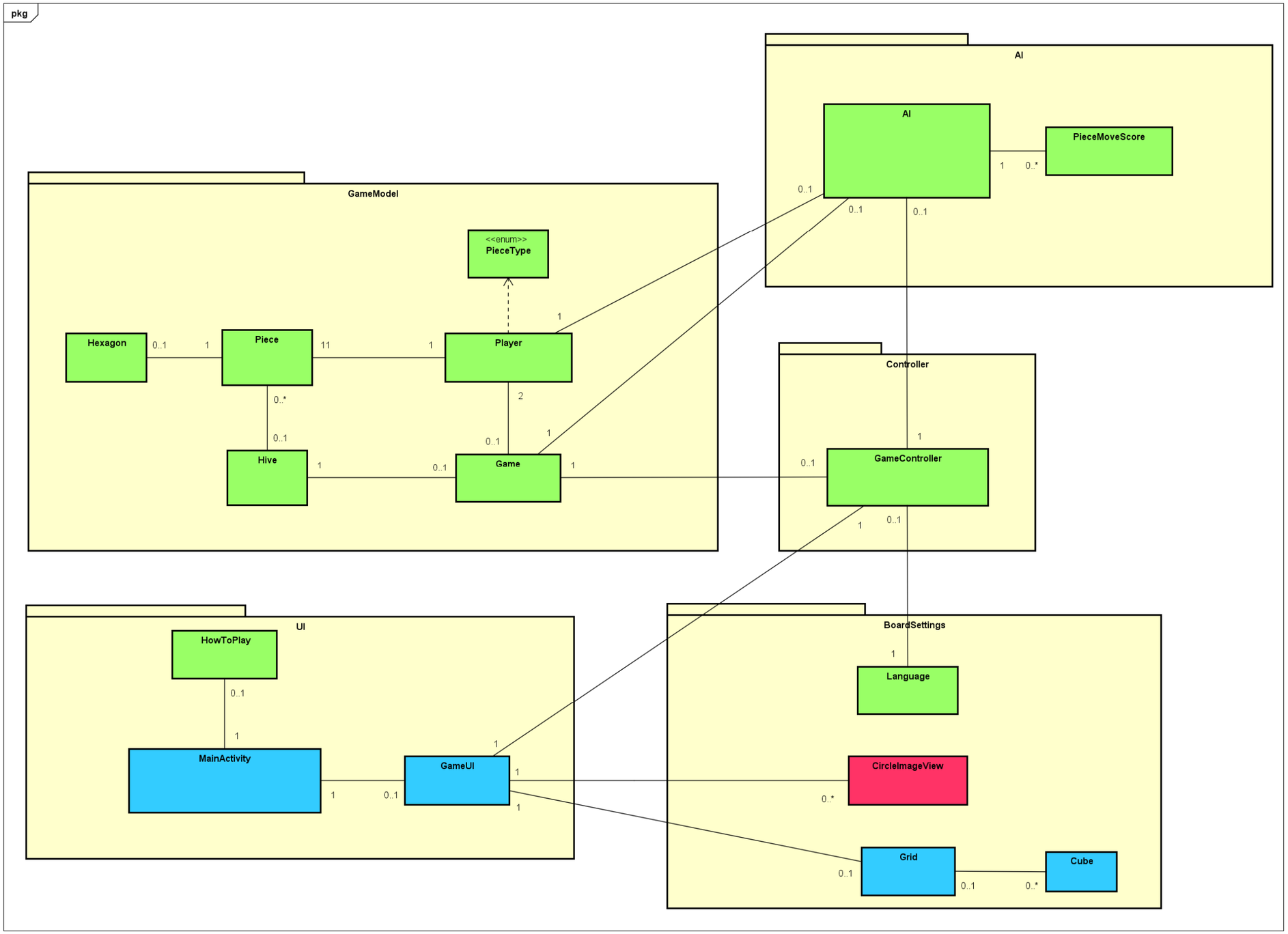

Figura 38. Diagrama de clases

4.9 Diagrama de Clases Completo y al detalle

Como el diagrama de clases completo mostrando todos sus métodos y atributos no se verían bien en este informe, se va a mostrar por partes y se va a guardar el diagrama completo en la carpeta "Diagramas/Diagrama de clases" del CD que contenía esta memoria.

Paquete GameModel:

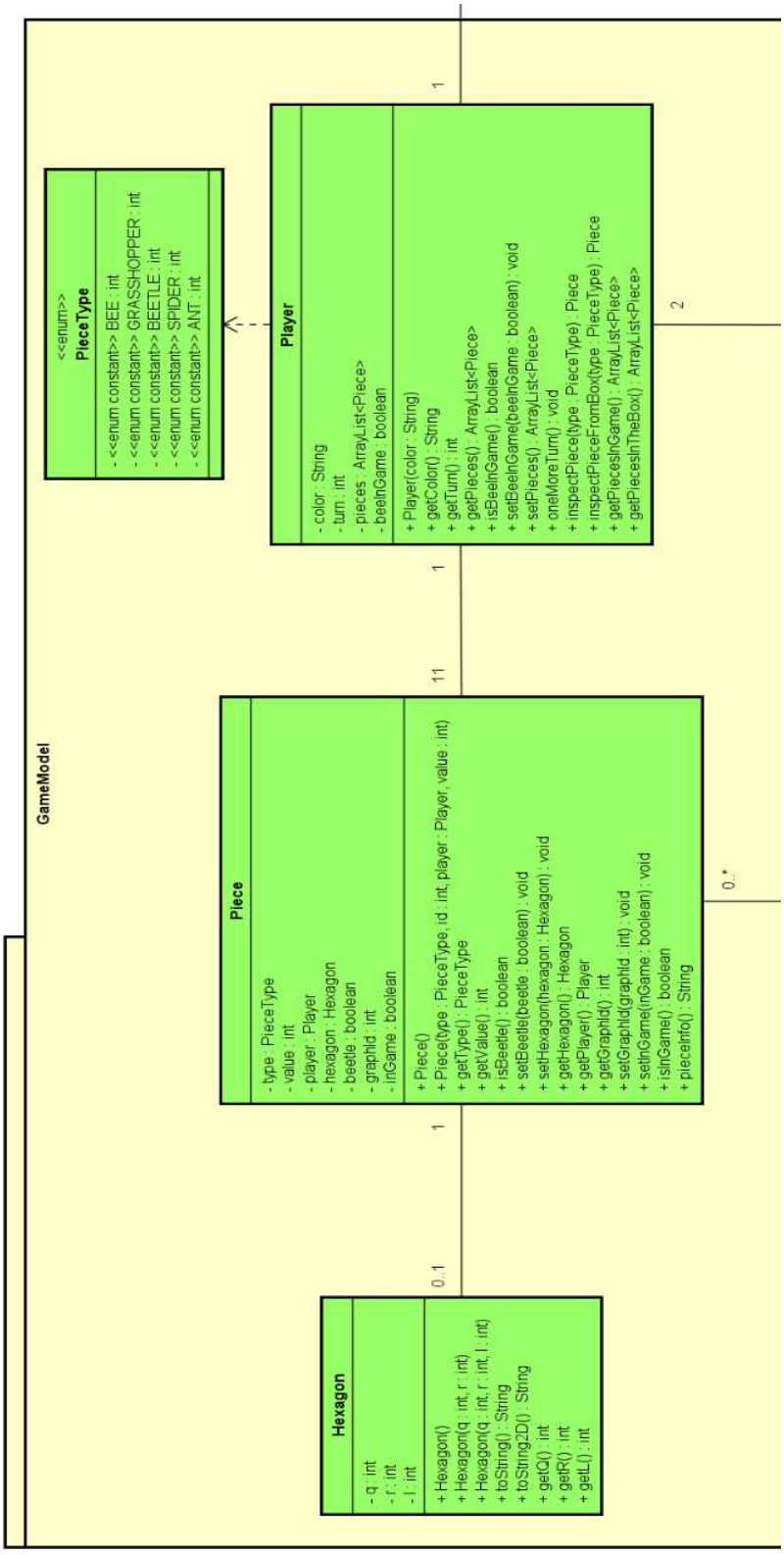

Figura 39. Paquete GameModel. Primera parte.

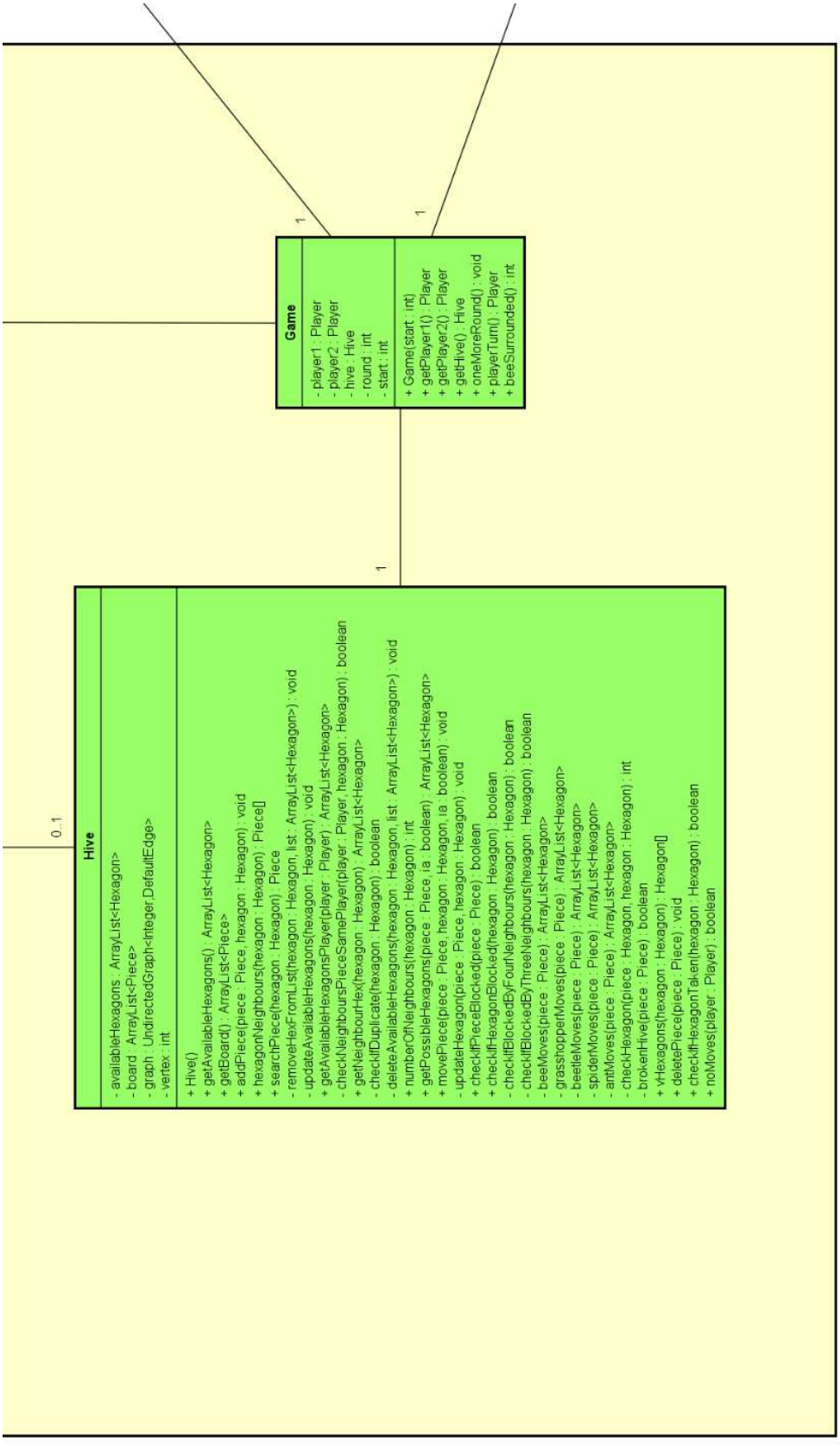

Figura 40. Paquete GameModel. Segunda parte.

Paquete AI:

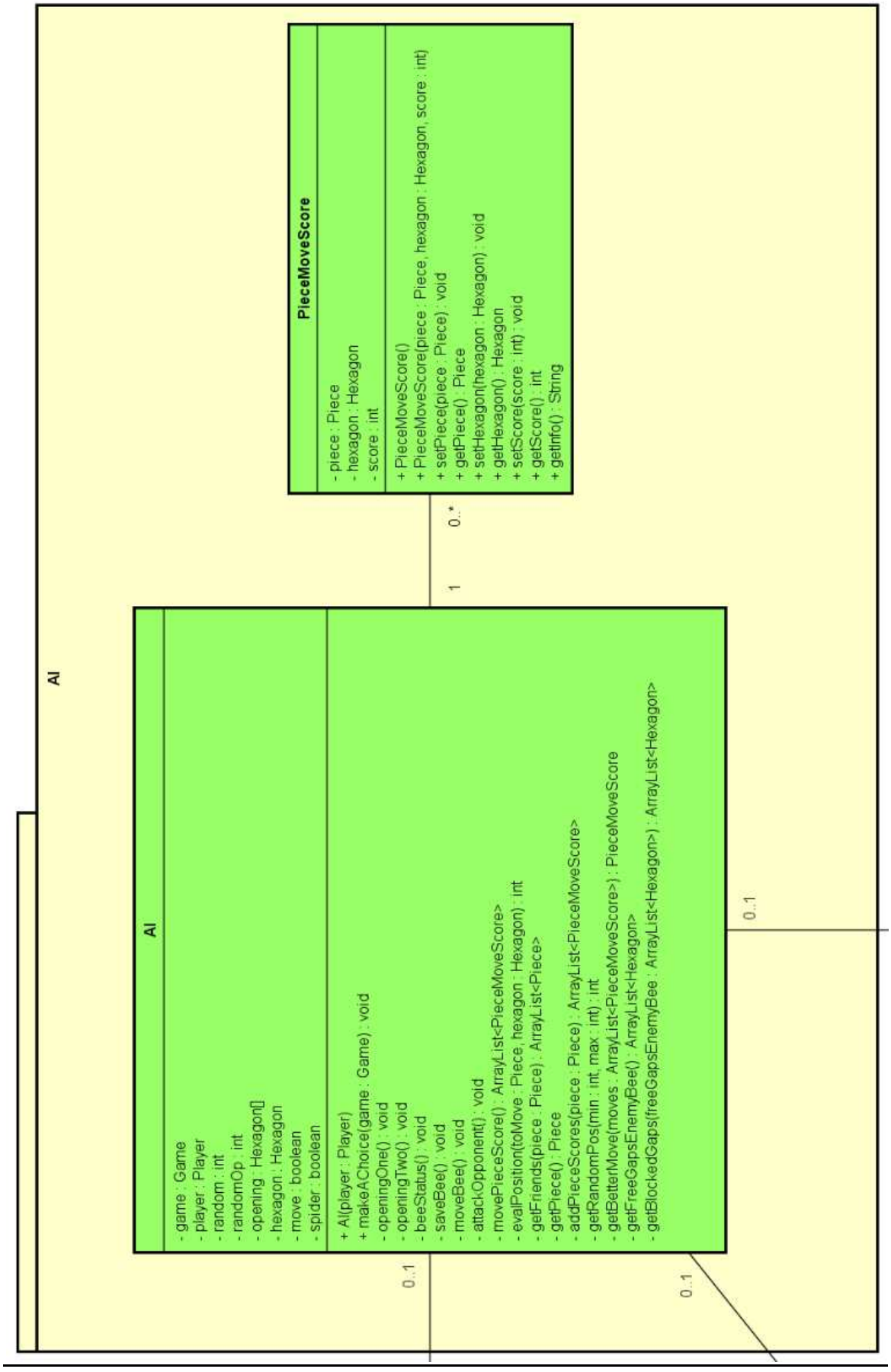

Figura 41. Paquete AI.

