Desarrollo de una Plataforma para Creación e Investigación en Videojuegos Multijugador de Deducción

Víctor García Rodríguez y Jesús Menéndez Montejo

FACULTAD DE INFORMÁTICA
DEPARTAMENTO DE INGENIERÍA DEL SOFTWARE
E INTELIGENCIA ARTIFICIAL
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Grado en Ingeniería Informática

Madrid, 8 de junio de 2018

Director: Prof. Dr. Federico Peinado Gil
Codirector: Dr. Nahum Álvarez Ayerza

Agradecimientos

En primer lugar queremos dar las gracias a Fede, nuestro director, por permitirnos realizar este Trabajo Fin de Grado. A nosotros nos encantan los videojuegos y siempre habíamos querido hacer uno. Este proyecto de alguna forma nos ha brindado la oportunidad de hacerlo, aunque sea un juego pequeño. También agradecemos a Nahum las revisiones sobre nuestra memoria y el darnos consejos frecuentes a pesar de estar en Japón.

Agradecerle a Laura, la encargada del arte del juego, que a pesar de lo atareada que se encuentra en su día a día haya podido sacar algo de tiempo para realizar los diseños de los jugadores y la portada del juego.

A todas las personas que se pasaron por ambas sesiones de pruebas, os agradecemos también el esfuerzo en estas fechas próximas al final del curso en las que todos estamos con el agua al cuello. Vuestras impresiones nos ayudaron mucho, de verdad.

Y por último no queremos olvidarnos de agradecer a nuestros familiares y amigos por apoyarnos en la elección de esta carrera que francamente nos encanta.

¡Muchas gracias a todos!

Índice general

Resumen

Crear juegos es a menudo un trabajo complejo que requiere capacidad de abstracción, análisis y meticulosidad. Hoy día el diseño de juegos es una disciplina académica que se imparte y se investiga en universidades de todo el mundo. En el caso concreto de los videojuegos multijugador de deducción, este trabajo se convierte en todo un desafío.

Este proyecto trata de simplificar la creación e investigación en este tipo de videojuegos proponiendo una herramienta de asistencia para los diseñadores profesionales y para los académicos, que incluso permitirá conectar *bots*, también llamados jugadores virtuales, en un futuro a través de su API.

El objetivo principal es el desarrollo de esta plataforma informática en Unity, *DeductionLab*, que deberá ser altamente configurable. Como parte de los objetivos está también la creación de un juego de ejemplo, llamado *MiceMaze*, que ilustra la capacidad de la plataforma para realizar experimentos de forma cómoda y rápida.

En el trabajo se estudia el mercado de los juegos, digitales o no, pertenecientes al género de los detectives, la lógica y la deducción. Para el desarrollo del software se utiliza una metodología ágil inspirada en Scrum, con reuniones de seguimiento de todo el equipo cada dos semanas, así como unas herramientas adecuadas tanto para implementar como para gestionar el proyecto.

Como validación se realizan dos sesiones de prueba con usuarios reales. Esta realimentación permite corregir el enfoque del trabajo y sobre todo mejorar el funcionamiento de la herramienta. El curso académico se ha dividido en tres grandes hitos en los que presentar el trabajo realizado y dejar que sea discutido y evaluado por los miembros del equipo.

Como conclusión se ha obtenido una primera versión estable de esta plataforma, gratuita y de código libre, que ayudará a todo aquel interesado en explorar el videojuegos multijugador de deducción.

Palabras clave

Diseño de Juegos, Estudio de Juegos, Desarrollo de Videojuegos, Informática del Entretenimiento, Interacción Persona-Ordenador, Ingeniería del Software, Metodologías Ágiles, Negociación, Argumentación, Lógica, Detectives

Capítulo 1

Introducción

La creación de juegos no suele resultar una tarea tan sencilla y divertida como puede parecer *a priori*. El trabajo de un diseñador de juegos requiere capacidad de análisis y abstracción, razonamiento sistemático y rigor en la experimentación y documentación de prototipos, entre otras aptitudes. Hoy día el estudio de los juegos, su diseño y los distintos aspectos de su desarrollo son verdaderas disciplinas académicas que las universidades de todo el mundo incluyen en sus programas docentes y en sus planes de investigación.

Si además hablamos de videojuegos multijugador con una componente de deducción u otro tipo de razonamiento lógico, así como de otros comportamientos sociales como la negociación o el intercambio de información -falible y hasta insincero- entre jugadores, la tarea del diseñador se convierte en todo un desafío intelectual que requiere invertir mucho esfuerzo en un proceso largo y tedioso de prueba y error.

Este trabajo parte del concepto de un Trabajo Fin de Grado anterior y lo lleva a la práctica, abordando los problemas de la creación e investigación en videojuegos multijugador de deducción proponiendo una herramienta que ayude tanto a los diseñadores profesionales como a los académicos a crear, analizar y probar nuevos juegos, así como a conectarlos en un futuro con jugadores virtuales creados mediante Inteligencia Artificial (IA).

1.1. Objetivos

El propósito de este proyecto, como ya indica su título, es el desarrollo de una plataforma software que ayude a estudiosos y profesionales del sector en materia de creación e investigación en videojuegos multijugador de deducción. Dicha plataforma estará orientada a la creación de juegos en Unity, posiblemente el entorno de desarrollo de videojuegos más popular actualmente, será muy flexible y adaptable mediante el uso de ficheros de configuración, y permitirá la realización de múltiples experimentos de manera cómoda y rápida.

Como objetivos necesarios para alcanzar esta meta, nos planteamos los siguientes:

1. Estudio del mercado de los juegos del género de la lógica y la deducción, o simplemente de detectives y misterio en general, sean digitales o no. Se revisarán sus características y mecánicas principales, para conocer mejor el dominio escogido para este trabajo.

2. Estudio de las plataformas software que permitan experimentar con jugadores humanos y también con *bots* programados mediante IA y conectados de manera directa al juego.

3. Desarrollo completo (especificación, análisis, diseño, implementación y pruebas) de una plataforma para la creación e investigación de este género de videojuegos, utilizando una metodologías y unas herramientas adecuadas para un proceso de desarrollo lo más sencillo y eficaz posible. Esto incluye el aprendizaje del entorno de desarrollo a utilizar, Unity y C#, mediante su documentación oficial y algunos tutoriales, así como su API especializada en el desarrollo de videojuegos multijugador.

4. Creación de un videojuego multijugador de deducción de ejemplo que ilustre el funcionamiento y el potencial de la plataforma anterior.

5. Experimentación con el citado videojuego de ejemplo en una sesión de pruebas con usuarios reales, y validación por tanto de la utilidad de la plataforma desarrollada.

1.2. Estructura del trabajo

Tras explicar el contexto y los objetivos de este trabajo en el presente capítulo, exponemos a continuación cual es la estructura del resto de la memoria. El Capítulo 2 revisa el estado de la técnica en IA, los entornos de desarrollo y herramientas existentes, así como algunos títulos del género de deducción que se han tomado como referencia para el desarrollo de nuestro propio juego. En el Capítulo 3 se presenta la metodología, el plan de trabajo general y las herramientas relacionadas con la gestión del proyecto que han sido utilizadas durante este curso. La especificación y los requisitos software que servirán de base para el desarrollo se exponen en detalle en el Capítulo 4. El Capítulo 5 contiene el grueso de la contribución de este trabajo, ya que es donde se detalla todo el análisis, diseño e implementación de la plataforma. Prácticamente lo desarrollado es común al juego de ejemplo que se ha creado para demostrar el funcionamiento de la herramienta. Los resultados de las dos sesiones de prueba realizadas con usuarios reales se recogen en el Capítulo 6. Finalmente, el Capítulo 7 presenta nuestras conclusiones tras la realización de este proyecto, cuyos resultados consideramos satisfactorios y generadores de varias líneas de trabajo futuro de sumo interés para consolidar la plataforma que acaba de ser puesta en marcha.

Capítulo 2

Estado de la cuestión

Hoy en día la inteligencia artificial (Hoffman, 2015) todavía está lejos de poder simular el comportamiento del ser humano. No obstante, los avances a lo largo de los últimos años ponen de manifiesto que día a día se mejora la capacidad de los ordenadores para reproducir la mente humana. Alcanzar una inteligencia artificial "autosuficiente" es algo que aún no se ha conseguido, hablamos de IA que sean capaces de discernir cuando les beneficia mentir al jugador o a otra IA para conducirles al error o encubrir sus actos. Este tipo de IA debería tomar estas decisiones en base al desarrollo del juego en lugar de tener una serie de opciones predefinidas en el código.

Nuestra idea es desarrollar una plataforma que proporcione a investigadores de IA en experimento que puedan configurar y así hagan las pertinentes pruebas a sus "creaciones". Nos hemos centrado en los juegos de deducción. Sería un gran avance en el mundo de los videojuegos el hecho de que un NPC te llegara a mentir para conseguir sus fines. En Detroit: Become Human (Quantic Dream, 2018), juego del género de películas interactiva, hay un momento en el que el jugador (que es un androide) debe obtener información de un androide que debe reactivar. Para ello dispone de un par de opciones pero una de ellas consiste en no activar sus ojos una vez ha sido activado e imitar la voz de su líder lo cual hace que el androide se confíe y suelte la información. Que una IA decidiera hacer eso por si misma y no por simple selección de las opciones que tiene disponibles para realizar sería un gran avance en el mundo de la inteligencia artificial en videojuegos. Los agentes que mienten para conseguir sus fines no es algo novedoso pero si que es algo que sólo se encuentra en un pequeño campo de todos los que hay en el mundo de la IA.

Por ello nos hemos centrado en los juegos de deducción. Que está relacionado con el hecho de descubrir mentiras de otros jugadores. Un articulo (Nide y Takata, s.f.)interesante relacionado con esto que habla sobre un sistema para que las inteligencias artificiales descubran quien es el hombre lobo del clásico juego de mesa Los hombres lobo de Castronegro (Dimitry Davidoff, Philippe des Pallières y Hervé Marly, 2001) o al menos a una versión de este.

Sobre los juegos de deducción hablaremos en este capítulo así como el motor principal que usaremos para desarrollar el proyecto. Además hablaremos de nuestro punto de partida, un TFG de un curso anterior a este.

2.1. Juegos similares o relacionados

Para crear una herramienta capaz de crear juegos basados en las mecánicas del engaño y la deducción concluimos que los juegos tendrían que tener dos fases, una de simulación para que se pudiesen cometer los "crimenes" y otra de interrogatorio para encontrar a los culpables.

Debido a que el juego tiene una parte relacionada con interrogatorios buscamos juegos de deducción que se asemejaran al nuestro. Como en el mundillo de los juegos de mesa hay muchos más juegos de deducción o el típico "descubre al asesino" nos centramos en juegos de mesa en lugar de videojuegos. Algunos de estos juegos son:

2.1.1. El Club de los Martes

El club de los Martes (Guerrero, 2007) es un juego narrativo en el que de tres a seis jugadores asumen el papel de "detectives de sillón" y uno el de anfitrión en la época del Londres Victoriano. A través de la narración y la deducción, deberán resolver el crimen que les proponga su anfitrión, respondiendo a las preguntas, quien, cómo, cuándo, dónde y por qué. Es un juego que pone a prueba tanto la perspicacia como la intuición y, sobretodo, la inteligencia lógica y deductiva. El anfitrión desafía al resto de miembros de El Club de los Martes a encontrar una solución que casi nunca está cerca o al alcance de la mano. Las partidas suelen durar entre 1 y 2 horas.

Este juego se asemeja bastante al que queremos desarrollar en diversos aspectos, como la diferenciación entre dos tipos de jugadores. En nuestro juego tendríamos los ratones y al científico. También tenemos un interrogatorio que se estructura con preguntas de quien (otros ratones), donde (alguna baldosa del laberinto) y acción a realizar (comer queso o romper una pared).

Figura 2.1: Portada del juego de mesa .El Club de los Martes"

2.1.2. Sherlock Holmes Detective Asesor

Sherlock Holmes Detective Asesor (Raymond Edwards, 1900) es un juego cooperativo donde los jugadores intentan resolver casos misteriosos interrogando sospechosos, consultando periódicos y recorriendo las calles en busca de indicios. Una vez hayas completado tu investigación, compararás tus habilidades detectivescas con las del propio maestro de detectives, Sherlock Holmes.

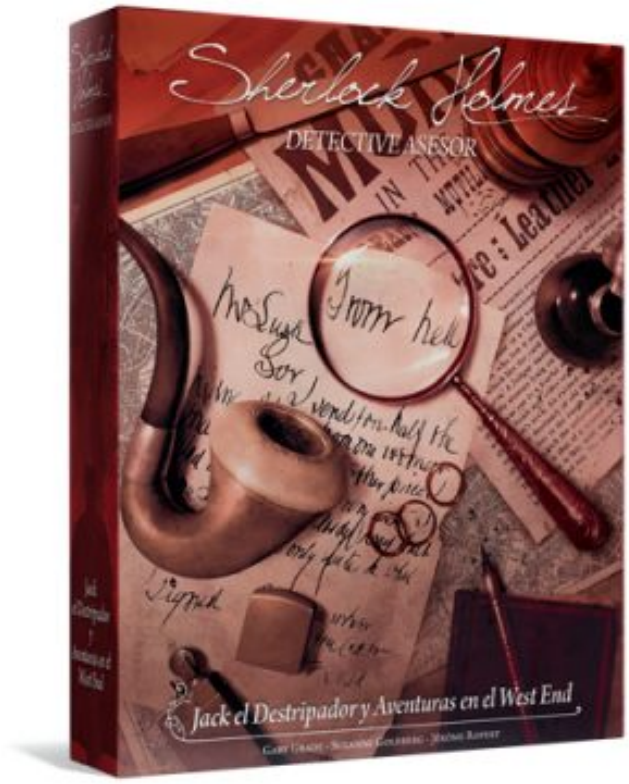

Figura 2.2: Portada del juego de mesa "Sherlock Holmes Detective Asesor"

2.1.3. Cluedo

El Cluedo (Pratt, 1949) es un juego que consta de un tablero en el que se muestran las habitaciones, corredores y pasajes secretos de una casa ingles de campo. También trae peones que representan a los personajes, piezas de armas en miniatura, varios dados, tres sets de cartas y las notas del detective. Los jugadores deberán descubrir quien llevó a cabo el asesinato de uno de los personajes del juego siguiendo las pistas que se encuentran en el tablero del juego.

Este juego clásico se podría decir que fue nuestro “punto de partida” pues cuando se nos ocurría algún elemento nuevo que añadir al juego lo pensábamos en el contexto de que fuera Cluedo ya que es un clásico en el mundo detectives. Algunas mecánicas de este juego requieren de ir eliminando posibilidades hasta que sólo quede la más evidente, y si todo lo hemos hecho bien descubriremos de manera satisfactoria quien es el asesino. Estas mecánicas y similares han sido implementadas en la fase del interrogatorio.

Figura 2.3: Portada del juego de mesa Çluedo"

2.1.4. Misterio de la Abadía

El Misterio de la Abadía (Faidutti, 1996) es un juego de mesa similar a los anteriores, es decir, del tipo deducción, salvo que este no se desarrolla en el presente ni en una mansión como Cluedo sino en una abadía medieval. Los jugadores compiten entre ellos para resolver el misterio moviéndose a través de la abadía y realizando las preguntas adecuadas, mecánicas muy similares a las que queremos implementar en nuestro interrogatorio. Además este juego también tiene un sistema de turnos.