Paquete Controller:

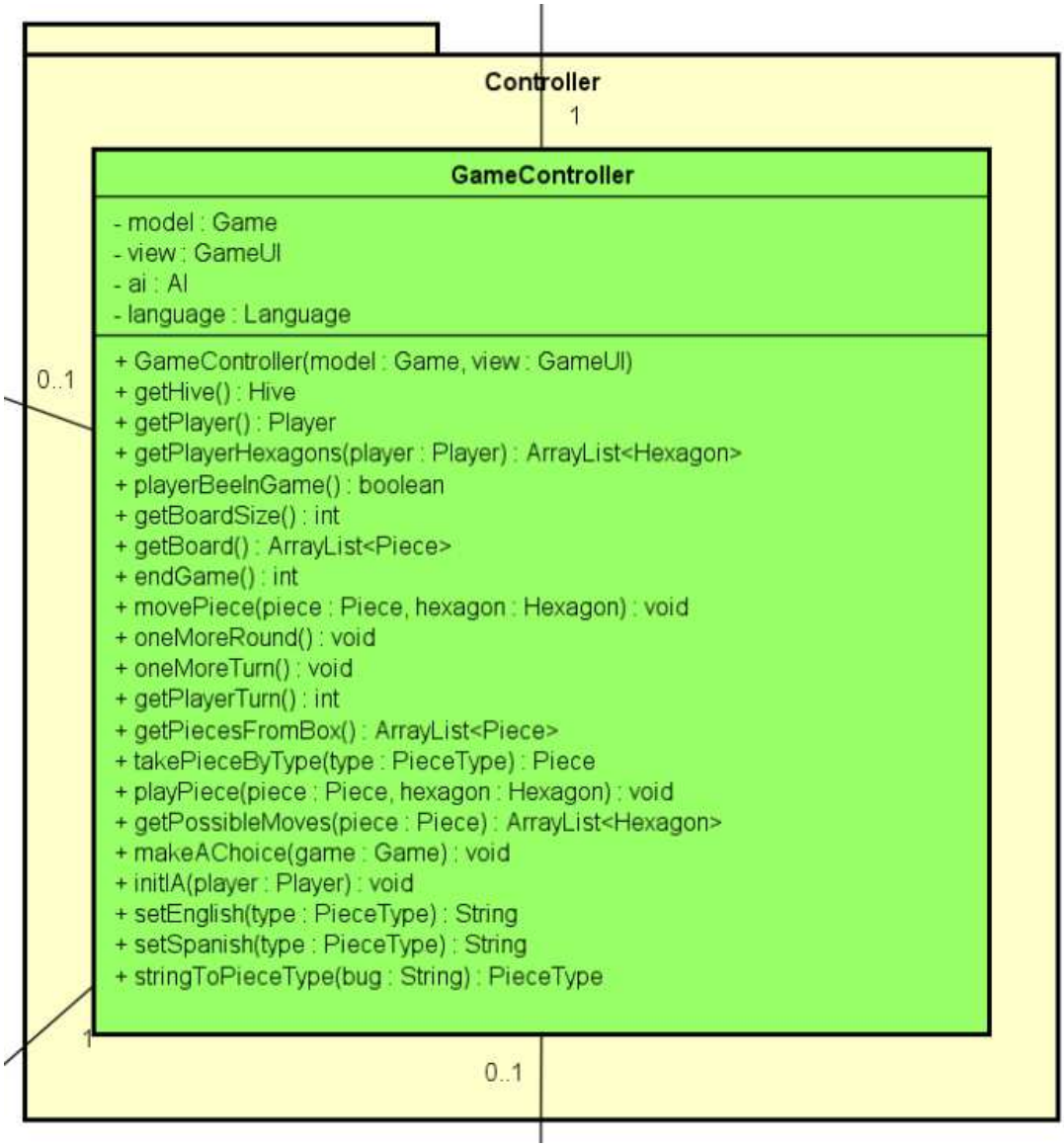

Figura 42. Paquete Controller.

Paquete UI:

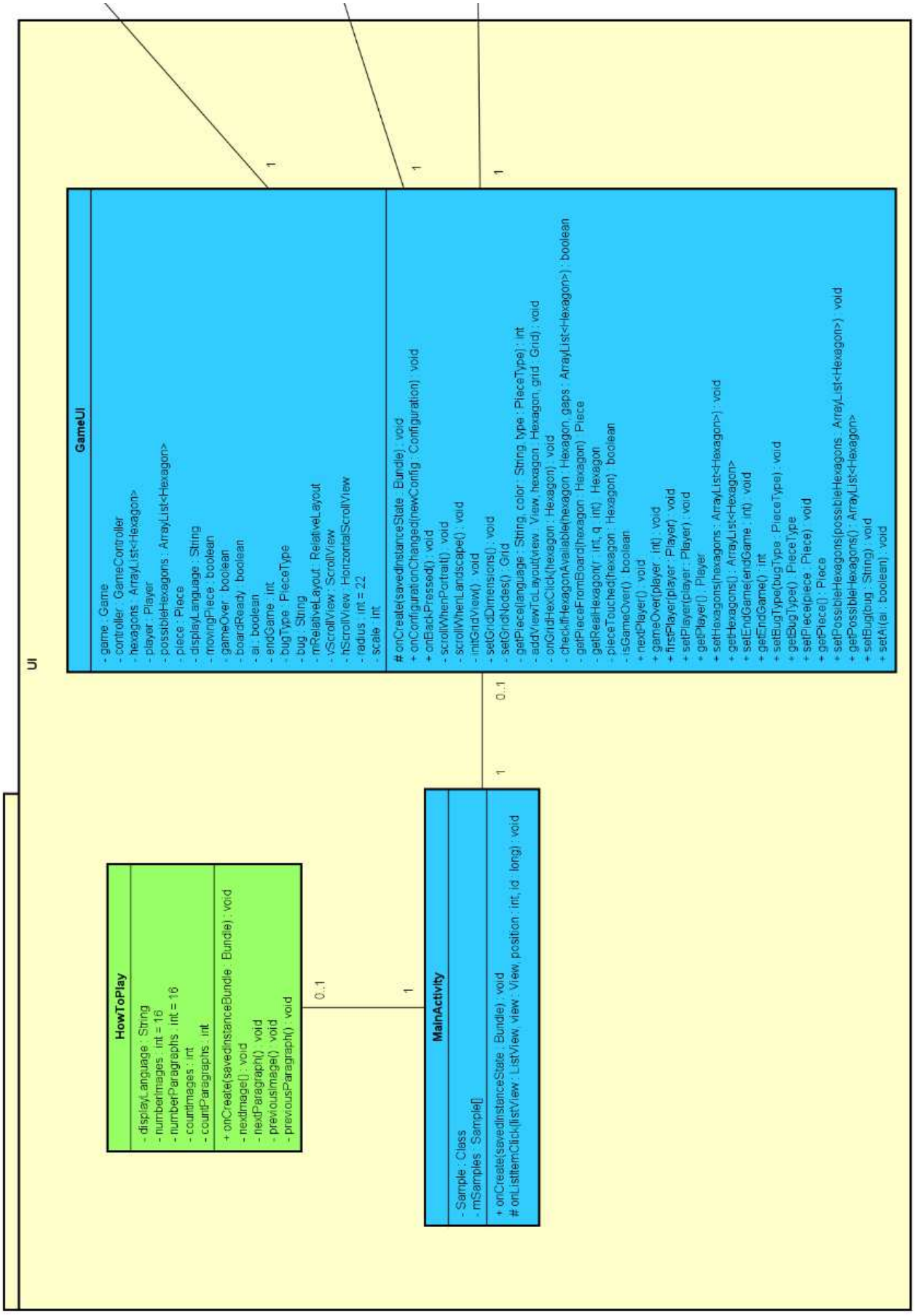

Figura 43. Paquete UI.

Paquete BoardSettings:

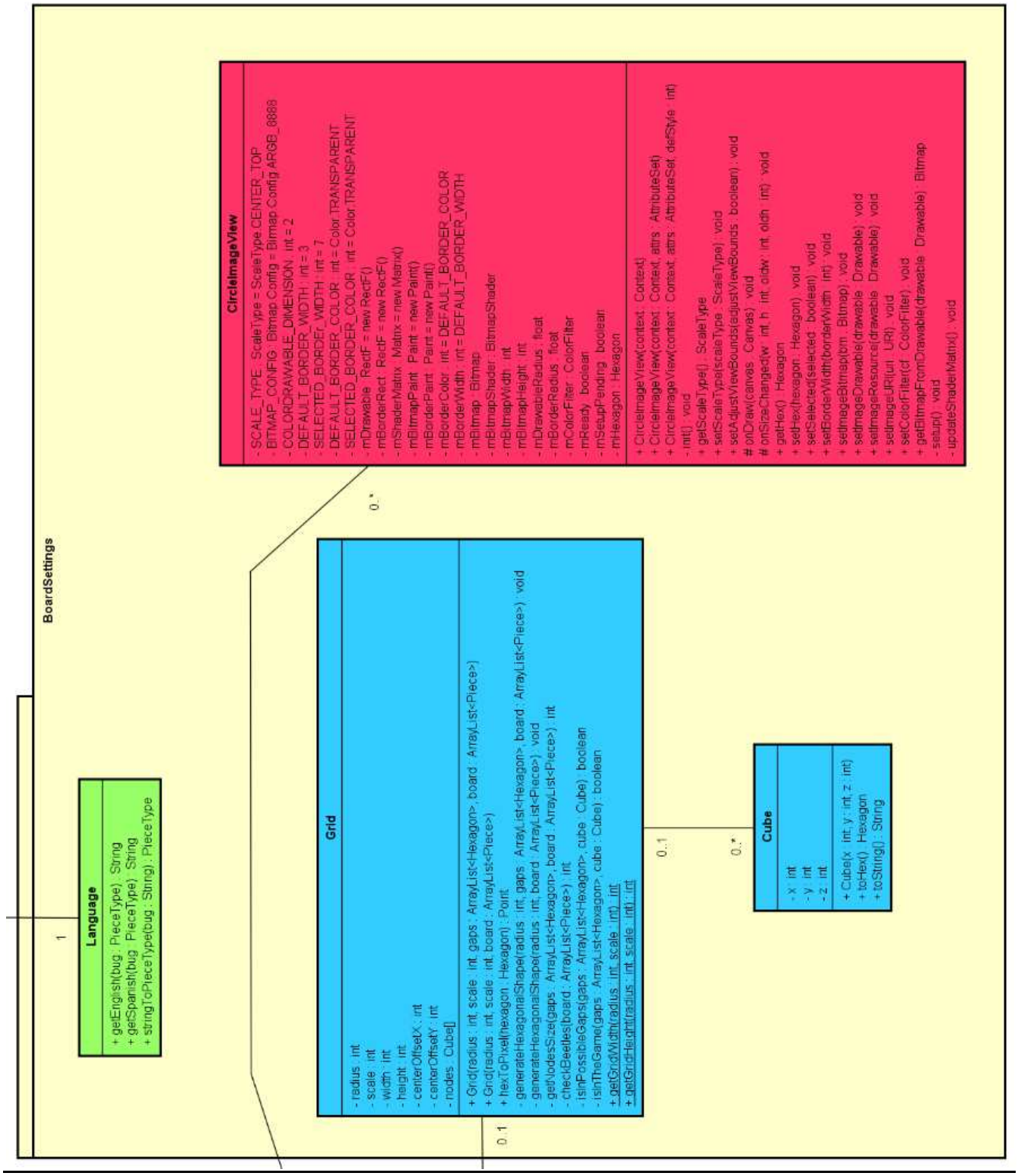

Figura 44. Paquete BoardSettings.

5. Implementación

Este capítulo de implementación está pensado para detallar cómo funcionan los elementos más importantes de la aplicación. Para determinar qué elementos eran los que cobraban más importancia se decidió observar cuáles eran los más imprescindibles para el juego desarrollado, y la respuesta era muy obvia: Las fichas.

Como ya se ha aclarado en capítulos anteriores las fichas representan insectos diferentes, lo que las caracteriza a cada una de ellas con un movimiento que pretende simular el comportamiento del insecto que representa.

A continuación se va a explicar en profundidad cómo se determinan las posibles posiciones a las que puede ir una ficha según su tipo.

5.1 Abeja

La abeja sólo puede desplazarse de uno en uno y deslizándose, lo que quiere decir que no puede forzar la entrada a un hueco que no permita esta acción. Los pasos que se siguen para obtener los posibles movimientos son los siguientes:

1. Se comprueba que la abeja no esté bloqueada. Es decir, que pueda salir de su posición actual deslizándose y sin que esta acción provoque que otras fichas se muevan.
2. Se obtienen todos los hexágonos que la rodean, estén vacíos u ocupados por otras fichas.
3. Para cada una de esos hexágonos se comprueba:
    3.1 Que el hexágono devuelto pertenezca al nivel 0. Lo que indica que está vacío.
    3.2 Si el hexágono está vacío comprueba si puede deslizarse hasta él sin mover otras fichas.

Las posiciones devueltas son aquellas que cumplen todos los pasos del punto 3.
El método que hace todas estas comprobaciones se llama beeMoves y se encuentra en la clase Hive.

5.2 Saltamontes

Los saltamontes se desplazan saltando por encima de otras fichas, nunca podrán saltar huecos. Por lo tanto, la estrategia que se siguió para averiguar sus posibles movimientos fue muy simple: Por cada una de las seis caras que tiene comprueba si tiene un vecino, si lo tiene seguirá comprobando vecinos en esa dirección hasta que encuentre un hueco, que será el destino que se añadirá a la lista de posibles movimientos. Si inicialmente la cara a comprobar no tiene vecino, cancela la acción de buscar más vecinos en ese sentido porque no puede saltar huecos.

El método que hace todas estas comprobaciones se llama grasshopperMoves y se encuentra en la clase Hive.

5.3 Escarabajo

Los escarabajos tiene un movimiento muy similar a la abeja, solo pueden desplazarse un hueco a la vez, pero con una propiedad añadida: pueden situarse encima de otras fichas. Para determinar los posibles movimientos de un escarabajo se realizan los siguientes pasos:

1. Obtiene todos los hexágonos que la rodean, tengan ficha o no.
2. Por cada uno de esos hexágonos hace una serie de comprobaciones:

2.1 Si no se encuentra en el nivel 0: Comprueba que puede ocupar esa posición, para que esto suceda ese hueco deberá no tener vecinos o tener sólo 1.

2.2 Si está en el nivel 0 hará dos comprobaciones: Si el escarabajo a mover está también en el nivel 0 hará la misma comprobación que la abeja: ver si se puede desplazar a ese hueco sin mover otras fichas. Si el escarabajo no está en el nivel 0 solo tendría que bajar a él, sin necesidad de comprobar nada.

Los movimientos del escarabajo pueden ser muy complejos de entender y las reglas oficiales del juego [18] no indican qué hacer en todas las situaciones, como por ejemplo: no se especifica si un escarabajo podrá subirse encima de otra ficha sin tener que desplazarse encima de ella deslizándose. Por lo tanto, se han tenido que completar las reglas con lo que la autora de este proyecto a considerado más lógico.

El método que hace todas estas comprobaciones se llama beetleMoves y se encuentra en la clase Hive.

5.4 Araña

La araña ha sido sin duda la ficha más difícil de implementar. Esta ficha se mueve exactamente tres posiciones en la misma dirección y sin la posibilidad de retroceder. Los pasos seguidos para averiguar los movimientos de una araña han sido los siguientes:

1. Guardar la posición original de la araña.
2. Obtener hexágonos vecinos.
3. Por cada hexágono vecino: Si estaba vacío y se podía deslizar a él se guarda esa posición y se desplaza la araña a ella.
4. Se obtienen los vecinos de esa nueva posición y se comprueba si se puede desplazar a él, si y solo si, la araña no ha estado ya en ese hexágono.