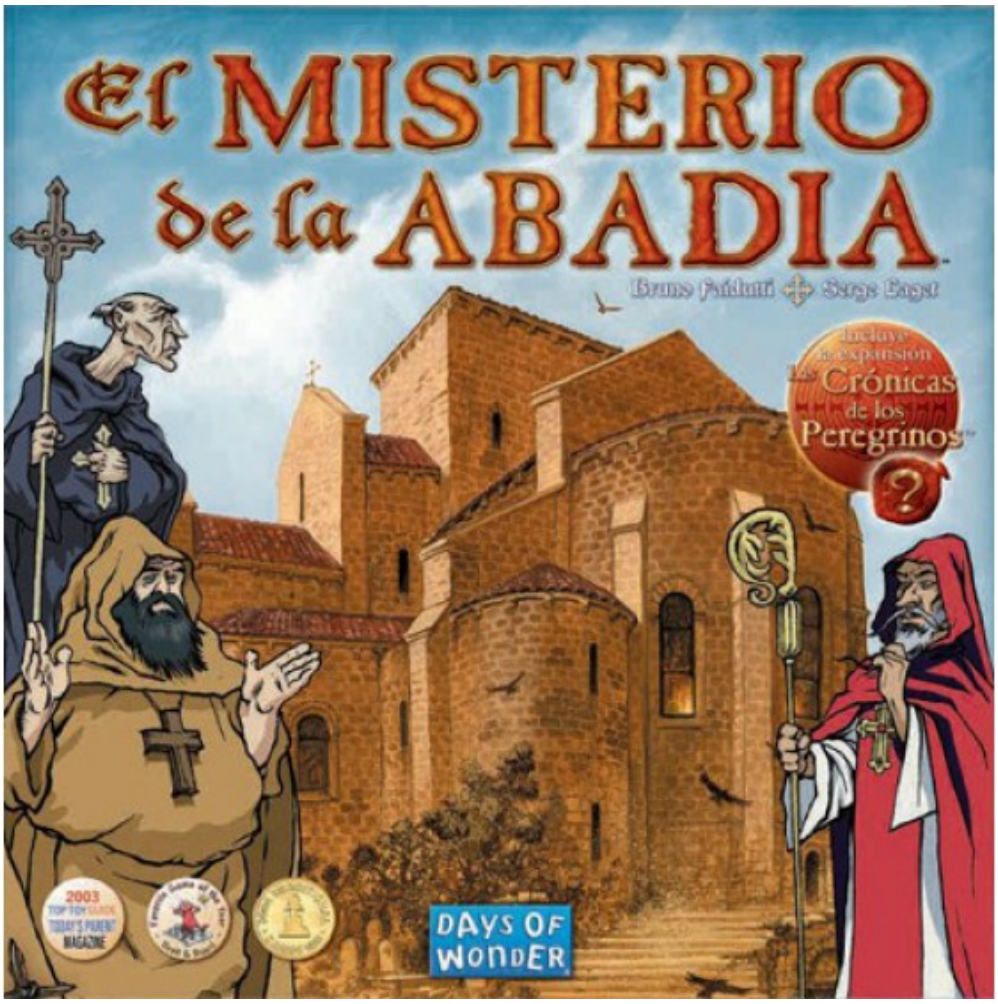

Figura 2.4: Portada juego de mesa .[El] Misterio de la Abadia

2.1.5. Alien Isolation

Alien Isolation (Assembly, 2014) es un videojuego multiplataforma del género survival horror y sigilo en primera persona desarrollado por The Creative Assembly y publicado por Sega el 7 de octubre de 2014. La protagonista Amanda deberá escapar de la nave en la que se encuentra atrapada junto con el Alien, pero él no es el único peligro pues también te encontrarás con androides que tienen instintos violentos y que no dudarán en atacarte nada más verte. A parte de los androides también hay otros humanos en la nave que intentan sobrevivir acabando con todo ser vivo que se encuentran incluido tú.

Aunque este juego no está desarrollado para probar inteligencias artificiales si que es interesante la que desarrollaron para el Alien, es más uno de los grandes alicientes con los que se vendía el juego era dicha inteligencia artificial. El Alien aprendía a lo largo de la partida, por ejemplo el jugador ha hecho un ruido muy fuerte y entonces se esconde en una taquilla para que el Alien no le encuentre, este truco te puede funcionar una o incluso dos veces pero a la tercera el Alien te encontrará entre las taquillas. Y ese es uno de los muchos ejemplos sobre está inteligencia artificial que aunque al final no era todo lo que nos querían vender si que llegó a ser una IA notable y bastante innovadora en el mundo de los videojuegos.

Figura 2.5: Portada de Alien Isolation

2.1.6. L.A Noire

L.A Noire (Bondi, 2011) es un videojuego desarrollado por Team Bondi y distribuido por Rockstar Games para PlayStation 3, Xbox 360 y PC.

En este juego tomamos el papel de un exmarine que tras la segunda guerra mundial comienza a trabajar como agente del departamento de policía de Los Ángeles.

El juego consta de varios asesinatos que debes resolver para culminar con el culpable de todos ellos al final del juego. En dichos casos debes interrogar a testigos y a familiares y amigos de la víctima. Lo interesante de estos interrogatorios es que tienes que fijarte en las expresiones faciales, muy conseguidas, del NPC para saber si te está mintiendo o no y el jugador puede llevar a cabo el interrogatorio de diferentes maneras. Puedes mentir al testigo para que le entre miedo y te diga la información que necesitas o incluso puedes inculparle para intimidarle y que suelte la información. Algunas de estas mecánicas se han contemplado para nuestro proyecto.

Figura 2.6: Portada de L.A Noire

2.2. Trabajo previo: Mouse Mind

Este proyecto parte de la idea de un TFG realizado anteriormente por el alumno Alejandro Villarín. La idea nos pareció muy interesante, y al ver que en dicho TFG no se llegó muy lejos con la idea principal hemos decidido retomarla nosotros. Además dicho trabajo anterior seguía la idea de un artículo escrito por nuestro Director y Codirector del proyecto (Álvarez y Peinado, 2015)

Dicho TFG pretendía desarrollar tanto un juego para probar una IA como la propia IA que sería probada en ese juego. Pretendía crear una inteligencia artificial que adquiriera conocimientos a lo largo del desarrollo del juego y que usara dicho conocimiento para poder responder de forma razonada a preguntas. Finalmente el desarrollo de este IA quedó muy lejos de lo que querían hacer en un principio, por eso nosotros cogemos esta idea y la desarrollaremos más. Pero nosotros nos enfocaremos en crear el juego para que desarrolladores de inteligencias artificiales puedan hacer las pruebas necesarias a su IA con nuestro juego, se podría decir que el juego hace de "Test de Turing" salvando las distancias.

Este trabajo partía de la idea de crear una herramienta para poner a prueba la Inteligencia Artificial. Las primeras ideas relacionadas con la inteligencia artificial surgieron junto a la lógica, los algoritmos y las matemáticas, los cuales tienen su origen en las culturas griegas o árabes varios años antes de Cristo, pero no es hasta 1950 cuando **Alan Turing**publica un articulo, donde se propone una respuesta para considerar a una máquina inteligente, momento en el que se define formalmente el término de inteligencia artificial. Surge así el llamado **Test de Turing**.

Según este test, lo que dota a un ser de inteligencia es su capacidad para imitar el comportamiento humano. Para superar el test, la máquina debe ser capaz de engañar al juez para que determine que no es una máquina sino un ser humano.

Ya son algunas las inteligencias artificiales que proclaman haber superado este test, como Egene Goostman. Una inteligencia artificial que simulaba ser un adolescente ucraniano de 13 años de edad [1].

Esta primera definición de inteligencia artificial ha ido evolucionando hacia un enfoque donde se pretende imitar el comportamiento racional.

Los primeros pasos dados en este campo fueron dados a través del ajedrez, donde se pretendía que la máquina fuera capaz de ganar a un ser humano. Poco a poco se ha ido evolucionando.

[1]Egene Goostman: `http://www.abc.es/ciencia/20140609/abci-superordenador-supera-primera-test-2014060911`

2.3. Unity

Unity, el entorno de desarrollo de videojuegos multiplataforma creado por Unity Technologies tiene como una de las características principales dar soporte a la compilación sobre diferentes tipos de plataforma, desde la Web al ordenador personal, a los dispositivos móviles y consolas, ya sean de sobremesa o portátiles.

Unity posee características similares a la de otros motores de videojuegos que son más potentes, por ejemplo Unreal. Ofrece sistemas de iluminación, materiales, físicas, etc. La ventaja de Unity es el hecho de que hay una gran cantidad de documentación y tutoriales que nos ayudarán en nuestra labor.

También ofrece una tienda interna rica en contenido, llamada Unity Asset Store a la que pueden acceder todos los desarrolladores ya sea desde el navegador o desde el mismo Unity. Dicho contenido incluye sonidos, modelos de personajes y objetos, texturas e incluso demos. Hay contenido gratuito y también de pago.

Algunos juegos conocidos que usan el motor de Unity, son *Hearthstone: Heroes of Warcraft* (Blizzard Entertainment, 2014), *Pokemon GO* (Niantic Inc, 2016), *Nihilumbra* (BeautiFun Games, 2012) o *Pillars Of Eternity* (Obsidian Entertainment, 2015), lo que refleja de todo lo que es capaz este entorno de desarrollo.

2.4. Retos

Teniendo en cuenta el estado actual y los objetivos que nos hemos marcado, nuestro mayor reto es aunar los elementos de los juegos de deducción y simulación en una herramienta para configurar juegos multijugador online a la vez que dichos elementos creen un entorno rico en posibilidades a la hora de tomar decisiones tanto en la fase de simulación como posteriormente en la fase del interrogatorio. Aunque la fase de simulación es mas fácil de implementar porque sera una serie de preguntas y respuestas entre el investigador y los sospechosos requiere que este bien pensado, nuestro reto es crear una interfaz para crear preguntas relacionadas con el estado final del laberinto, paredes rotas, quien se comió la comida, o quien estuvo en "x" casilla, cuyas respuestas sean si o no. Pero ademas queremos que de pie al engaño y la manipulación, por ejemplo pudiendo preguntar a un sospechoso quien rompió una pared que no acabó rota en la fase de simulación para así comprobar si nos esta mintiendo.

Capítulo 3

Metodología y herramientas

Para la realización del proyecto hemos seguido una metodología ágil, adaptada a las necesidades de nuestro proyecto y las capacidades del equipo. El modelo de proceso que estamos usando se basa en Scrum, realizando los sprints con una duración de dos semanas y teniendo al director de proyecto presente en estas reuniones. En dichas reuniones mostrábamos el trabajo realizado en esas dos semanas, comentábamos problemas que hubiéramos podido tener a la hora de realizar el trabajo asignado y se asignaban nuevas tareas para las próximas dos semanas.

3.1. Metodología

Al comienzo del proyecto las tareas o hitos asignados en estas reuniones presenciales eran principalmente para aprender a usar Unity. Para ello realizábamos "juegos" mediante tutoriales que ofrecía la propia página de Unity. Además de esto íbamos comentando aspectos relacionados con el juego que teníamos pensado desarrollar, como por ejemplo el número mínimo de jugadores, las acciones que podrían realizar los jugadores en el juego y otras características y restricciones. Cada uno de los componentes del equipo nos dedicábamos especialmente a una parta en concreto. Las reuniones estaban organizadas de manera que apareciesen nuevas tareas para cada uno de los participantes y así ir avanzando.

Estas reuniones se llevaban acabo cada dos semanas. En el primera mitad del curso se realizaban los martes y en la segunda mitad los viernes, con alguna excepción por problemas de agenda. El propósito era que para cada reunión estuviesen los objetivos detallados en la anterior reunión. Entre una reunión y otro los participantes se comunicaban entre sí ya fuera a través de Whatsapp o Slack.

Las tareas las uníamos en un repositorio de Github según íbamos teniendo alguna nueva funcionalidad del juego, por minuciosa que fuera, y así enseñársela al director del proyecto en la próxima reunión.

Más allá de estas reuniones también teníamos una serie de presentaciones a lo largo del curso en las que enseñábamos al resto de personas pertenecientes al grupo de investigación que estábamos haciendo y cuál era nuestro progreso. Una presentación antes de navidades, otra antes de semana santa y una última antes de una entrega final.

3.2. Plan de trabajo

Ya que seguíamos una metodología ágil no teníamos definido un plan de trabajo fijo sino que íbamos añadiendo funcionalidad a razón de nuestra capacidad y dificultad de lo propuesto, pero si que nos marcamos dos hitos a cumplir desde el principio.

Para el primer cuatrimestre se acordó tener especificados los requisitos del juego para tener claro como sería este y poder "venderlo" en la presentación de antes de navidad de la que ya hablamos anteriormente. Además de esto deberíamos haber tenido nuestra primera toma de contacto con Unity y haber aprendido lo suficiente como para ponernos manos a la obra con el juego.

Para el segundo cuatrimestre tendríamos que tener una versión funcional del juego, o al menos con la primera parte del mismo. También debería funcionar el multijugador. Todo lo acordado se enseñaría en la presentación de antes de Semana Santa.

3.3. Herramientas

Las herramientas que utilizamos para la comunicación entre los componentes y coordinadores del proyecto, el alojamiento de las diferentes secciones y además de la realización de la memoria fueron las siguientes:

3.3.1. Comunicación

Las herramientas principales de comunicación han sido Slack, Trello y Skype.

Slack

Esta herramienta la usamos como recomendación del director del proyecto, ya que la había usado anteriormente y le dio buenos resultados. Slack es una herramienta de comunicación en equipo.

Slack te proporciona diferentes canales en los que podíamos hablar con el resto de miembros del grupo de investigación con el que colaboramos. Dichos canales pueden ser públicos o privados. Para cada proyecto se tenía un canal privado y también contábamos con varios canales públicos en los que podíamos interactuar con otras personas que fueran de nuestro propio proyecto. Además de estos canales, Slack también ofrece "chats" o mensajes directos con cada uno de los individuos y con un bot de Slack.

Todo lo anterior está relacionado con las facilidades que ofrece Slack en cuanto a la comunicación entre miembros de un equipo, pero también ofrece conexión con otras aplicaciones o servicios. Como por ejemplo Google Drive, Google Calendar, GitHub o Trello entre otras aplicaciones.

Trello

Trello es un software de administración de administración de proyectos. Dicho software lo usamos para asignarnos las tareas a realizar e ir viendo el progreso que llevaban las tareas ya asignadas para poder hacer un seguimiento de como iba avanzando el proyecto.

Al igual que Slack, Trello te permite sincronizarte con otras aplicaciones o programas como algunos de los citados anteriormente.

Contemplamos la idea de usar Asana como gestor de tareas en lugar de Trello, pero finalmente nos decidimos por esta última debido a que Asana es mucho menos intuitivo que Trello.

3.3.2. Alojamiento compartido y control de versiones

Para el alojamiento web del proyecto necesitábamos una buena plataforma online de control de versiones, y un sistema general para compartir información.

GitHub

Usamos GitHub debido a que en algún momento de la carrera aprendes los aspectos básicos de esta herramienta y por tanto no tendríamos que asignar tiempo del proyecto en aprender una nueva tecnología. GitHub nos facilita trabajar en paralelo en el mismo proyecto sin poner en riesgo este. Además como hemos dicho antes, muchas de las herramientas que hemos decidido usar para desarrollar el proyecto nos permiten sincronizarnos con GitHub, algo que no nos permiten otras herramientas como por ejemplo TortoiseSVN.

Google Drive

Como herramienta de alojamiento y compartición de ficheros utilizamos Google Drive. Hemos elegido esta herramienta en lugar que otras similares como DropBox o OneDrive por la posibilidad de realizar edición de documentos online por varios usuarios a la vez, como por ejemplo los power points de las presentaciones. Lo cual nos permite trabajar en paralelo en la documentación facilitando un desarrollo más ágil del proyecto. Además con la cuenta de la Universidad Complutense de Madrid disponemos de almacenamiento ilimitado, algo que no tendríamos en las herramientas de la competencia citadas con anterioridad.

3.3.3. Documentación

Para borradores y notas de las reuniones usamos Google Docs. Pero para elaborar una buena documentación para el proyecto, era necesario contar con una herramienta específica para la redacción y maquetación final de la memoria del trabajo.

Google Docs

Utilizamos los Google Docs para crear documentos a sucio a los que todos tuviésemos acceso y fuese de fácil acceso.

LaTeX y Overleaf

Utilizamos esta herramienta de escritura y edición de documentos como aplicación principal para desarrollar esta memoria. Overleaf es una herramienta *on-line* del conocido sistema de composición de textos LaTeX y que además permite la implantación de plantillas como por ejemplo la que está siendo usada, que es la que propone la Facultad de Informática de la Universidad Complutense de Madrid.