Los pasos del 2 al 4 se repiten tres veces y los últimos huecos obtenidos al final de la tercera iteración son los correspondientes a los posibles destinos de la araña.

El método que hace todas estas comprobaciones se llama spiderMoves y se encuentra en la clase Hive.

5.5 Hormiga

Esta ficha ha sido la más fácil de implementar, pues puede ir en cualquier hueco mientras este no esté bloqueado. Por lo tanto, los pasos seguidos para averiguar sus movimientos han sido los siguientes:

1. Clonar la lista de huecos en los que se puede colocar una ficha a una lista nueva.
2. Guardar la situación actual de la hormiga.
3. Comprobar que no esté bloqueada.
4. Quitar la hormiga del "tablero".
5. De la lista nueva se borra aquellos huecos que no tienen vecinos.
6. Por cada uno de los hexágonos que quedan de la lista nueva se comprueba:
    6.1 Que esté en el nivel 0.
    6.2 Que no esté bloqueada.

Todas los huecos que cumplan todos los pasos del punto 6 serán candidatos para que la hormiga se desplace a ellos.

El método que hace todas estas comprobaciones se llama antMoves y se encuentra en la clase Hive.

6. Pruebas

6.1 Introducción

En este tema se van a exponer las pruebas a las que se sometió la aplicación para comprobar que su funcionamiento era correcto y el esperado.

Se realizaron pruebas a tres niveles:

1. **Acceso a las distintas secciones de la aplicación**: Recorrido que un usuario que ve por primera vez la aplicación realiza para familiarizarse con ella. Se accede a los distintos modos de juego y a las reglas del juego y se navega de unos a otros para comprobar que la aplicación responde correctamente y se muestran los diálogos correspondientes.
2. **Partida contra otro jugador**: En esta parte se realizan pruebas con las fichas y se confirma que al finalizar una partida se despliegan los diálogos que muestran el ganador y preguntan al usuario si quieren echar otra partida. Para hacer pruebas en esta fase se echaron tanto partidas contra otro usuario como partidas prueba, que sirven para comprobar que todo responde correctamente.
3. **Partida contra la máquina**: Esta sección fue posiblemente la más monótona a la hora de realizar las pruebas, pues consistía básicamente en jugar partidas contra la máquina y comprobar si realizaba movimientos legales. Se elaboró una tabla en la que se registraban datos de las partidas guardadas, como quién empezó, quién ganó, captura de la situación final y comentarios de la partida. Dicha tabla se encuentra en el ANEXO IV.

6.2 Clasificación de Tipos de Pruebas

Hay dos tipos de pruebas: Indirectas o directas.

- **Indirectas**: No se comprueba el correcto funcionamiento del código ejecutándolo en una máquina, sino que se inspecciona el código en busca de errores.
- **Directas**: Se selecciona un conjunto de casos de prueba y se ejecutan en una máquina real. Los hay de dos tipos:
    - **Pruebas de Caja Negra**: Los casos de prueba no se seleccionan al ver el código, sino que se crean a partir de las especificaciones.
    - **Pruebas de Caja Blanca**: Los casos de prueba se seleccionan según el código del programa.

  [29]

Para la creación de pruebas de esta aplicación se ha elegido “Pruebas de Caja Negra” pues era la forma más sencilla de realizarlas y sobretodo porque se podía ver con claridad las respuestas que recibiría el usuario final y si dichas respuestas eran las correctas.

También se aplicaron durante toda la fase de desarrollo del juego pruebas indirectas, inspeccionando el código en busca de errores cada vez que se añadía algún método que no devolvía el resultado esperado.

6.3 Dispositivos de Pruebas

Para realizar las pruebas se han utilizado 3 dispositivos Android de distinta versión, modelo y tamaño de pantalla. Las características de dichos dispositivos se muestran a continuación junto con un identificador que servirá para nombrarlos en la sección de “Realización de Pruebas”.

Identificador	Dispositivo_1
Modelo	Samsung Galaxy Tab 4
Número de Modelo	SM-T230
Versión de Android	4.4.2
Memoria RAM	1.8GB
Memoria del dispositivo	8GB
Tamaño de la pantalla	7’’

Tabla 58. Descripción del dispositivo de prueba 1.

Identificador	Dispositivo_2
Modelo	Samsung Galaxy Tab 3
Número de Modelo	SM-T310
Versión de Android	4.4.2
Memoria RAM	1.5GB
Memoria del dispositivo	16GB
Tamaño de la pantalla	8’’

Tabla 59. Descripción del dispositivo de prueba 2.

Identificador	Dispositivo_3
Modelo	BQ Aquaris M5
Número de Modelo	Aquaris M5
Versión de Android	6.0.1
Memoria RAM	2GB
Memoria del dispositivo	16GB
Tamaño de la pantalla	5’’

Tabla 60. Descripción del dispositivo de prueba 3.

Los dispositivos 1 y 2 son tablets de la misma marca (Samsung) y casi idénticas características, salvo el tamaño de pantalla que fue la razón principal por la que se optó a realizar pruebas en estos dos dispositivos. El dispositivo 3 es un teléfono móvil inteligente (SmartPhone) con una versión de Android superior a la de los dispositivos 1 y 2 Tiene instalada la versión Android 6.0.1 mientras que los otros tienen la 4.4.2, y el tamaño de la pantalla es notablemente más pequeño, por lo que este dispositivo se usó principalmente para comprobar que todos los elementos que se tienen que mostrar no salieran sobredimensionados o mal centrados y no para jugar muchas partidas como se hizo con los otros dos dispositivos.

6.4 Casos de Prueba

Prerrequisitos para todas las pruebas:

- El idioma del dispositivo esta en español o inglés. Si se encontrase en algún otro idioma se mostrarían los textos en inglés.

6.4.1 Interfaz de Usuario (Recorrido por la aplicación)

Las pruebas que se realizaron para comprobar el correcto funcionamiento de la interfaz de usuario fueron las siguientes:

P-01 Visualización del menú principal en la tablet.
P-02 Visualización del menú principal en el teléfono móvil.
P-03 Visualización del inicio de una partida contra otro jugador en una tablet.
P-04 Visualización del inicio de una partida contra otro jugador en un teléfono inteligente.
P-05 Visualización del inicio de una partida contra la máquina en una tablet.
P-06 Visualización del inicio de una partida contra la máquina en un teléfono inteligente.
P-07 Visualización de las reglas del juego en una tablet.
P-08 Visualización de las reglas del juego en un teléfono inteligente.
P-09 Rotar el dispositivo (tablet) durante una partida.
P-10 Rotar el dispositivo (Teléfono Inteligente) durante una partida.

A continuación se mostrarán los resultados de dichas pruebas mediante unas tablas que indicarán: número de prueba, título, dispositivos empleados para realizarla, descripción, pasos a seguir para completar la prueba, salida esperada, salida obtenida, errores detectados y solución de errores si los hubiera, y resultado de la prueba: si fue satisfactoria o no.

P-01	Visualización del menú principal en la tablet
Dispositivos	Dispositivo_1, Dispositivo_2.
Descripción	Verificación de la correcta visualización del menú principal y sus componentes en una tablet.
Pasos a seguir	1. Abrir la aplicación, accediendo así al menú principal.
Salida Esperada	El logo y el nombre de la aplicación se muestran correctamente en la parte izquierda de la pantalla, el menú interactivo en la derecha y el idioma es el correspondiente.
Salida Obtenida	El logo y el nombre de la aplicación se muestran correctamente en la parte izquierda de la pantalla y el menú interactivo en la derecha. El idioma en que se visualiza el menú es el correspondiente.
Errores detectados	Ninguno.
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba.

Tabla 61. Descripción de la prueba 1.

P-02	Visualización del menú principal en el teléfono móvil
Dispositivos	Dispositivo_3.
Descripción	Verificación de la correcta visualización del menú principal y sus componentes en un teléfono móvil.
Pasos a seguir	1. Abrir la aplicación, accediendo así al menú principal.
Salida Esperada	El logo y el nombre de la aplicación se muestran correctamente en la parte izquierda de la pantalla, el menú interactivo en la derecha y el idioma es el correspondiente.
Salida Obtenida	El logo y el nombre de la aplicación se muestran correctamente en la parte izquierda de la pantalla y el menú interactivo en la derecha. El idioma en que se visualiza el menú es el correspondiente.
Errores detectados	Ninguno.
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba.

Tabla 62. Descripción de la prueba 2.

P-03	Visualización del inicio de una partida contra otro jugador en una tablet
Dispositivos	Dispositivo_1, Dispositivo_2.
Descripción	Verificación de la correcta visualización del inicio de una partida
Pasos a seguir	Prerrequisito: Haber accedido a la aplicación. 1. Acceder a “Jugador VS Jugador” (español) y “Player VS Player” (inglés). 2. Aceptar el diálogo.
Salida Esperada	Se muestra un hexágono gris centrado sobre un fondo amarillento
Salida Obtenida	Se muestra un hexágono gris centrado sobre un fondo amarillento
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 63. Descripción de la prueba 3.

P-04	Visualización del inicio de una partida contra otro jugador en un teléfono inteligente
Dispositivos	Dispositivo_3
Descripción	Verificación de la correcta visualización del inicio de una partida
Pasos a seguir	Prerrequisito: Haber accedido a la aplicación. 1. Acceder a “Jugador VS Jugador” (español) y “Player VS Player” (inglés). 2. Aceptar el diálogo
Salida Esperada	Se muestra un hexágono gris centrado sobre un fondo amarillento
Salida Obtenida	Se muestra un hexágono gris **descentrado** sobre un fondo amarillento
Errores detectados	El hexágono gris no queda centrado en la pantalla
Solución de Errores	No se implementó ninguna solución al ser un error visual de menor importancia
Resultado de la prueba	Insatisfactorio

Tabla 64. Descripción de la prueba 4.

P-05	Visualización del inicio de una partida contra la máquina en una tablet
Dispositivos	Dispositivo_1, Dispositivo_2
Descripción	Verificación de la correcta visualización del inicio de una partida
Pasos a seguir	Prerrequisito: Haber accedido a la aplicación. 1. Acceder a “Jugador VS Máquina” (español) y “Player VS Machine” (inglés). 2. Aceptar el diálogo
Salida Esperada	Se muestra un hexágono gris centrado sobre un fondo amarillento
Salida Obtenida	Se muestra un hexágono gris centrado sobre un fondo amarillento
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 65. Descripción de la prueba 5.

P-06	Visualización del inicio de una partida contra la máquina en un teléfono inteligente
Dispositivos	Dispositivo_3
Descripción	Verificación de la correcta visualización del inicio de una partida
Pasos a seguir	Prerrequisito: Haber accedido a la aplicación. 1. Acceder a “Jugador VS Máquina” (español) y “Player VS Machine” (inglés). 2. Aceptar el diálogo
Salida Esperada	Se muestra un hexágono gris centrado sobre un fondo amarillento
Salida Obtenida	Se muestra un hexágono gris **descentrado** sobre un fondo amarillento
Errores detectados	El hexágono gris no queda centrado en la pantalla
Solución de Errores	No se implementó ninguna solución al ser un error visual de menor importancia
Resultado de la prueba	Insatisfactorio

Tabla 66. Descripción de la prueba 6.

P-07	Visualización de las reglas del juego en una tablet
Dispositivos	Dispositivo_1, Dispositivo_2
Descripción	Verificación de la correcta visualización de las reglas del juego
Pasos a seguir	Prerrequisito: Haber accedido a la aplicación en un dispositivo en español. 1. Acceder a “Cómo Jugar” 2. Pulsar en el texto para pasar las pantallas hasta llegar al final. 3. Pulsar en la imagen para pasar pantallas hasta llegar al principio.
Salida Esperada	Correcta visualización de todas las pantallas
Salida Obtenida	Correcta visualización de todas las pantallas
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 67. Descripción de la prueba 7.

P-08	Visualización de las reglas del juego en un teléfono inteligente
Dispositivos	Dispositivo_3
Descripción	Verificación de la correcta visualización de las reglas del juego
Pasos a seguir	Prerrequisito: Haber accedido a la aplicación en un dispositivo en español. 1. Acceder a “Cómo Jugar” 2. Pulsar en el texto para pasar las pantallas hasta llegar al final 3. Pulsar en la imagen para pasar pantallas hasta llegar al principio
Salida Esperada	Correcta visualización de todas las pantallas
Salida Obtenida	Correcta visualización de todas las pantallas
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 68. Descripción de la prueba 8.

P-09	Rotar dispositivo (tablet) durante una partida
Dispositivos	Dispositivo_1, Dispositivo_2
Descripción	Verificación de la correcta visualización de los elementos de una partida cuando se gira el dispositivo
Pasos a seguir	Prerrequisito: Haber iniciado una partida de cualquier tipo y haber jugado o no algunas rondas. El dispositivo también tiene que tener activada la rotación de la pantalla. 1. Girar el dispositivo 360º lentamente.
Salida Esperada	Las fichas de la partida siempre aparecen centradas. Si no se jugó ninguna partida el hexágono gris inicial deberá aparecer siempre en el centro de la pantalla.
Salida Obtenida	Las fichas de la partida siempre aparecen centradas. Si no se jugó ninguna partida el hexágono gris inicial aparece en el centro de la pantalla.
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 69. Descripción de la prueba 9.

P-10	Rotar dispositivo (Teléfono Inteligente) durante una partida
Dispositivos	Dispositivo_3
Descripción	Verificación de la correcta visualización de los elementos de una partida cuando se gira el dispositivo
Pasos a seguir	Prerrequisito: Haber iniciado una partida de cualquier tipo y haber jugado o no algunas rondas. El dispositivo también tiene que tener activada la rotación de la pantalla. 1. Girar el dispositivo 360º lentamente.
Salida Esperada	Las fichas de la partida siempre aparecen centradas. Si no se jugó ninguna partida el hexágono gris inicial deberá aparecer siempre en el centro de la pantalla.
Salida Obtenida	Las fichas no aparecen centradas al rotar la pantalla
Errores detectados	Las fichas no aparecen centradas cuando se cambia de sentido el dispositivo
Solución de Errores	Ninguno, error estético que podría arreglarse en versiones futuras
Resultado de la prueba	Insatisfactorio

Tabla 70. Descripción de la prueba 10.

6.4.2 Partida contra otro jugador

Las pruebas que se realizaron para comprobar el correcto funcionamiento de una partida contra otro jugador real fueron las siguientes:

P-11 Acceso a una partida contra otro jugador
P-12 Iniciar a una partida contra otro jugador
P-13 Abandonar una partida contra otro jugador
P-14 Cancelar el abandono de una partida contra otro jugador
P-15 Añadir una ficha al juego
P-16 Cancelar la acción de añadir una ficha al juego
P-17 Comprobar que no se pueden añadir más fichas cuando el jugador tiene la caja vacía
P-18 Añadir todas las fichas al juego
P-19 Comprobar que al cuarto turno solo se puede añadir la abeja
P-20 Obtener los posibles movimientos de una ficha según su tipo
P-21 Obtener los posibles movimientos de una ficha que no se puede mover
P-22 Cancelar la vista de los posibles movimientos de una ficha seleccionada para ello
P-23 Comprobar que no se pueden mover fichas ni ver sus posibles movimientos si la abeja del jugador no está en juego
P-24 Mover una ficha
P-25 Comprobar que tocar la zona fuera de la colmena no produce ningún efecto
P-26 Comprobar que al ganar/perder una partida se muestra un diálogo
P-27 Comprobar que al querer abandonar una partida finalizada sale un diálogo que se puede confirmar
P-28 Comprobar que al querer abandonar una partida finalizada sale un diálogo que se puede rechazar

La forma de mostrar las pruebas será la misma que en el apartado anterior: número de prueba, título, dispositivos empleados para realizarla, descripción, pasos a seguir para completar la prueba, salida esperada, salida obtenida, errores detectados y solución de errores si los hubiera, y resultado de la prueba: si fue satisfactoria o no.