Balsamiq

Hemos usado esta herramienta para crear prototipos interactivos para poder mostrar en presentaciones y las sesiones que teníamos con el profesor cada dos semanas. Además de esto también esta herramienta nos ha ayudado a materializar las ideas que teníamos en la cabeza a cerca de como se vería o sería el juego, principalmente la interfaz y el desarrollo del mismo, cuando este aún se encontraba en sus primeras iteraciones.

Capítulo 4

Especificación y requisitos

Lo que se busca con esta herramienta es evitar tener personajes predeterminados que no razonan sobre su comportamiento y que utilicen el conocimiento obtenido para que puedan responder de forma razonada a preguntas. Los desarrolladores de IAs podrán conectar sus "creaciones" a este juego para probar que dichas IAs cumplen con lo anterior.

Como es un problema de gran tamaño y de gran complejidad hemos definido un entorno lo más reducido posible que mantenga las ideas principales que caracterizan al reto que nos hemos propuesto hacer frente.

Dado que debemos interactuar con personajes que se relacionan unos con otros compartiendo información, de la cual no se tiene la certeza de si es verdad o no, vamos a definir el juego como un videojuego multijugador que consta de dos partes bien diferenciadas de las que se hablará más adelante.

4.1. El laberinto y los sospechosos

Nos encontramos en un laberinto en el que se encuentran encerrados los sospechosos. Por el día los sospechosos deben permanecer inmóviles en las posiciones que tienen designadas, pero por la noche aprovechan para hacer lo que les plazca encontrarse con otros sospechosos, romper las paredes mas frágiles del laberinto, e incluso comerse la comida que se encuentra en el laberinto, acto que esta terminantemente prohibido.

Este alboroto puede realizarse, pero no sin tomar ciertas precauciones ya que el investigador vendrá a la mañana siguiente a valorar la situación de los sospechosos. Este confía en que los sospechosos se encuentren en sus posiciones originales, que las paredes permanezcan intactas y que nadie se haya comido la comida. De no ser así, el investigador llevara a cabo un interrogatorio con cada uno de los sospechosos con el fin de descubrir quien fue el artífice del desastre que se produjo la noche anterior.

Ademas del interrogatorio el científico buscara en el laberinto pruebas que le facilite saber que ratones mienten y cuales no. Dichas pruebas pueden ser marcas que tengan los sospechosos por haberse puesto en contacto unos con otros, deducir que camino es mas corto para llegar a cada objetivo, paredes rotas.

El juego por tanto consta de dos fases fuertemente diferenciadas, una que tiene lugar durante la noche, en la que los sospechosos realizan sus travesuras mientras el investigador duerme y otra, que ocurre por la mañana tras esta primera fase, en la que transcurre el interrogatorio.

4.2. Movimiento y acciones de los sospechosos

En primer lugar se quiere evitar que nuestros personajes actúen de forma predeterminada y en su lugar razonen que acciones son las mejores en función de los hechos que ocurren durante la ejecución. Es por ello que tenemos que desarrollar una inteligencia capaz de controlar el movimiento y actuaciones de los sospechosos por el tablero.

Las acciones serán guiadas por una serie de deseos, que varían de unos sospechosos a otros. Los sospechosos también pueden tener sentimientos hacia otros sospechosos, que no necesariamente tienen que ser correspondidos. Estos deseos y sentimientos serán traducidos en intenciones, dando lugar a las siguientes posibilidades:

- Buscar y comerse el queso.

- Buscar a otro ratón para jugar con él (puede que el otro ratón no quiera jugar con nosotros).
- Caminar por el tablero interactuando con los diferentes elementos de este.
- Permanecer quieto.

Según estos deseos tenemos cuatro sospechosos bien diferenciados. Estos se diferencian por colores y deben cumplir sus propios objetivos si quieren progresar en el juego. Uno de ellos, el "buenazo", será penalizado por romper paredes frágiles. En cambio otro sospechoso es recompensado por realizar travesuras como romper estas paredes. Hay otro al que podemos llamar el "inquieto" obtiene puntos tocando a otros sospechosos del laberinto. Y por último tenemos el "glotón" el cual su principal objetivo es comerse la comida, este ratón obtiene una mayor recompensa que los demás sospechosos por comerse la comida. Con estas motivaciones ponemos objetivos a los jugadores para que se vean motivados a la hora de moverse por el laberinto, ya que pensamos que si no hay una recompensa ¿Por qué razón iban los jugadores a intentar comerse la comida, romper paredes o interactuar con otros jugadores?

Con este fin las IAs a examinar reciben como entrada una serie de deseos a cumplir. Cada uno de estos deseos tiene un peso específico, que indica cómo de valioso es ese deseo para el ratón.

Además de sus deseos el sospechoso conocerá su color, la posición que ocupa en el tablero y tendrá información de todo lo que ha visto a lo largo de la ejecución. La información de lo que ha visto anteriormente es muy útil, porque nos ayudará a encontrar nuestros posibles objetivos.

4.3. Requisitos de información

Requisitos con la configuración del proyecto vía XML:

- Disposición en el laberinto de los objetos vía XML.

- Existe la posibilidad de que se añadan nuevos objetos al juego vía XML, siempre y cuando existan los recursos gráficos en el interior del proyecto.
- Número de jugadores, incluye el número mínimo y máximo de jugadores.
- Número máximo de turnos que dura una partida.

4.4. Requisitos funcionales

Requisitos relacionados con la primera fase del juego:

- Mostrar por pantalla un tablero, que hará las veces de laberinto.
- Mostrar sobre dicho tablero a los cuatro ratones.
- Mostrar sobre algunas casillas muros que serán las paredes del laberinto.
- Mostrar sobre algunas casillas paredes fragiles (paredes que pueden romperse).
- Mostrar en alguna casilla comida.
- Definir un formato XML (maze.xml) que sirva para configurar los diferentes aspectos del juego. Ya sea el tamaño del tablero o laberinto, los elementos del laberinto, número de jugadores.
- Cargar dicho fichero maze.xml (siempre el mismo, por defecto) y colocar los elementos encima de su casilla correspondiente.
- Crear un sistema de turnos.
- Llevar el sistema de turnos a multiplayer, de manera que cada jugador pueda ver en pantalla de quien es el turno.
- Que los sospechosos se puedan mover una casilla concreta en su turno, sólo pueden moverse a una casilla adyacente a su posición. Dejar la "puerta abierta" a que sea algo controlado por humano o por IA.

- Que los movimientos de los sospechosos se lleve al multiplayer.
- Crear un sistema de visibilidad (al que nos referiremos en algunas ocasiones como niebla) en el que puedes ver "todo" en línea recta hasta llegar a un muro (o el límite del tablero).
- Llevarnos ese sistema de visibilidad a cada aplicación cliente.
- Que se puedan realizar acciones como romper una pared frágil y pasar a través de esta.
- Que los sospechosos puedan comerse la comida.
- Que esta fase del juego finalice tras el transcurso de un número X de turnos.

Requisitos relacionados con la segunda fase del juego, es decir, el interrogatorio:

- Que se pueda conectar un investigador a la fase del interrogatorio.
- Que el investigador pueda usar comboboxes para componer una pregunta.
- Que el sospechoso pueda usar comboboxes para componer una respuesta.
- Que finalmente el investigador pueda realizar acusaciones, asociar distintos "crímenes" a los sospechosos.
- Que se realice el cálculo de la puntuación final para los sospechosos.
- Que se muestre una ventana final con la puntuación una vez finalice el interrogatorio.
- Que los sospechosos solo puedan responder cuando sea su turno.

4.5. Requisitos no funcionales

- Usar Unity para el desarrollo del software.
- Obtener los assets necesarios (modelos de los personajes, paredes frágiles, comida, etc) para el aspecto visual.
- Usar las posibilidades que nos ofrece Unity con su clase Networking para llevar a cabo el multijugador.
- Que el juego sea multiplataforma y permita jugar vía online desde distintas plataformas. Por ejemplo un usuario juega en su PS4, otro en su portátil y un último desde su smartphone.
- La interfaz debe adaptarse a todos los dispositivos en los que el software esté disponible.

Capítulo 5

Análisis, diseño e implementación

En este capítulo abordaremos el análisis y diseño, tanto de la parte relacionada con la interacción del cliente y la aplicación, como de la parte correspondiente con la interacción del usuario y el juego desarrollado.