P-11	Acceso a una partida contra otro jugador
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de acceso a una nueva partida, en la que se debe mostrar un diálogo indicando qué jugador empieza primero
Pasos a seguir	Prerrequisito: Haber accedido a la aplicación. 1. Acceder a “Jugador VS Jugador”
Salida Esperada	Se accede a la partida y se muestra un diálogo indicando qué jugador empieza primero
Salida Obtenida	Se accede a la partida y se muestra un diálogo indicando qué jugador empieza primero
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 71. Descripción de la prueba 11.

P-12	Iniciar a una partida contra otro jugador
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación del inicio de una nueva partida
Pasos a seguir	Prerrequisito: Haber accedido a la aplicación. 1. Acceder a “Jugador VS Jugador” 2. Aceptar el diálogo
Salida Esperada	Tablero de juego con un hexágono gris sobre un fondo anaranjado
Salida Obtenida	Tablero de juego con un hexágono gris sobre un fondo anaranjado
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 72. Descripción de la prueba 12.

P-13	Abandonar una partida contra otro jugador
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación del abandono de una partida y acceso al menú principal
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador. 1. Pulsar botón de retroceso en el dispositivo. 2. Confirmar diálogo
Salida Esperada	Se muestra el menú principal
Salida Obtenida	Se muestra el menú principal
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 73. Descripción de la prueba 13.

P-14	Cancelar el abandono de una partida contra otro jugador
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de la cancelación del abandono de una partida
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador. 3. Pulsar botón de retroceso en el dispositivo. 4. Cancelar
Salida Esperada	Se sigue mostrando la partida y su estado no se ha alterado
Salida Obtenida	Se sigue mostrando la partida y su estado no se ha alterado
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 74. Descripción de la prueba 14.

P-15	Añadir una ficha al juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de añadir una ficha al juego
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina y que sea su turno. 1. Pulsar un hexágono gris 2. Seleccionar un insecto de la lista desplegada
Salida Esperada	La ficha se añade al juego, dicha ficha coincide en color con la del jugador, en idioma con el del dispositivo y es del tipo seleccionado
Salida Obtenida	La ficha se añade al juego, dicha ficha coincide en color con la del jugador, en idioma con el del dispositivo y es del tipo seleccionado
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 75. Descripción de la prueba 15.

P-16	Cancelar la acción de añadir una ficha al juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de la cancelación de la acción de añadir una ficha al juego
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina y que sea su turno. 3. Pulsar un hexágono gris 4. Cancelar el diálogo que muestra las fichas
Salida Esperada	El tablero se mantiene como estaba antes de pulsar el hexágono y el turno sigue siendo del jugador
Salida Obtenida	El tablero se mantiene como estaba antes de pulsar el hexágono y el turno sigue siendo del jugador
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 76. Descripción de la prueba 16.

P-17	Comprobar que no se pueden añadir más fichas cuando el jugador tiene la caja vacía
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que el jugador no puede añadir más fichas cuando ya no tiene más para añadir a la partida
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina, que sea su turno y que haya añadido 11 fichas al juego. 1. Pulsar ficha del color del jugador 2. Pulsar ficha del otro jugador 3. Pulsar fondo
Salida Esperada	El tablero de juego no muestra hexágonos grises y no aparece el diálogo con la lista de fichas disponibles de ninguna forma
Salida Obtenida	El tablero de juego no muestra hexágonos grises y no aparece el diálogo con la lista de fichas disponibles de ninguna forma
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 77. Descripción de la prueba 17.

P-18	Añadir todas las fichas al juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que el jugador puede añadir todas sus fichas al juego
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina, que sea su turno y no haber añadido ninguna ficha al juego. 1. Pulsar un hexágono gris 2. Seleccionar un insecto de la lista desplegada 3. Repetir pasos 1 y 2 once veces
Salida Esperada	Las once fichas del jugador aparecen en el tablero donde este las colocó
Salida Obtenida	Las once fichas del jugador aparecen en el tablero donde este las colocó
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 78. Descripción de la prueba 18.

P-19	Comprobar que al cuarto turno solo se puede añadir la abeja
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que el jugador solo puede añadir una abeja si está en su cuarto turno y no la había añadido antes
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina, que sea su cuarto turno y ninguna de las fichas que ha añadido al juego sea la abeja. 1. Pulsar el hexágono gris
Salida Esperada	La lista desplegada solo muestra la abeja
Salida Obtenida	La lista desplegada solo muestra la abeja
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 79. Descripción de la prueba 19.

P-20	Obtener los posibles movimientos de una ficha según su tipo
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que el jugador puede comprobar a dónde puede mover una ficha
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina, que sea su turno, que esté su abeja en juego y que la ficha que quiere mover pueda hacerlo (Comprobar reglas del juego en el ANEXO I) 1. Tocar la ficha que se quiere mover Nota: Repetir esto por cada tipo de ficha
Salida Esperada	Se muestran hexágonos con los posibles destinos de la ficha según su tipo (Reglas del juego)
Salida Obtenida	Se muestran hexágonos con los posibles destinos de la ficha según su tipo (Reglas del juego)
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 80. Descripción de la prueba 20.

P-21	Obtener los posibles movimientos de una ficha que no se puede mover
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que el jugador no puede obtener los posibles movimientos de una ficha que no se puede mover
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina, que sea su turno, que esté su abeja en juego y que la ficha que quiere mover no pueda hacerlo (Por ejemplo: Una ficha que al ser movida rompiese la colmena o que esté bloqueada por otras fichas) 1. Tocar la ficha
Salida Esperada	Los hexágonos grises que se muestran no han cambiado de posición, siguen mostrando dónde se puede añadir una ficha por primera vez
Salida Obtenida	Los hexágonos grises que se muestran no han cambiado de posición, siguen mostrando dónde se puede añadir una ficha por primera vez
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 81. Descripción de la prueba 21.

P-22	Cancelar la vista de los posibles movimientos de una ficha seleccionada para ello
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que el jugador puede cancelar la vista de los posibles movimientos de una ficha
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina, que sea su turno, que esté su abeja en juego y que la ficha que quiere mover pueda hacerlo (Comprobar reglas del juego en el ANEXO I) 1. Tocar la ficha que se quiere mover 2. Tocar una ficha enemiga
Salida Esperada	Los hexágonos grises vuelven a mostrar las posiciones donde se puede colocar una ficha por primera vez
Salida Obtenida	Los hexágonos grises vuelven a mostrar las posiciones donde se puede colocar una ficha por primera vez
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 82. Descripción de la prueba 22.

P-23	Comprobar que no se pueden mover fichas ni ver sus posibles movimientos si la abeja del jugador no está en juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que el jugador solo puede mover fichas y ver sus posibles movimientos si su abeja está en juego
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina, que sea su turno y no haber añadido su abeja al juego. 1. Pulsar una ficha que se pueda mover (ver la reglas del juego [**ANEXO**])
Salida Esperada	Los hexágonos grises no cambian de posición, siguen mostrando lugares donde el jugador puede colocar una ficha por primera vez
Salida Obtenida	Los hexágonos grises no cambian de posición, siguen mostrando lugares donde el jugador puede colocar una ficha por primera vez
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 83. Descripción de la prueba 23.

P-24	Mover una ficha
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que una ficha se puede cambiar de sitio
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina, que sea su turno, tener la abeja en juego y que la ficha a mover pueda hacerlo (Comprobar reglas del juego en el ANEXO I). 1. Tocar la ficha 2. Seleccionar un hexágono sombreado
Salida Esperada	La ficha se encuentra ahora en el lugar del hexágono elegido como destino
Salida Obtenida	La ficha se encuentra ahora en el lugar del hexágono elegido como destino
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 84. Descripción de la prueba 24.

P-25	Comprobar que tocar la zona fuera de la colmena no produce ningún efecto
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que tocar la zona exterior a la colmena no provoca ninguna acción
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina y que haya algunas fichas ya en juego. 1. Tocar la zona amarillenta de la pantalla
Salida Esperada	No ocurre nada
Salida Obtenida	No ocurre nada
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 85. Descripción de la prueba 25.

P-26	Comprobar que al ganar/perder una partida se muestra un diálogo
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de la aparición de un diálogo al final de la partida
Pasos a seguir	Prerrequisito: Haber iniciado una partida contra otro jugador o contra la máquina y que una de las abejas o las dos solo tengan un hueco vacío a su alrededor al que se puede acceder con una de las fichas puestas en juego. 1. Mover una ficha hacía ese hueco
Salida Esperada	Aparece un diálogo con el nombre del ganador o empate si se rodeó las dos abejas a la vez y la opción de aceptar el diálogo
Salida Obtenida	Aparece un diálogo con el nombre del ganador o empate si se rodeó las dos abejas a la vez y la opción de aceptar el diálogo
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 86. Descripción de la prueba 26.

P-27	Comprobar que al querer abandonar una partida finalizada sale un diálogo que se puede confirmar
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de la aparición de un diálogo al querer abandonar una partida finalizada y se puede confirmar
Pasos a seguir	Prerrequisito: Haber finalizado una partida de cualquiera de los dos tipos. 1. Pulsar el botón de retroceso del dispositivo 2. Aparece un diálogo preguntando si queremos jugar otra partida 3. Pulsar “Sí”
Salida Esperada	Se inicia una nueva partida y aparece el diálogo que indica qué jugador empieza primero
Salida Obtenida	Se inicia una nueva partida y aparece el diálogo que indica qué jugador empieza primero
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 87. Descripción de la prueba 27.

P-28	Comprobar que al querer abandonar una partida finalizada sale un diálogo que se puede rechazar
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de la aparición de un diálogo al querer abandonar una partida finalizada y se puede rechazar
Pasos a seguir	Prerrequisito: Haber finalizado una partida de cualquiera de los dos tipos. 1. Pulsar el botón de retroceso del dispositivo 2. Aparece un diálogo preguntando si queremos jugar otra partida 3. Pulsar “No”
Salida Esperada	La aplicación nos devuelve al menú principal
Salida Obtenida	La aplicación nos devuelve al menú principal
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 88. Descripción de la prueba 28.

6.4.3 Partida contra la máquina

Las pruebas que se realizaron para comprobar el correcto funcionamiento de una partida contra la máquina fueron de dos tipos. En el primero se aplicaron las mismas pruebas que cuando se comprobó el correcto funcionamiento de una partida contra otro jugador:

P-11 Acceso a una partida contra otro jugador
P-12 Iniciar una partida contra otro jugador
P-13 Abandonar una partida contra otro jugador
P-14 Cancelar el abandono de una partida contra otro jugador
P-15 Añadir una ficha al juego
P-16 Cancelar la acción de añadir una ficha al juego
P-17 Comprobar que no se pueden añadir más fichas cuando el jugador tiene la caja vacía
P-18 Añadir todas las fichas al juego
P-19 Comprobar que al cuarto turno solo se puede añadir la abeja
P-20 Obtener los posibles movimientos de una ficha según su tipo
P-21 Obtener los posibles movimientos de una ficha que no se puede mover
P-22 Cancelar la vista de los posibles movimientos de una ficha seleccionada para ello
P-23 Comprobar que no se pueden mover fichas ni ver sus posibles movimientos si la abeja del jugador no está en juego
P-24 Mover una ficha
P-25 Comprobar que tocar la zona fuera de la colmena no produce ningún efecto
P-26 Comprobar que al ganar/perder una partida se muestra un diálogo
P-27 Comprobar que al querer abandonar una partida finalizada sale un diálogo que se puede confirmar
P-28 Comprobar que al querer abandonar una partida finalizada sale un diálogo que se puede rechazar

Lo único que cambia es que en vez de iniciar una partida contra otro jugador "Jugador VS Jugador", si el dispositivo está en español o "Player VS player" si no lo está, había que iniciar una partida contra la máquina, "Jugador VS Máquina" si el dispositivo está en español, o "Player VS Machine" si no lo está.

Las pruebas del segundo tipo consistieron en echar varias partidas contra la máquina y comprobar si realizaba movimientos coherentes o se comportaba de forma extraña. Como comportamiento extraño podemos incluir:

- Mover una ficha que no puede moverse por estar bloqueada o provocar la ruptura de la colmena.
- Que una ficha acceda a una posición que según las reglas del juego no podría.
- Colocar más de una ficha por turno.
- Colocar fichas por primera vez en el juego en posiciones ilegales.

Para comprobar que no rompía ninguna de las reglas se rellenó una tabla de seguimiento según se iban echando partidas para registrar si todo iba bien o si la máquina realizaba algún movimiento ilegal. También se tuvieron en cuenta desde el punto de vista lógico los movimientos que realizaba la inteligencia artificial, es decir, si las acciones que realizaba tenían sentido para el objetivo final del juego y si no era predecible lo que iba a hacer. Por último, se anotó por cada partida quién empezaba y quién ganaba, para comprobar si tenía ventaja el jugador que empezaba primero.

Dicha tabla de seguimiento se muestra en el ANEXO IV. En ella se observa que de las 20 partidas registradas 10 las empezó el usuario y 10 la máquina, esta parte es completamente aleatoria, luego es casualidad que sean mitad y mitad. De las 10 que empezó el usuario este ganó 4, la máquina otras 4 y 2 fueron empate. Las que empezó la máquina fueron ganadas 5 por ella, 3 por el usuario y 2 fueron empate. Luego, el que empieza primero no tiene ni más ni menos ventaja, no influye en el resultado final del juego.

6.4.4 Reglas del Juego

Las pruebas que se realizaron para comprobar el correcto funcionamiento de la visualización de las reglas del juego fueron las siguientes:

P-29 Comprobar que se puede acceder a las reglas del juego
P-30 Comprobar que se puede ver la siguiente página en las reglas del juego
P-31 Comprobar que se puede ver la página anterior en las reglas del juego
P-32 Comprobar que no ocurre nada si se intenta ver la página siguiente a la última en las reglas del juego
P-33 Comprobar que no ocurre nada si se intenta ver la página anterior a la primera en las reglas del juego

Las pruebas se mostrarán de la misma forma que en apartados anteriores: número de prueba, título, dispositivos empleados para realizarla, descripción, pasos a seguir para completar la prueba, salida esperada, salida obtenida, errores detectados y solución de errores si los hubiera, y resultado de la prueba: si fue satisfactoria o no.

P-29	Comprobar que se puede acceder a las reglas del juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que se puede acceder a las reglas del juego
Pasos a seguir	Prerrequisito: Haber iniciado la aplicación. 1. Acceder a “Cómo jugar”
Salida Esperada	Se muestran las reglas del juego
Salida Obtenida	Se muestran las reglas del juego
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 89. Descripción de la prueba 29.

P-30	Comprobar que se puede ver la siguiente página en las reglas del juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que se puede acceder a la siguiente página en las reglas del juego
Pasos a seguir	Prerrequisito: Haber accedido a las reglas del juego y no estar en la última página de las instrucciones. 1. Pulsar el texto que aparece en pantalla
Salida Esperada	La imagen y el texto cambian. Muestran la siguiente explicación
Salida Obtenida	La imagen y el texto cambian. Muestran la siguiente explicación
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 90. Descripción de la prueba 30.