En primer lugar, en el apartado 5.1 haremos un análisis de cómo se tiene previsto usar el sistema. A continuación se mostrarán los diagramas de caso de uso que reflejarán lo contado previamente.

El siguiente apartado, 5.2, contendrá todo aquello que tiene que ver con el diseño de la herramienta, mostrando las relaciones que tienen las entidades y explicando la arquitectura en general. De esta manera abarcaremos todo lo relacionado con la comunicación entre las partes señaladas al principio del capítulo.

Por último, en el apartado 5.3 indicaremos la estructura y las tecnologías usadas en cada capa, así como detalles de la implementación de la aplicación.

5.1. Análisis

Como se pensó desde un principio, una función principal de nuestra herramienta debe sera la creación de un escenario, completamente configurable. Para ello tanto el diseño del mapa de la fase de exploración como varias características de la configuración(numero de turnos, tipos de visión, puntos que obtienen los jugadores...) se pueden especificar en varios XML.

En esta herramienta se hace una división de usuarios entre los denominados Jugadores y otro que se denominará Investigador. La diferencia entre ellos es que el Investigador creara el servidor y los XML de configuración, mientras que el usuario solo podrá participar en el juego ya sea como detective o sospechoso.

Todos los jugadores podrán conectarse desde su ordenador a un servidor ya creado a partir de la dirección Ip, una vez dentro del lobby podrán escoger el nombre de su ratón y el color, dependiendo del color del ratón se les aplicara unas reglas de puntuación u otras. El detective es un tipo de personaje especial que participa solo en la segunda parte del juego pero también entra a la partida en este punto.

Por otro lado el investigador creara dos archivos XML uno para definir que aspecto tendrá el laberinto en el que se moverán los sospechosos en la primera fase y el otro para configurar los detalles del experimento, numero de turnos de las fases, numero de sospechosos, etc. Una vez creados ambos archivos abrirá la aplicación para crear un servidor dedicado al que se unirán los jugadores desde sus ordenadores, también tiene la opción de crear el servidor y participar en la partida.

5.1.1. Diagramas de caso de uso

En este apartado se mostrarán los diagramas de casos de uso que corresponderán a lo contado en el apartado anterior. Como los nombres y relaciones de los casos de uso son muy claros acerca de lo que hacen o necesitan, hemos optado por hacer diagramas generales en vez de detallar enteros y uno por uno los distintos casos.

Diagrama de caso de uso de creación del servidor

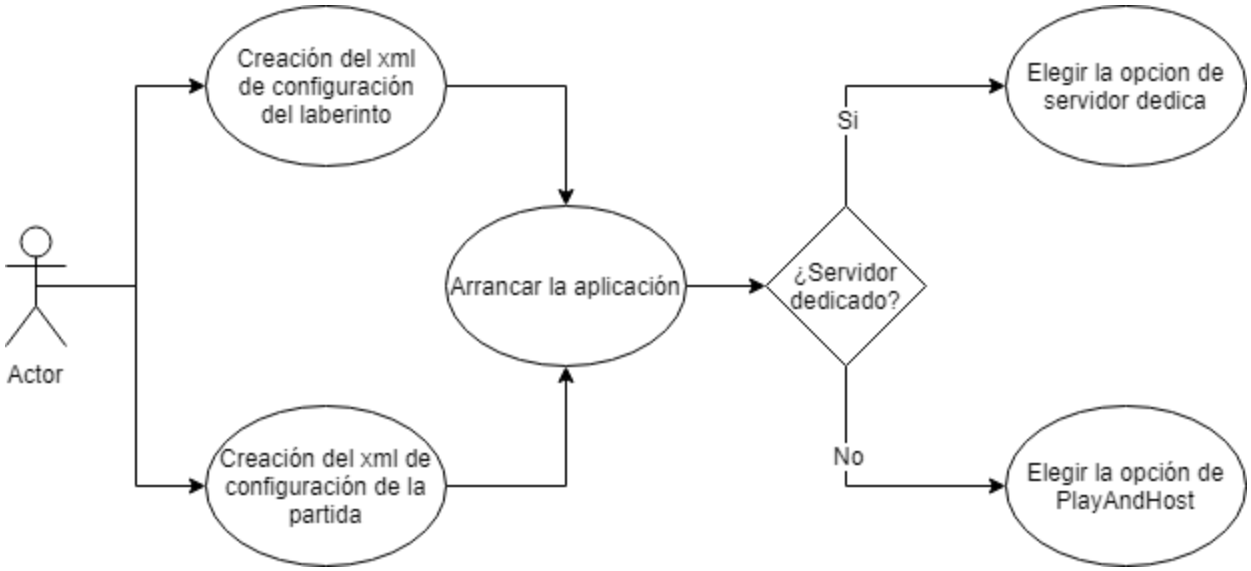

Figura 5.1: Diagrama de caso de uso de creación del servidor.

Diagrama de caso de uso de unirse a una partida

Figura 5.2: Diagrama de caso de uso de unirse a una partida.

Diagrama de caso de uso de desarrollo de una partida

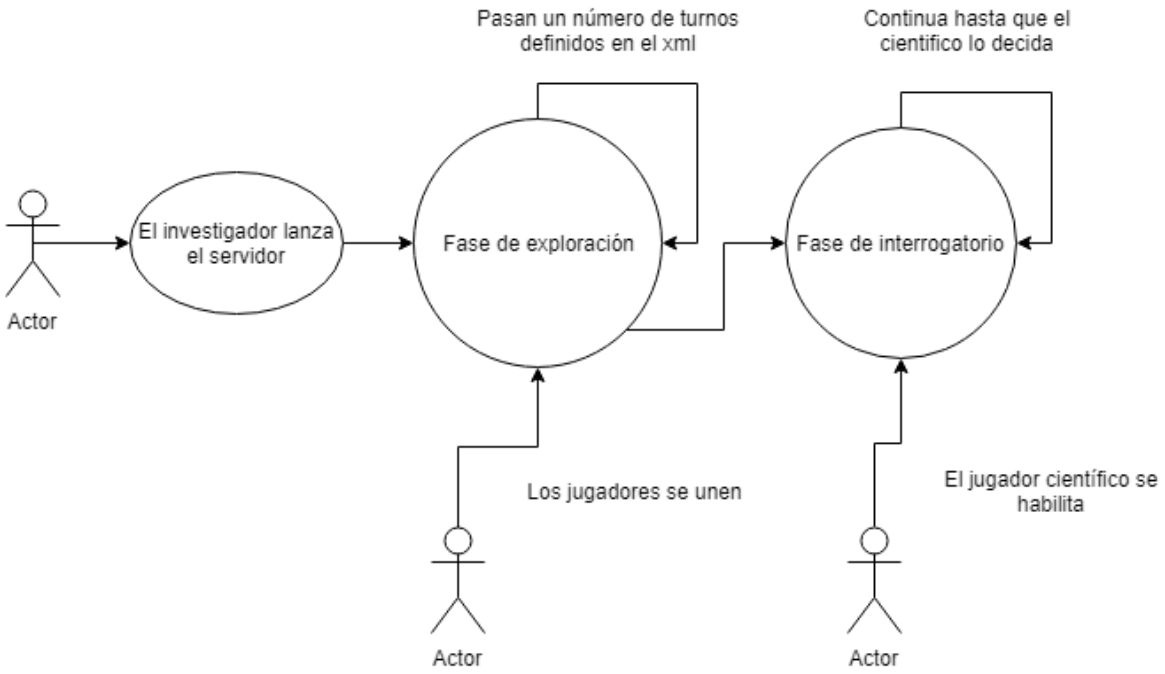

Figura 5.3: Diagrama de caso de uso de desarrollo de una partida.

5.2. Diseño y arquitectura

En este sección vamos a explicar el diseño y la arquitectura que sigue la aplicación junto a sus diferentes capas.

5.2.1. Diseño

Comenzamos diseñando un proyecto en Unity muy sencillo en el que se creaba un laberinto que solo constaba de paredes y sospechosos como se puede observar en las figuras 5.4 y 5.5, mas tarde añadimos un sistema de turnos que funcionaba en local, posteriormente se transformo la herramienta para que el juego funcionase en red en lugar de en local. Añadimos la fase del interrogatorio al terminar la fase de simulación. A partir de aquí fuimos añadiendo elementos al laberinto para dar mas posibilidades a los jugadores figura 5.6.