P-31	Comprobar que se puede ver la página anterior en las reglas del juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que se puede acceder a la página anterior en las reglas del juego
Pasos a seguir	Prerrequisito: Haber accedido a las reglas del juego y no estar en la primera página de las instrucciones. 1. Pulsar la imagen que aparece en pantalla
Salida Esperada	La imagen y el texto cambian. Muestran la explicación anterior
Salida Obtenida	La imagen y el texto cambian. Muestran la explicación anterior
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 91. Descripción de la prueba 31.

P-32	Comprobar que no ocurre nada si se intenta ver la página siguiente a la última en las reglas del juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que la página se mantiene si el usuario intenta ver la página siguiente de las reglas del juego cuando está ya en la última
Pasos a seguir	Prerrequisito: Haber accedido a las reglas del juego y estar en la última página de las instrucciones. 1. Pulsa el texto que aparece en pantalla
Salida Esperada	La imagen y el texto no cambian
Salida Obtenida	La imagen y el texto no cambian
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 92. Descripción de la prueba 32.

P-33	Comprobar que no ocurre nada si se intenta ver la página anterior a la primera en las reglas del juego
Dispositivos	Dispositivo_1, Dispositivo_2, Dispositivo_3
Descripción	Verificación de que la página se mantiene si el usuario intenta ver la página anterior de las reglas del juego cuando está en la primera
Pasos a seguir	Prerrequisito: Haber accedido a las reglas del juego y estar en la primera página de las instrucciones. 1. Pulsar la imagen que aparece en pantalla
Salida Esperada	La imagen y el texto no cambian
Salida Obtenida	La imagen y el texto no cambian
Errores detectados	Ninguno
Solución de Errores	-
Resultado de la prueba	No se han detectado errores en la prueba

Tabla 93. Descripción de la prueba 33.

7. Conclusiones y Trabajo Futuro

7.1 Conclusiones

La principal conclusión que se puede deducir de la realización de este trabajo es que se han cumplido satisfactoriamente los objetivos propuestos al inicio del proyecto:

- Se ha desarrollado una aplicación para dispositivos Android.
- Dicha aplicación permite jugar al juego elegido para ser implementado.
- Permite echar partidas contra otros jugadores utilizando el dispositivo como tablero de juego.
- Se ha desarrollado una inteligencia artificial que permite al usuario echar partidas contra el dispositivo.
- Se han creado unas reglas de juego junto con un manual de usuario que puede ser accedido desde el menú principal de la aplicación.

Desde un punto de vista más general se puede concluir que la realización de este trabajo ha supuesto el mayor reto académico al que nunca antes me había enfrentado. Suponía horas de trabajo y esfuerzo dedicados a un solo proyecto a lo largo de un cuatrimestre. Intuía que me iba a costar mucho llevar un ritmo de trabajo constante durante tanto tiempo y con un alto nivel de implicación, pero también sabía que el esfuerzo iba a merecer la pena y me iba a sentir orgullosa del trabajo realizado.

A lo largo de cada una de las fases que componen este proyecto he podido comprobar no solo cómo ponía en práctica los conocimientos adquiridos a lo largo de todo el grado sino que iba adquiriendo nuevos de igual o más valía al ser de mérito propio.

También se han valorado más todos los trabajos en equipo realizados a lo largo de varias asignaturas, pues se echaba en falta tanto en lo personal como en lo profesional alguien con quien debatir aspectos del proyecto o repartir horas de trabajo.

Pero no solo se han aprendido aspectos relacionados con el trabajo propiamente dicho, sino que se ha tenido que aprender a sacrificar días asignados al descanso, como fines de semana, y a compaginar el trabajo de fin de grado con prácticas extracurriculares.

La conclusión final es que este trabajo ha hecho que valore más mi capacidad para afrontar y resolver retos y me ha aportado experiencia muy valiosa que podré llevar conmigo a lo largo de toda mi carrera profesional.

7.2 Trabajo Futuro

Durante la realización del trabajo hubo que priorizar determinados aspectos de la aplicación sobre otros debido a la inexperiencia que se tenía de estos, el limitado tiempo del que se disponía para realizar el trabajo y que su realización no era clave para el funcionamiento básico de la aplicación.

Por ello se elaboró una lista de posibles mejoras para una segunda versión:

- Investigar sobre tecnologías específicas para la realización de videojuegos como Unity 3D, pudiendo así mejorar la experiencia del usuario con el juego.

- Mejorar notablemente la Inteligencia Artificial, empleando no solo una serie de reglas implementadas en la aplicación, sino desplazando la lógica de la Inteligencia a un servidor para poder realizar operaciones más complejas y de más coste evitando así que el rendimiento no dependa sólo del dispositivo en el que se ejecuta la aplicación.

- Corregir las pruebas fallidas que surgen al ejecutar el juego en un teléfono inteligente que, aunque permiten el correcto funcionamiento del juego, mejorarían la estética de la interfaz de usuario.

- Crear un historial de partidas para el usuario, en la que se muestren los resultados de dichas partidas y si fue contra la inteligencia artificial o contra otro usuario.

8. Bibliografía

[1] Board Game Geek (última visita: Julio 2016), Disponible en: https://boardgamegeek.com/

[2] Google Play: Hive™ - board game for two (última visita: Julio 2016), Disponible en: https://play.google.com/store/apps/details?id=com.hive&hl=es

[3] iTunes: HiveDe Tom Schulz (última visita: Julio 2016), Disponible en: https://itunes.apple.com/es/app/hive/id493894488?mt=8&ign-mpt=uo%3D4

[4] XBOX: H.i.v.e. (Hive the board game) (última visita: Julio 2016), Disponible en: https://marketplace.xbox.com/en-US/Product/Hive-Hive-the-board-game/66acd000-77fe-1000-9115-d80258550ca2

[5] Android (última visita: Julio 2016), Disponible en: https://www.android.com/intl/es_es/

[6] Joshua Mattassi Android sigue dominando el mercado de smartphones (20 Agosto 2015) (última visita: Julio 2016), Disponible en: https://www.wayerless.com/2015/08/android-sigue-dominando-el-mercado-de-smartphones/

[7] Ian Millington, "Chapter 1: Introduction" en *Artificial Intelligence for Games,* CRC Press, 2006.

[8] C. González Morcillo "*Lógica Difusa*" (última visita: Julio 2016) Disponible en: http://www.esi.uclm.es/www/cglez/downloads/docencia/2011_Softcomputing/LogicaDifusa.pdf

[9] Ian Millington, "Chapter 8: Board Games" en *Artificial Intelligence for Games,* CRC Press, 2006.

[10] B. Hughes y M. Cotterel, "Chapter 8" en *Software Project Management* 2Ed, McGraw Hill, 2002.

[11] Lunapic (última visita: Mayo 2016), Disponible en: http://www.lunapic.com/editor/

[12] MakeAppIcon (última visita: Junio 2016), Disponible en: https://makeappicon.com/

[13] Google Documentos (última visita: Julio 2016), Disponible en: https://www.google.es/intl/es/docs/about/

[14] Google Hojas de Cálculo (última visita: Julio 2016), Disponible en: https://www.google.es/intl/es/sheets/about/

[15] GitHub (última visita: Julio 2016), Disponible en: https://github.com/

[16] Google Drive (última visita: Julio 2016), Disponible en: http://www.google.es/drive/apps.html

[17] Dropbox (última visita: Julio 2016), Disponible en: https://www.dropbox.com/

[18] Board Game Geek: Hive (última visita: Julio 2016), Disponible en: https://boardgamegeek.com/boardgame/2655/hive

[19] Gen 42: Hive (última visita: Julio 2016), Disponible en: http://gen42.com/hive

[20] Hive (Game) (última visita: Mayo 2016), Disponible en: https://en.wikipedia.org/wiki/Hive_(game)

[21] Imagen de una red de hexágonos (última visita: Julio 2016), Disponible en: https://openclipart.org/image/2400px/svg_to_png/22140/Steren-Futurist-hexagons.png

[22] Imagen de hexágonos amarillos (última visita: Junio 2016), Disponible en: https://pbs.twimg.com/profile_images/672432933391704064/2OKoFTtQ.png

[23] JGraphT (última visita: Abril 2016), Disponible en: http://jgrapht.org/

[24] Repositorio GitHub de Vogella sobre un menú (última visita: Mayo 2016), Disponible en: https://github.com/vogellacompany/codeexamples-android/blob/master/com.vogella.android.inlineprogressreporting/src/com/example/android/animationsdemo/MainActivity.java

[25] Repositorio GitHub de starwheel sobre una red hexagonal (última visita: Mayo 2016), Disponible en: https://github.com/omplanet/android-hexagonal-grids

[26] Captura de pantalla obtenida del repositorio GitHub de starwheel sobre una red hexagonal (última visita: Mayo 2016), Disponible en: https://github.com/omplanet/android-hexagonal-grids/blob/master/Screenshots/Screenshot_2015-03-02-19-11-56.png

[27] M.A. Gómez Martín, B. Díaz Agudo. "*Evaluación de estrategias de razonamiento para sistemas basados en reglas*". Universidad Complutense de Madrid. (última visita: Julio 2016) Disponible en: https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwj78fqC6OPNAhVJ6RQKHf33C4IQFggjMAE&url=http%3A%2F%2Fgaia.fdi.ucm.es%2Faigaion2%2Findex.php%2Fattachments%2Fsingle%2F23&usg=AFQjCNEIwBJJIpFfB_pg5MZh8jMWO4iwUg&sig2=cMRn8tHZeE-d-l9c0nWEWA&bvm=bv.126130881,d.d24&cad=rja

[28] Gen 42: Hive Tips & Tricks (última visita: Mayo 2016), Disponible en: http://gen42.com/hive-tips

[29] D. Bolaños, A. Sierra y M. Alarcón "Capítulo 1: Fundamentos de las pruebas de software" en *Pruebas de Software y JUnit*, Prentice Hall, 2007.

[30] Imagen de las fichas del juego (última visita: Junio 2016), Disponible en: https://en.wikipedia.org/wiki/Hive_(game)#/media/File:Hive-Pieces-1.jpg

ANEXOS

ANEXO I: Reglas del juego

Introducción

"Hive" (Colmena) es un juego diseñado por John Yianni y publicado en 2001 por la compañía de juegos de mesa Gen42 Games.

Es un juego de mesa para 2 jugadores en el que cada uno de ellos tiene las mismas fichas: 11 piezas hexagonales que representan diferentes insectos: 1 abeja, 3 hormigas, 3 saltamontes, 2 escarabajos y 2 arañas. Ver Figura 45.

Figura 45. Imagen de las fichas del juego. [30]

Dichas piezas se pondrán en juego sobre un tablero imaginario formado por las propias fichas, es decir, las propias fichas formarán la colmena sobre la que se desplazarán para conseguir su objetivo.

Objetivo del juego

Para ganar el juego se debe conseguir que la abeja del oponente quede completamente rodeada a la vez que evitas que rodeen la tuya, para ello deberá tener piezas bloqueando su movimiento en sus 6 caras, sin importar a qué jugador pertenecen dichas fichas.

Figura 46. Imagen una abeja completamente rodeada.

En la Figura 46 podemos observar cómo la abeja blanca ha quedado rodeada tanto por piezas blancas como por piezas negras, incluyendo la abeja enemiga.

Inicio del juego

Un jugador, elegido al azar, coloca una de sus fichas sobre la superficie de juego y el otro jugador colocará su ficha tocando cualquier borde de la que ya está en juego. Ver Figura 47.

Figura 47. Imagen de un inicio de juego.

En la primera ronda representada en la imagen se puede observar cómo los jugadores han colocado sus piezas juntas para empezar a formar la colmena.

En las siguientes rondas los jugadores irán poniendo fichas nuevas o moviendo las que ya están en juego en turnos consecutivos, cada uno en su turno.

Cómo colocar

Mientras tenga fichas por jugar el jugador puede ir añadiéndolas a la partida, pero nunca podrá sacar de la partida fichas que ya estén en juego.

Para poner una ficha en juego, a excepción de la primera ronda, esta sólo podrá tocar piezas de su mismo color.

Figura 48. Imagen explicando la colocación de fichas del jugador blanco.

En la Figura 48 se puede ver cómo, siendo el turno de jugador blanco, sólo puede colocar fichas en los hexágonos sombreados, que son justo los huecos que rodean la colmena y solo tocan piezas blancas.

Colocación de la abeja reina

La abeja es la pieza más importante del juego, es como la reina en el ajedrez, y al contrario del resto de fichas que pueden entrar en juego cuando el jugador prefiera, esta debe entrar en cualquiera de los 4 primeros turnos del jugador. Si llegado el 4º turno el jugador no ha puesto la abeja en juego, deberá ponerla obligatoriamente.

Movimientos

Una vez la abeja esté en juego, pero nunca antes, el jugador podrá elegir entre seguir poniendo fichas en juego o mover las que ya tiene en juego. Cada tipo de ficha (cada tipo de insecto) se mueve de forma diferente y una vez puesta en juego podrá tocar piezas del jugador contrario al realizar un movimiento. Una regla muy importante a tener en cuenta a la hora de mover una ficha es que si al mover dicha ficha provoca la partición de la colmena en dos o más partes esta ficha no podrá moverse. Esta regla se ampliará en el apartado “Regla de una Colmena”.

Movimientos de la abeja

La abeja sólo puede desplazarse un espacio por turno. A pesar de que es la ficha con más límites de movimiento un simple desplazamiento puede alterar los planes del oponente.

Figura 49. Imagen explicando cómo se mueve la abeja.

La abeja de la Figura 49 sólo puede desplazarse a los dos huecos sombreados.

Movimientos del escarabajo

El escarabajo, al igual que la abeja reina, sólo puede desplazarse un espacio por turno, la única diferencia es que esta puede subirse encima de otras fichas bloqueando así su movimiento. Por lo tanto, el escarabajo podrá desplazarse tanto rodeando la colmena como por encima de ella y llegando incluso a huecos que estén completamente rodeados por otras fichas. La única forma de bloquear un escarabajo es poniendo otro encima (No hay límite de colocación de escarabajos unos encima de otros, el único límite lo pone el número de escarabajos que pueden estar en juego, 4). A la hora de colocar una ficha en juego, si un escarabajo está encima de otra ficha prevalecerá el color del escarabajo.

Figura 50. Imagen explicando cómo se mueve un escarabajo.

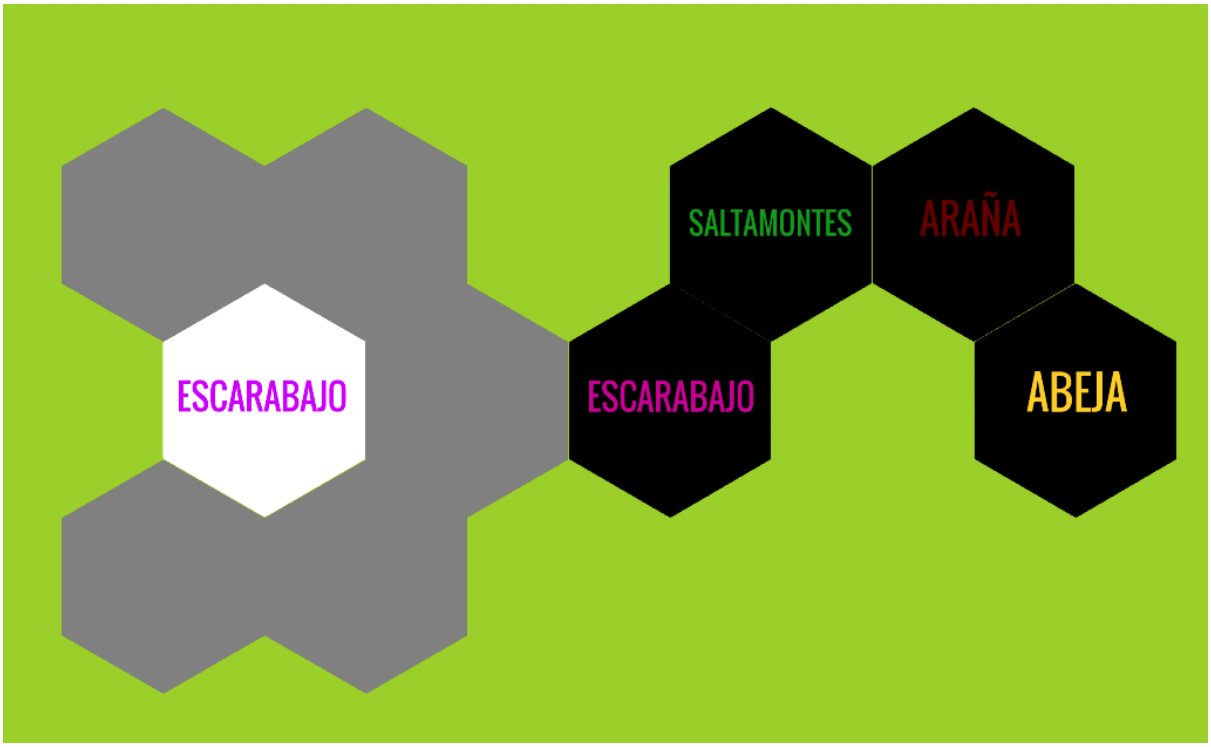

Figura 51. Imagen explicando cómo se mueve un escarabajo.

En la Figura 50 podemos observar la situación actual del juego siendo el turno del jugador blanco. En la Figura 51 se pueden ver sombreados los huecos en los que puede colocarse el escarabajo, tanto deslizándose al lado de una ficha como situándose encima suyo.

Movimientos del saltamontes

Los saltamontes se desplazan únicamente saltando por encima de otras fichas y en cualquier dirección, dándole la ventaja de llegar a huecos totalmente rodeados, al igual que al escarabajo. No podrán saltar sobre espacios vacíos.

Figura 52. Imagen explicando cómo se mueve un saltamontes.

El saltamontes de la Figura 52 solo podrá desplazarse a una de las dos zonas sombreadas, por encima de las abejas o por encima de la araña.

Movimientos de la araña

Las arañas pueden desplazarse exactamente 3 posiciones, no podrán desplazarse ni más ni menos, en el mismo sentido y sin poder retroceder.

Figura 53. Imagen explicando cómo se mueve una araña.

La araña de la Figura 53 podrá deslizarse a cualquiera de los dos huecos sombreados, pues están exactamente a tres huecos de distancia.

Movimientos de la hormiga

La hormiga es, después de la abeja, la ficha más valiosa del juego pues puede desplazarse rodeando la colmena a cualquier posición.

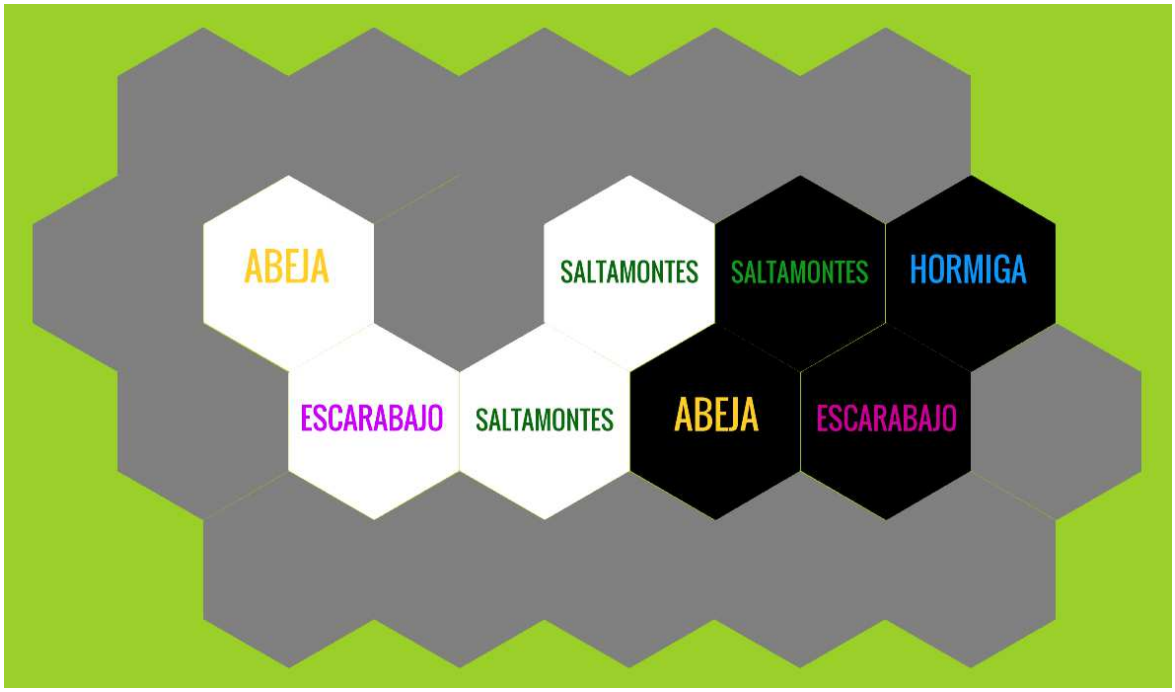

Figura 54. Imagen explicando cómo se mueve una hormiga.

La hormiga puede desplazarse en cualquiera de los huecos grises de la Figura 54.

Restricciones

Regla de una colmena

Las fichas del juego deberán estar conectadas entre sí en todo momento, no se podrá mover una ficha que provoque la ruptura de la colmena.

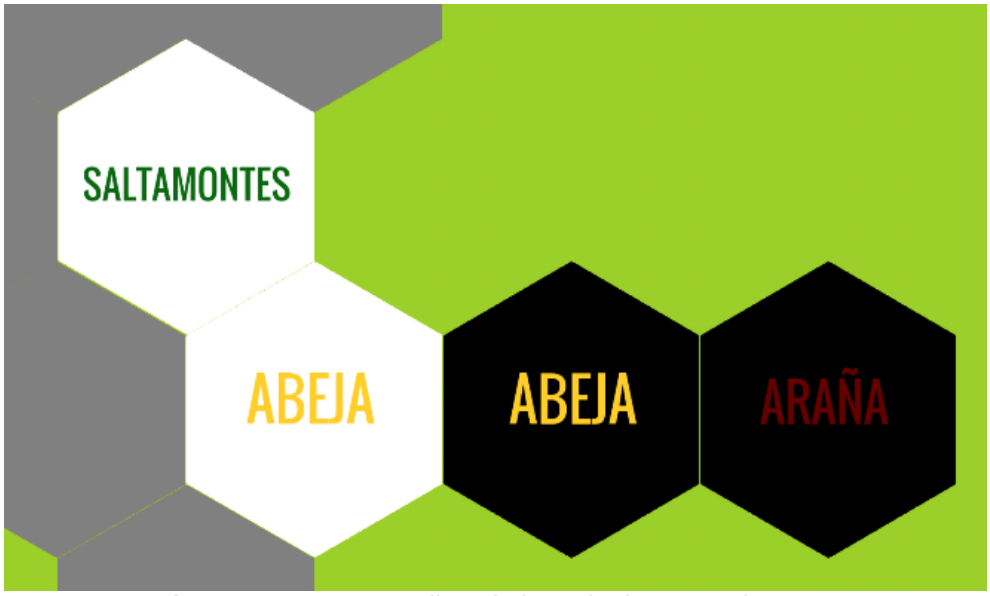

Figura 55. Imagen explicando la regla de "una colmena".

Si intentamos mover cualquiera de las dos abejas de la Figura 55 romperíamos la colmena, sin embargo, no pasaría lo mismo si movemos el saltamontes o la araña.

Libertad de movimiento

Todos los insectos, salvo los saltamontes, deberán desplazarse deslizándose, es decir, si una ficha está rodeada de tal forma que no podría moverse sin deslizarse o quiere ir a una posición a la que no puede acceder sin deslizarse, dicha ficha no podrá moverse y/o acceder a esa posición.

Figura 56. Imagen explicando la libertad de movimiento.

En la Figura 56 la abeja no puede desplazarse, al intentar hacerlo desplazaría físicamente el resto de fichas.

Figura 57. Imagen explicando la libertad de movimiento.

Sin embargo en la Figura 57 la hormiga negra puede desplazarse a todos los hexágonos sombreados menos al verde del centro pues no podría deslizarse hacia él.

No se puede mover o colocar

Cuando un jugador no puede ni colocar una ficha (porque ya tiene todas en juego o no puede colocarla de modo que no toque fichas del enemigo) ni mover una que tenga en juego pierde su turno. El juego continúa de esta forma hasta que el jugador pueda realizar una acción o una abeja quede completamente rodeada.

Final del juego

El juego termina cuando una de las dos abejas queda rodeada por completo independientemente del color de las fichas que la rodean. El jugador cuya abeja queda completamente rodeada pierde la partida.

Empate. Se puede producir de dos formas:

Cuando se coloca una ficha que provoca que las dos abejas queden a la vez completamente rodeadas por otras fichas.

Cuando los dos jugadores llegan al punto en que sólo mueven dos piezas para pasar de turno sin afectar al desarrollo del juego.

Documentación [18]-[20]

ANEXO II: Manual de instalación

Esta sección está pensada para explicar cómo instalar la aplicación creada en un dispositivo Android.

Lo primero de todo es asegurarnos que la versión Android del dispositivo permite la ejecución del juego, para ello deberá tener como mínimo la versión 4.1.2 (API 16) y como mucho la 6.0 (API 23).

El siguiente paso será comprobar que el dispositivo en el que se va a instalar la aplicación dispone de espacio suficiente para esta. La aplicación ocupa 3.06MB. No deberá cumplir ninguna otra condición.

Por último debemos desplazar el fichero Código/Hive/app/app-release.apk al dispositivo móvil. Para conseguir esto se podría subir a un servicio de alojamiento en la nube o enviarlo por correo. Una vez tenemos el fichero .apk en el dispositivo podemos iniciar la instalación pulsando sobre él y confirmar la instalación pulsando en “Instalar”. Ver Figura 58.

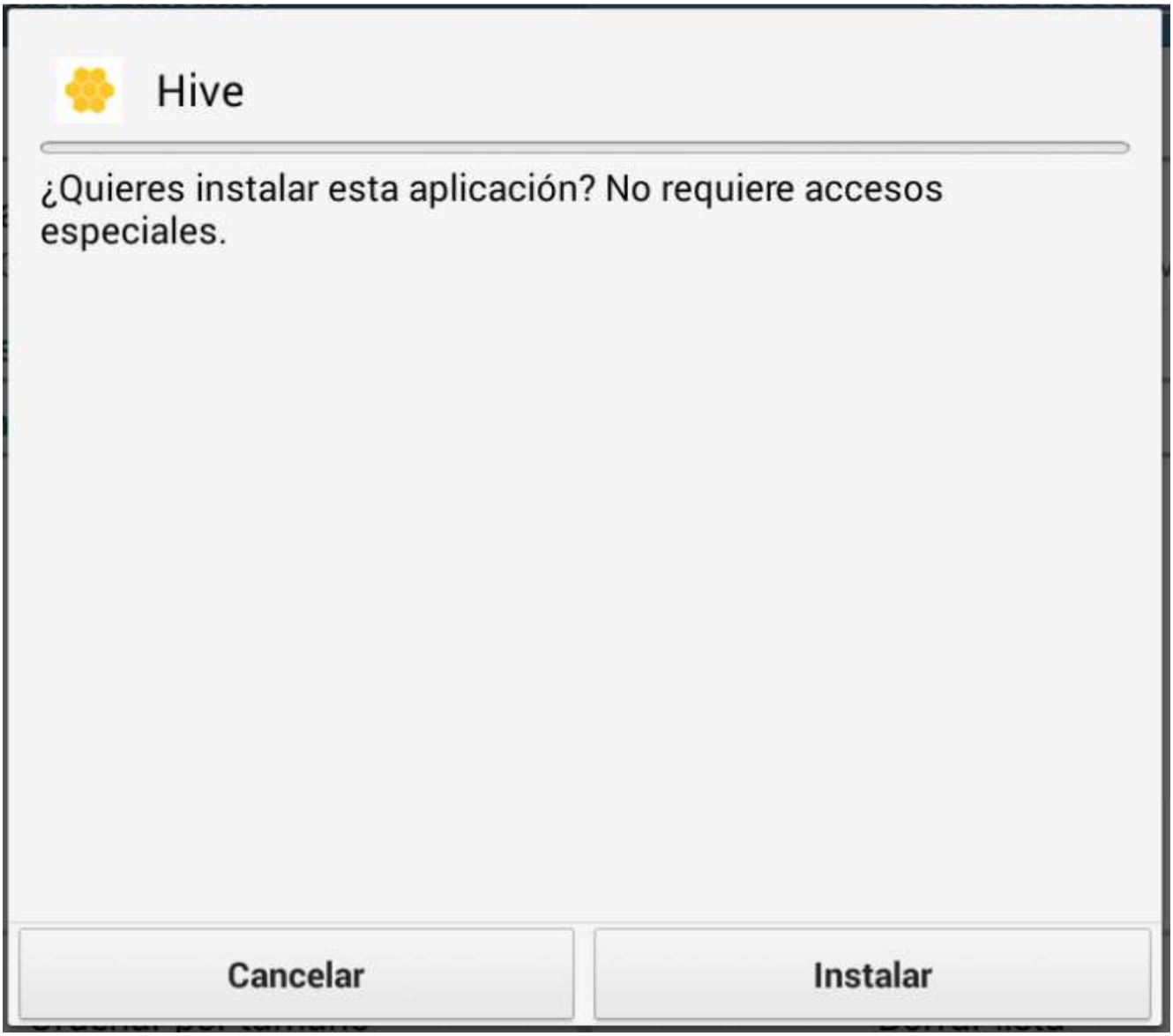

Figura 58. Imagen explicando el primer paso de la instalación.

Una vez finalice la instalación se mostrará el siguiente mensaje que aparece en la Figura 59.

Figura 59. Imagen explicando el segundo paso de la instalación.

Para iniciar el juego directamente se debe pulsar en “Abrir” y para realizar cualquier otra acción en “Hecho”.

Posibles problemas

Los dispositivos Android vienen por defecto configurados para proporcionar un mínimo de seguridad, por lo que muchas veces al intentar instalar una aplicación sin utilizar la plataforma especificada para ello, Play Store, aparece el mensaje que muestra la Figura 60.