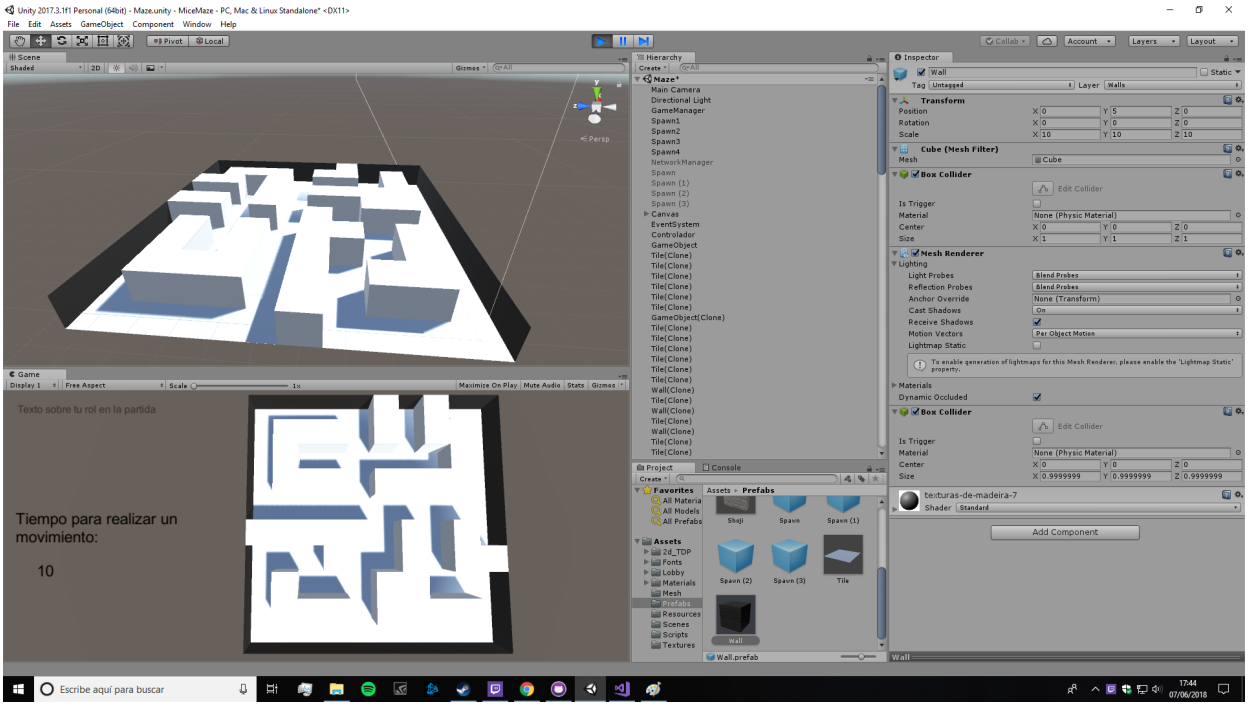

Figura 5.4: Laberinto vacío.

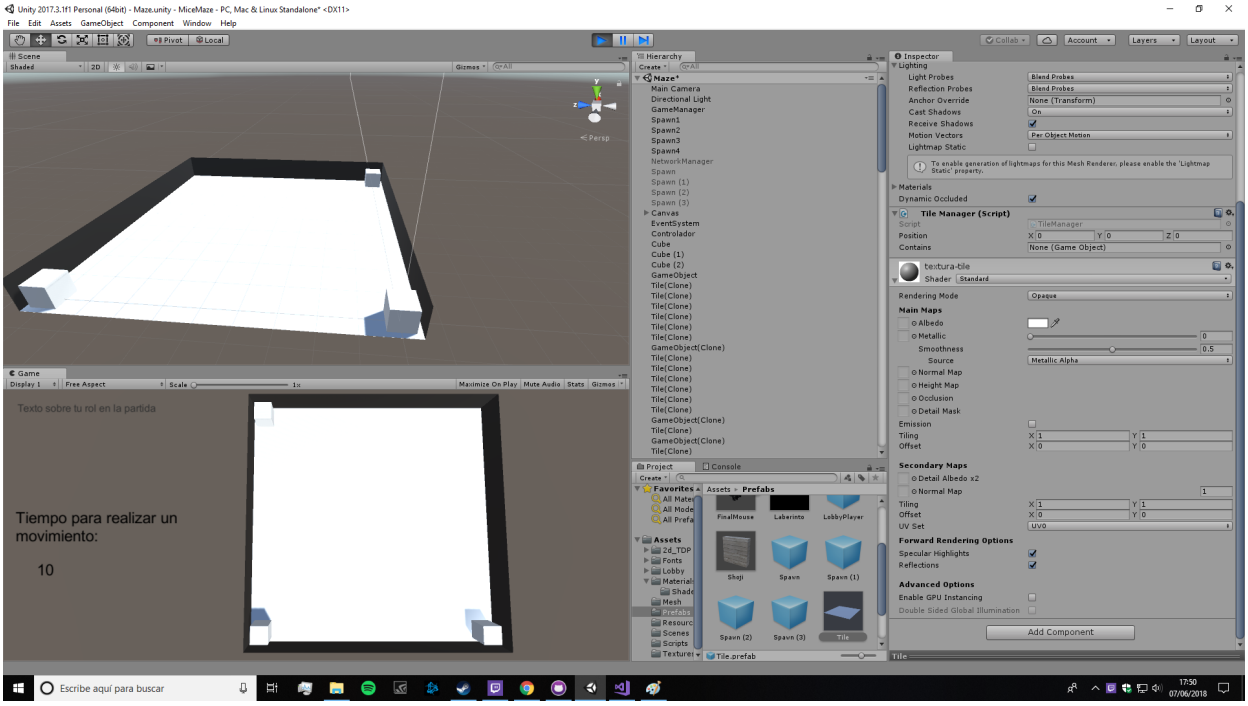

Figura 5.5: Laberinto vacío.

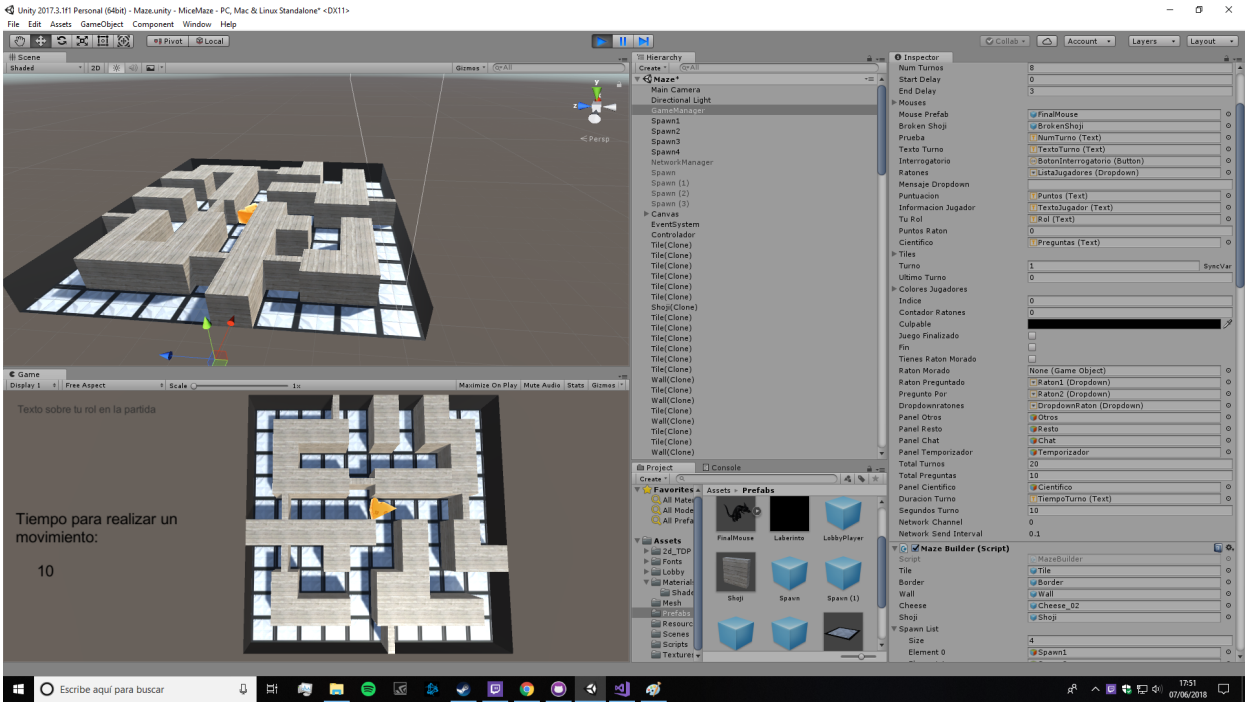

Figura 5.6: Laberinto vacío.