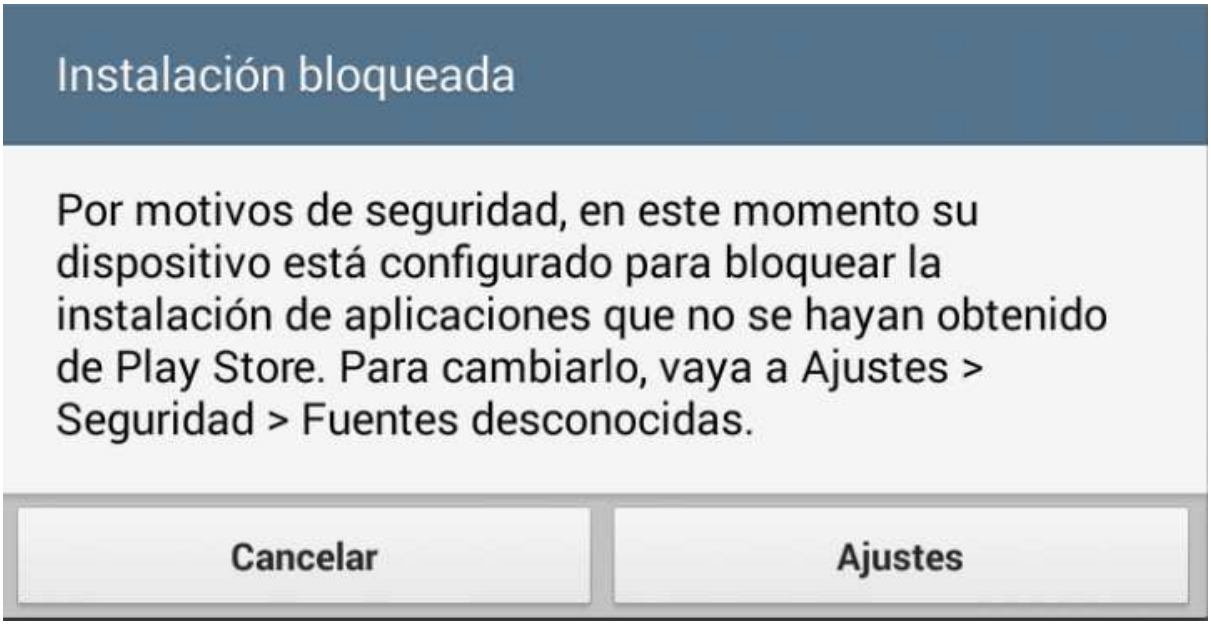

Figura 60. Imagen explicando el primer paso de cómo evitar el bloqueo de la instalación.

La aplicación a instalar es completamente segura, no contiene ningún archivo maligno y ni siquiera necesita conexión a Internet, por lo tanto deberemos ir a la dirección indicada en el mensaje y marcar la siguiente casilla que aparece señalada en la Figura 61.

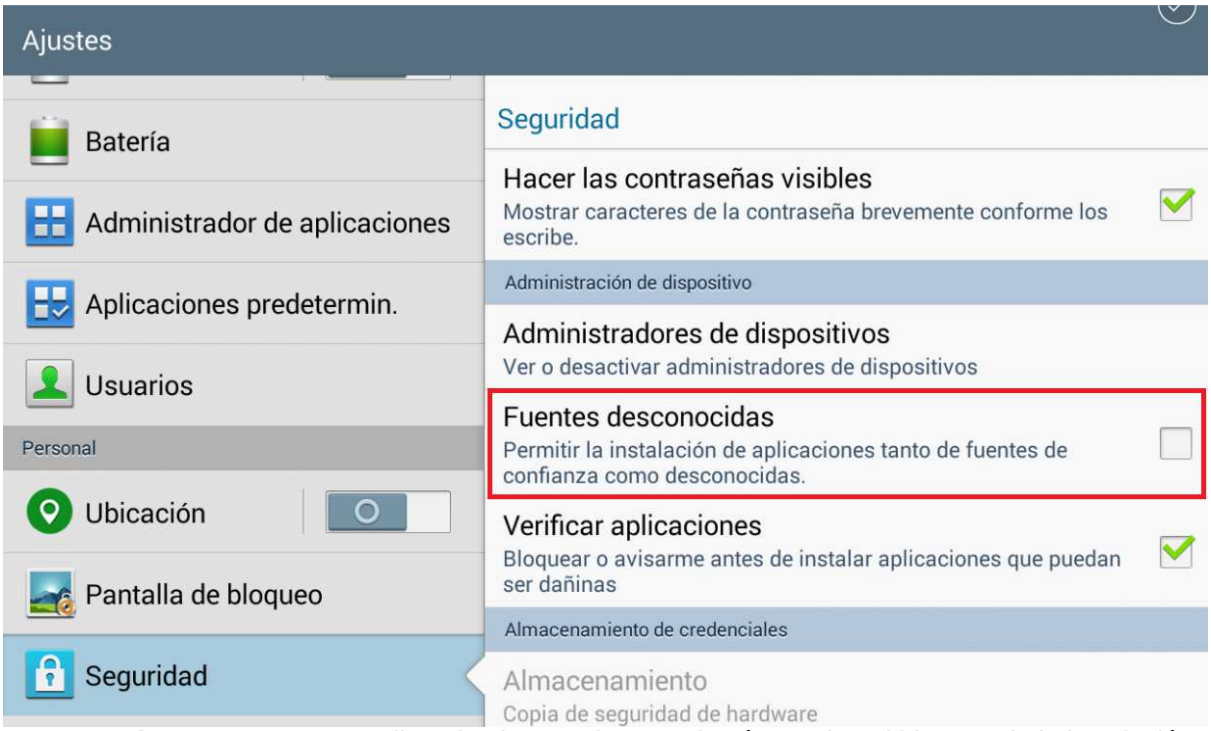

Figura 61. Imagen explicando el segundo paso de cómo evitar el bloqueo de la instalación.

Si queremos mantener esa casilla desmarcada una vez hayamos instalado la aplicación debemos mantener seleccionado “Permitir solo esta instalación” en el mensaje que aparecerá al marcar la casilla.

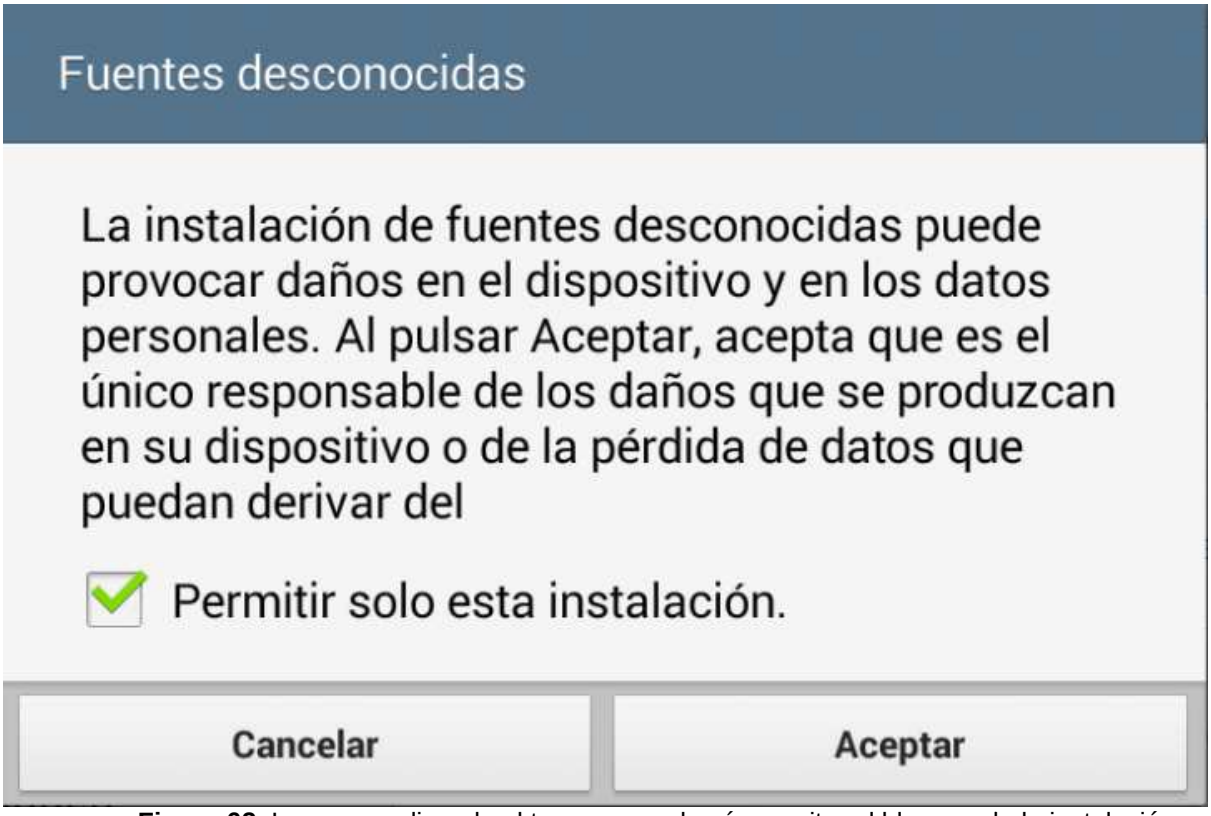

Figura 62. Imagen explicando el tercer paso de cómo evitar el bloqueo de la instalación.

El siguiente paso será confirmar la instalación y seguir con el flujo normal indicado en la introducción de la sección, es decir, al aceptar mostrará el mensaje que aparece en la Figura 58 en la que te pregunta si quieres instalar la aplicación.

ANEXO III: Manual de Usuario

Una vez se han leído y entendido las reglas de juego [ANEXO I] y se ha instalado la aplicación [ANEXO II], se debe comprender cómo interactuar con la aplicación para realizar las distintas acciones que el usuario quiere hacer.

Para acceder a la aplicación basta con pulsar el icono con el nombre de la aplicación (Hive) que apareció al ser instalada. Ver Figura 63.

Figura 63. Logo de la aplicación.

Una vez nos encontramos dentro de la aplicación debemos entender su estructura. Ver Figura 64.

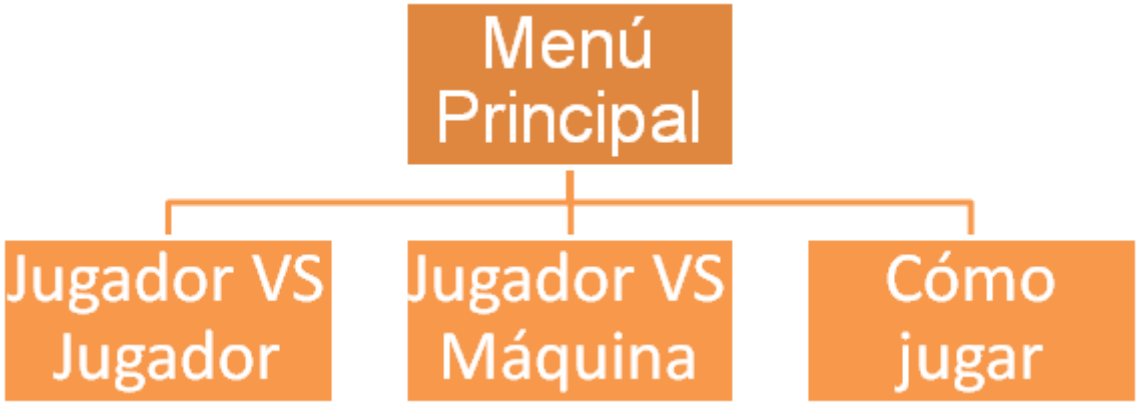

Figura 64. Estructura de las partes de la aplicación.

El menú principal tiene el aspecto que muestra la Figura 65 si el dispositivo se encuentra en español.

Figura 65. Captura de pantalla del menú en español.

Y está en cualquier otro idioma aparecerá el menú como muestra la Figura 66.

Figura 66. Captura de pantalla del menú en inglés.

La aplicación está ideada de tal forma que si el dispositivo se encuentra en español, el juego se mostrará en ese idioma, y si no lo está se mostrará en inglés.

Jugador VS Jugador

Para acceder a una partida contra otro jugador basta con pulsar en “Jugador VS Jugador”. La aplicación elige aleatoriamente quién empieza primero a jugar por lo que al iniciar una partida lo primero que aparecerá será un mensaje indicando qué color empieza a jugar y el usuario deberá confirmar el diálogo para que la partida pueda dar comienzo. Ver Figura 67.

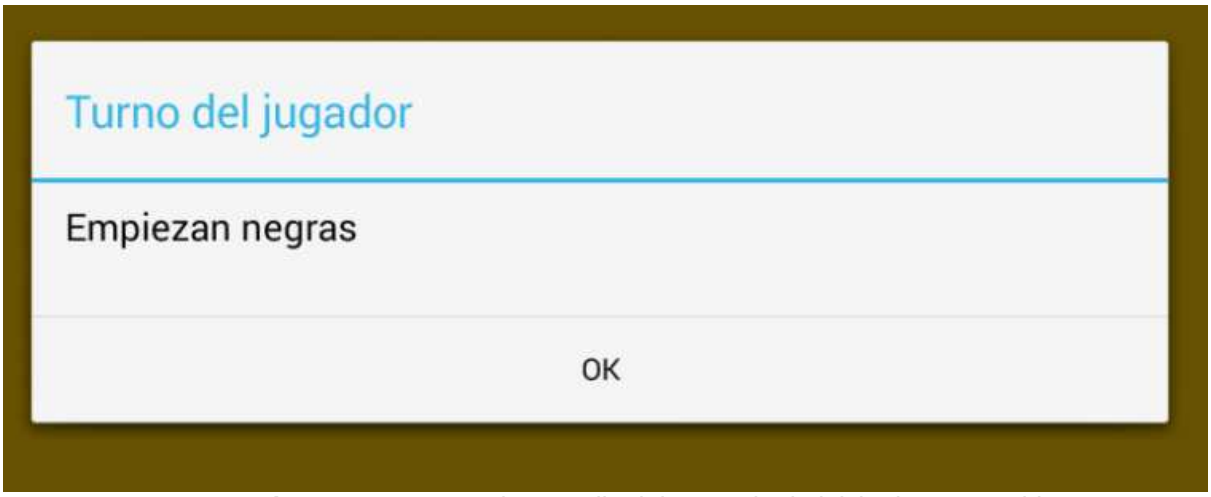

Figura 67. Captura de pantalla del mensaje de inicio de una partida.

Una vez se ha confirmado la acción, se mostrará por pantalla un hexágono gris indicando que esa es la única posición en la que el jugador que empieza puede colocar una ficha. Ver Figura 68.

Figura 68. Captura de pantalla del inicio de una partida.

Para colocar una ficha en dicho hexágono el jugador deberá pulsar sobre él, si pulsa fuera de él no ocurriría nada:

Figura 69. Captura de pantalla de las fichas que tiene el jugador en la caja.

Al pulsar sobre el hexágono se mostrará una lista de todas las fichas que tiene el jugador y que no ha colocado sobre el tablero (Ver Figura 69). Para colocar una ficha en el tablero se debe seleccionar un insecto de la lista. Si se quiere cancelar la acción se debe pulsar el botón de retroceso.

Al colocar o mover una ficha del juego pasa el turno al siguiente jugador, por lo que los hexágonos sombreados muestran los únicos huecos sobre los que puede colocar una ficha por primera vez. Dicha acción se puede ver en la Figura 70, el jugador negro colocó un saltamontes y ahora es el turno del jugador blanco.

Figura 70. Captura de pantalla del segundo turno de una partida.

Para mover una ficha puesta en juego solo habrá que pulsar sobre ella y los hexágonos sombreados pasarán a mostrar en vez de las posiciones donde puede añadir una ficha por primera vez al juego las posiciones (si las tuviera) a las que se puede mover esa ficha.

Por ejemplo, en la Figura 71 tiene el turno las blancas, y puede optar por colocar una por primera vez en la zona sombreada:

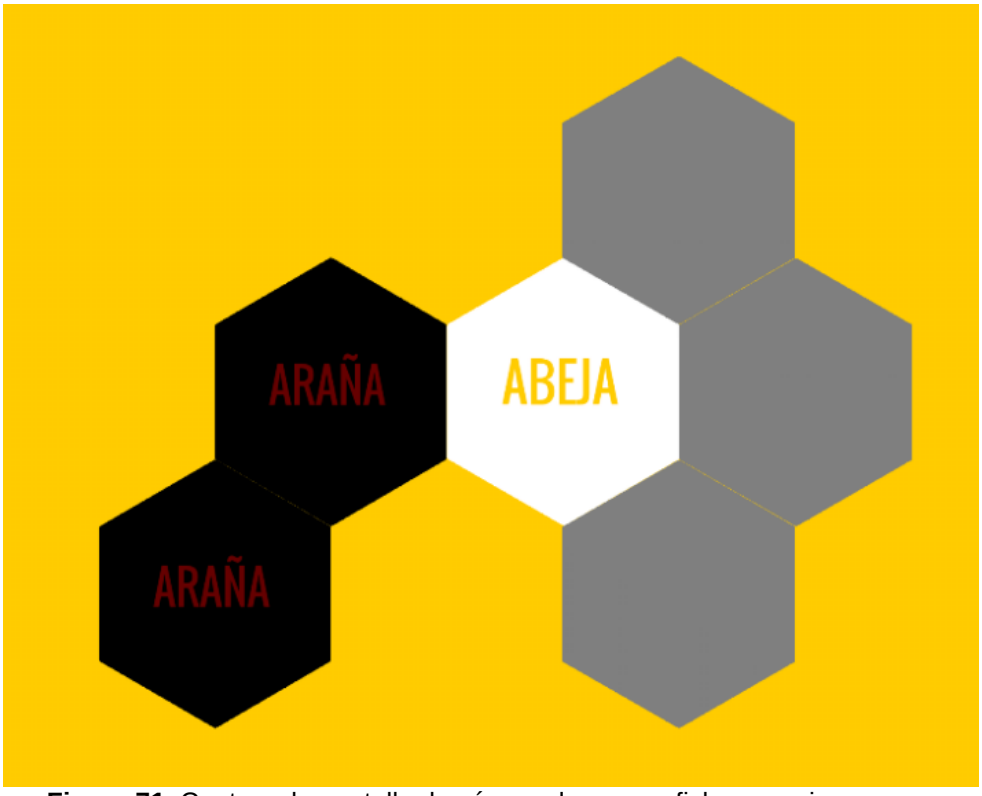

Figura 71. Captura de pantalla de cómo colocar una ficha por primera vez.

O mover la abeja, por lo que deberá pulsarla y los hexágonos mostraran sus posibles destinos. Ver Figura 72.

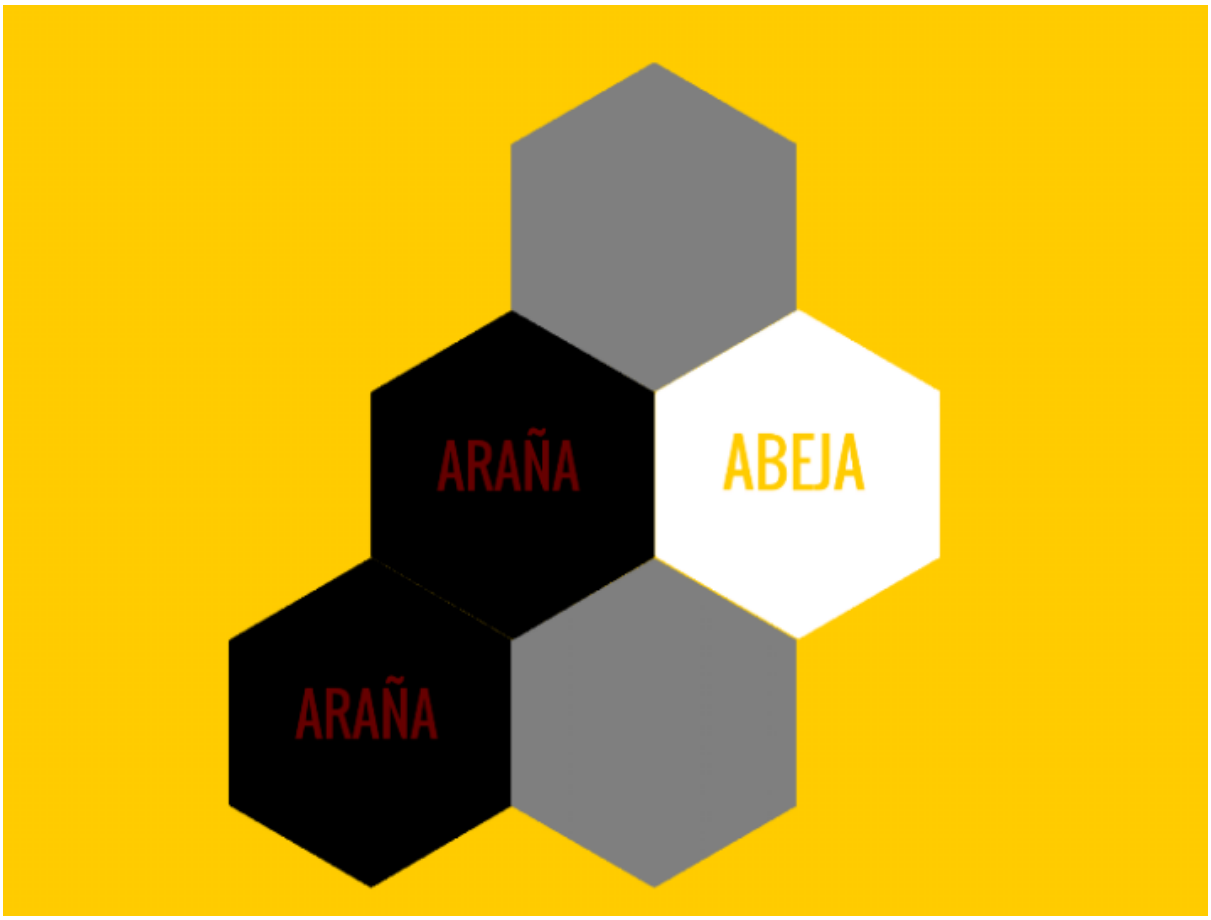

Figura 72. Captura de pantalla mostrando los posibles movimientos de una abeja.

Para anular la acción deberá tocar una ficha del enemigo y los hexágonos grises pasarán a mostrar de nuevo posiciones en los que el usuario puede colocar una ficha por primera vez.

Si se quiere abandonar una partida bastará con pulsar el botón de retroceso y confirmar el mensaje que se mostrará (Ver Figura 73). Si te arrepientes y no quieres salir o le has dado por error siempre puedes cancelar esta acción:

Figura 73. Captura de pantalla mostrando el mensaje de confirmación de abandono de una partida.

Cuando una abeja quede rodeada o las dos abejas queden rodeadas a la vez se indicará por pantalla quién es el jugador ganador (o empate si se atrapó a las dos abejas) (Ver Figura 74). Al confirmar el mensaje se mostrará cómo han quedado colocadas las fichas del juego en el tablero sin posibilidad de seguir moviéndolas.

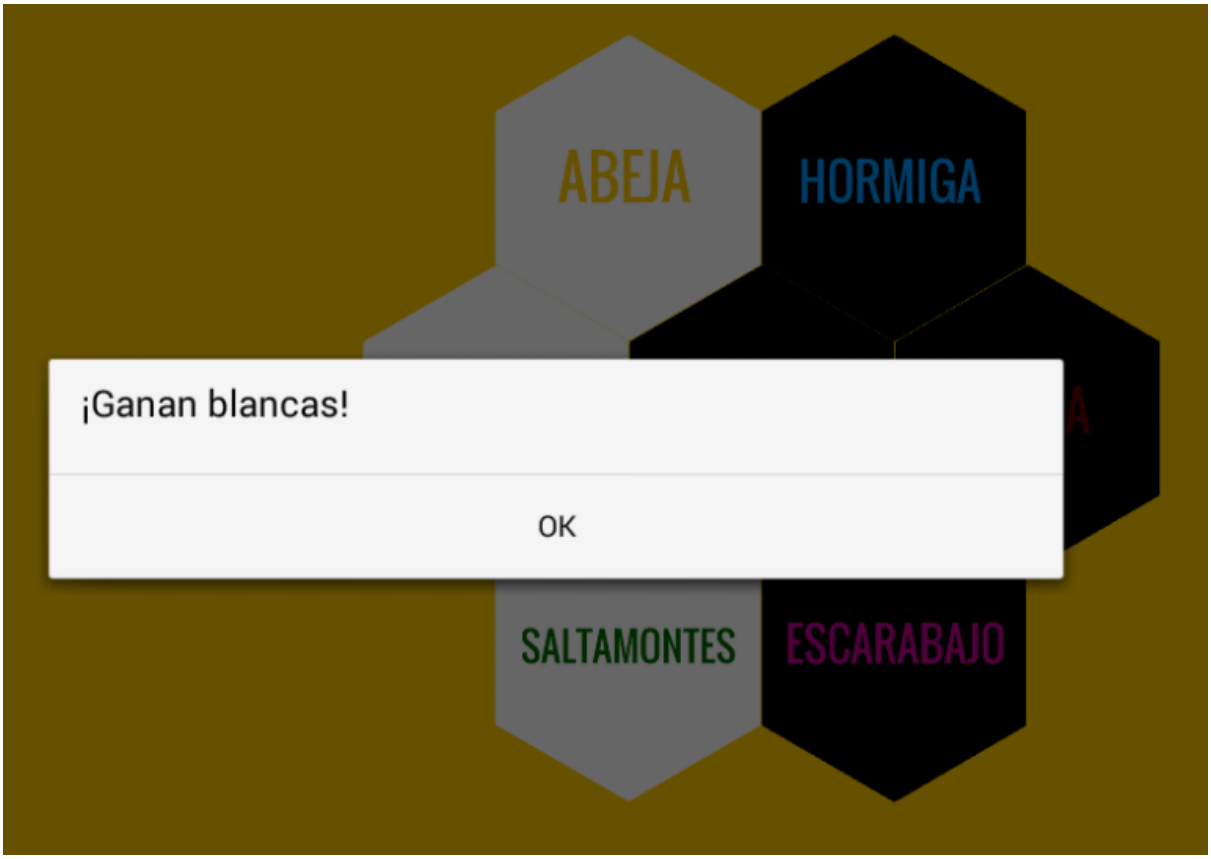

Figura 74. Captura de pantalla mostrando el mensaje que indica el ganador.

Una vez ha acabado la partida el usuario deberá pulsar el botón de retroceso y aparecerá un mensaje preguntando si desea jugar otra partida, tal y como muestra la Figura 75.

Figura 75. Captura de pantalla mostrando el mensaje que pregunta si se quiere jugar otra partida.

También hay que tener en cuenta situaciones especiales como las que se nombran a continuación:

- Si la caja está vacía, es decir, si ya ha puesto en juego todas las fichas, en su turno no aparecerán hexágonos sombreados, sólo cuando se seleccione una ficha que se pueda mover.

- Si un usuario no puede ni mover ni colocar una ficha se indicará mediante un diálogo que el turno pasa al siguiente jugador. Pulsando “Siguiente jugador” se pasa el turno.

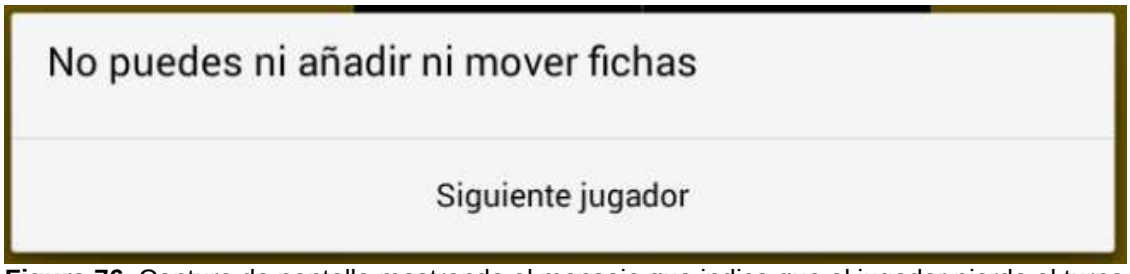

Figura 76. Captura de pantalla mostrando el mensaje que indica que el jugador pierde el turno.

Jugador VS Máquina

Para echar una partida contra el dispositivo habrá que elegir la opción “Jugador VS Máquina” e interactuar con el dispositivo de la misma forma que si fuera contra un usuario real. Todos los mensajes que aparecen durante ese modo de juego se aplican también a este.

Cuando el usuario coloque o mueva una ficha la máquina responderá con otro movimiento. El tiempo de respuesta variará, la máquina tarda más en colocar una ficha por primera vez que en mover una que ya esté puesta en juego.

Hay que tener en cuenta que mientras la máquina esté pensando qué acción realizar el dispositivo queda bloqueado, es decir, puede parecer que el juego se ha estropeado pero en verdad no es así, la máquina está pensando.

Cómo Jugar

Para ver las reglas del juego habrá que pulsar en “Cómo jugar”. Esta acción hará que se muestre lo que aparece en la Figura 77 por pantalla (Si el dispositivo está en español).

“Hive” (Colmena) es un juego diseñado por John Yianni y publicado en 2001 por la compañía de juegos de mesa Gen42 Games.

Es un juego de mesa para 2 jugadores en el que cada uno de ellos tiene las mismas fichas: 11 piezas hexagonales que representan diferentes insectos: 1 abeja, 3 hormigas, 3 saltamontes, 2 escarabajos y 2 arañas.

Figura 77. Captura de pantalla mostrando la primera página de las reglas.

Es la primera página de las reglas del juego. Para acceder a la siguiente página se debe pulsar en el párrafo. Esto provocará que la imagen y el texto que se muestran cambien enseñando la segunda página de las instrucciones. Ver Figura 78.

Dichas piezas se pondrán en juego sobre un tablero imaginario formado por las propias fichas, es decir, las propias fichas formarán la colmena sobre la que se desplazarán para conseguir su objetivo.

OBJETIVO DEL JUEGO

Para ganar el juego se debe conseguir que la abeja del oponente quede completamente rodeada a la vez que evitas que rodeen la tuya, para ello deberá tener piezas bloqueando su movimiento en sus 6 caras, sin importar a qué jugador pertenecen dichas fichas.
En esta imagen podemos observar como la abeja blanca a quedado rodeada tanto por piezas blancas como por piezas negras, incluyendo la abeja enemiga.

Figura 78. Captura de pantalla mostrando la segunda página de las reglas.

Y para acceder a la anterior se debe pulsar la imagen. Por lo tanto la dinámica para ver las instrucciones es la siguiente:

- Ver la página siguiente: Pulsar en el párrafo.
- Ver la página anterior: Pulsar en la imagen.

ANEXO IV: Partidas contra la inteligencia artificial

Este anexo contiene una tabla con información sobre las partidas jugadas contra la máquina. La tabla contiene Quién empezó, quién ganó, observaciones si las hubiese sobre el comportamiento de la máquina y el nombre del archivo que contiene una captura sobre el estado final de la partida. Dichas capturas se encuentra dentro del CD que contenía esta memoria en la carpeta llamada "Capturas de fin de partida".

El color blanco representa el usuario y el negro a la inteligencia artificial.

Empezó	Ganó	Observaciones	Captura de pantallas
Blancas	Negras	La hormiga no bloqueó la abeja	Captura_1
Negras	Negras		Captura_2
Blancas	Negras	Prefirió salvar su abeja que atacar la mía. Solo le quedaba 1 movimiento para ganar	Captura_3
Blancas	Blancas		Captura_4
Blancas	Blancas		Captura_5
Negras	Negras		Captura_6
Blancas	Blancas		Captura_7
Blancas	Empate		Captura_8
Negras	Blancas		Captura_9
Blancas	Negras		Captura_10
Blancas	Negras		Captura_11
Blancas	Empate		Captura_12
Negras	Empate		Captura_13
Negras	Blancas		Captura_14
Negras	Empate		Captura_15
Negras	Negras		Captura_16
Blancas	Blancas	Pudo no atacar a su abeja pero lo hizo y perdió	Captura_17
Negras	Negras		Captura_18
Negras	Blancas		Captura_19
Negras	Negras		Captura_20

Tabla 94. **Partidas contra la inteligencia artificial.**