5.2.2. Arquitectura

Nuestra aplicación cuenta con una arquitectura en la que el Investigador lanza un servidor al que se unen de 4 a 5 jugadores, uno de ellos como detective. Cada jugador inicia su ejecutable y se une al servidor que creó el investigador, esto hace que en los ordenadores de cada jugador haya una copia del juego, la cual se va actualizando mediante los Commands y ClientRpc que aporta Unity para controlar el intercambio de información entre servidor y clientes.

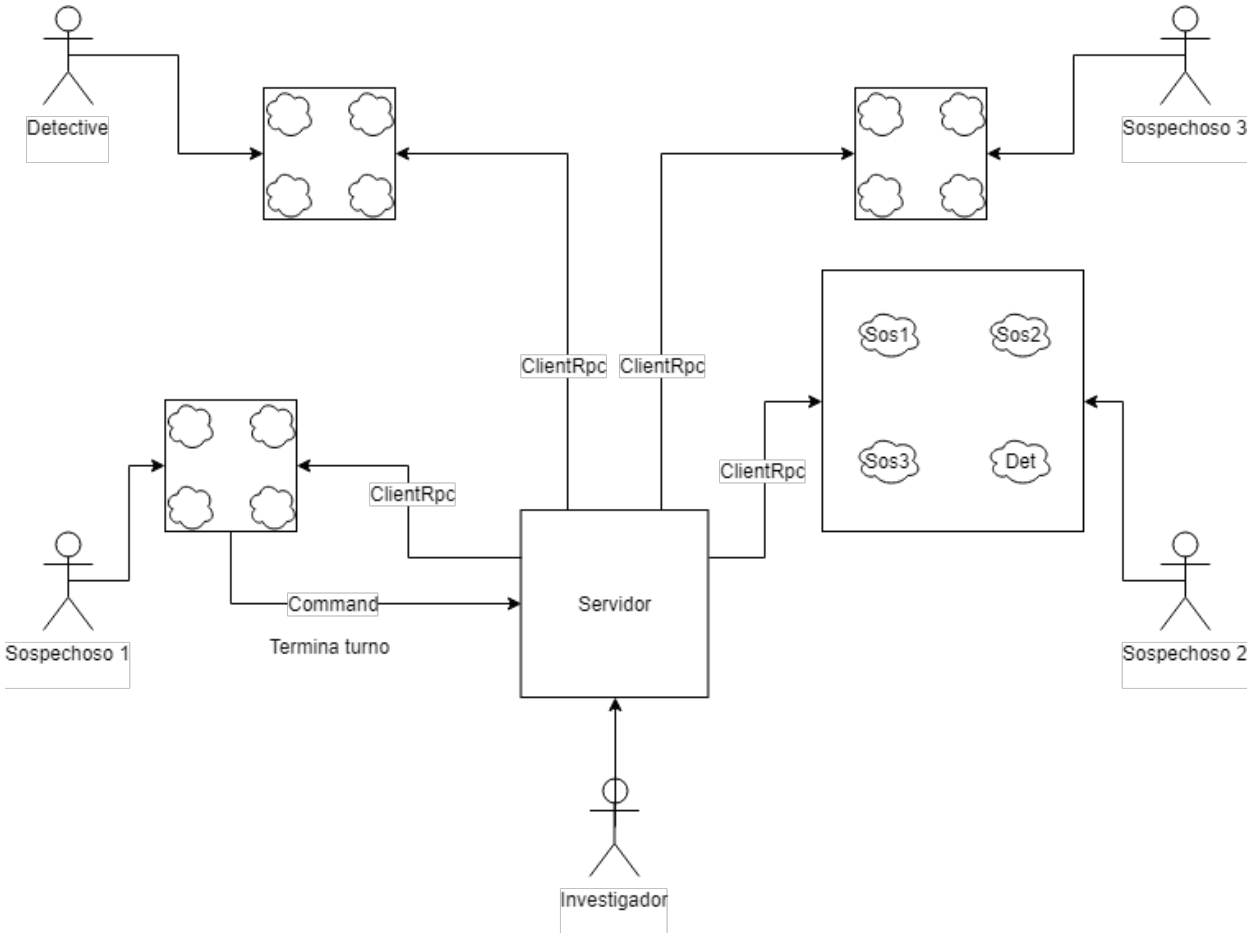

Figura 5.7: Arquitectura del proyecto con servidor dedicado.

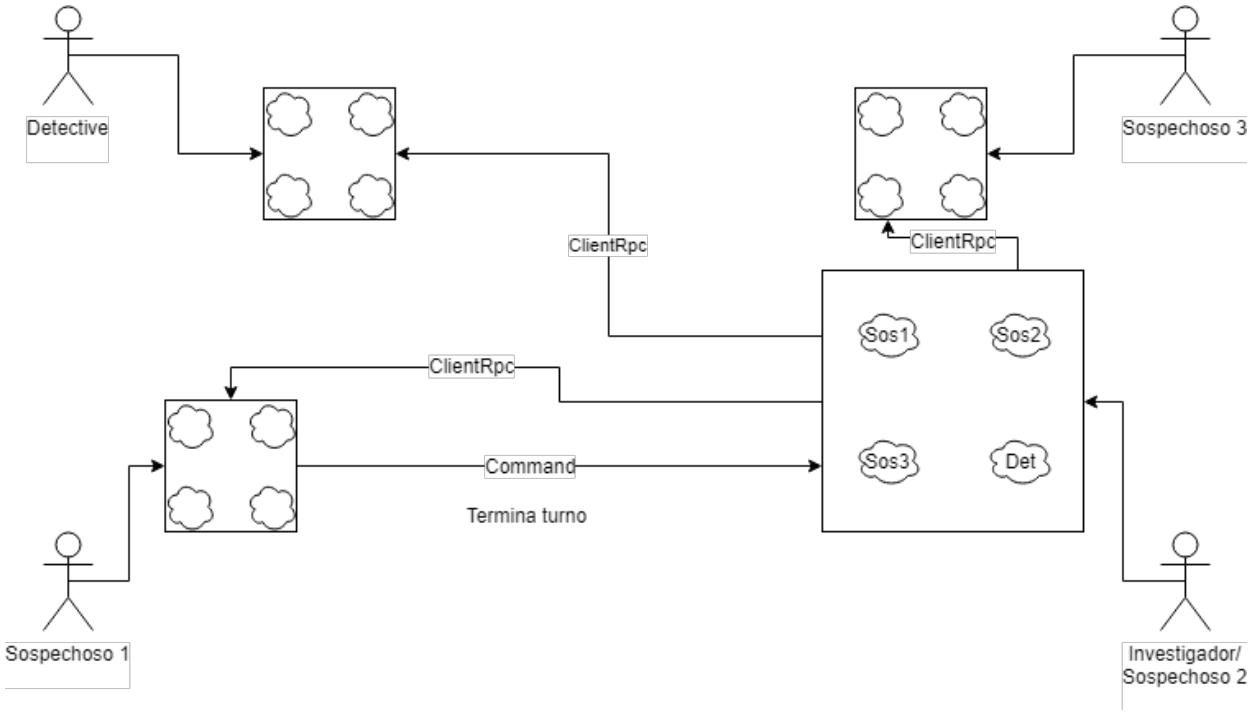

Figura 5.8: Arquitectura del proyecto sin servidor dedicado.

Servidor

El *servidor* corresponde con el dispositivo, ya sea un ordenador o un móvil, del Investigador en el que se lance la aplicación, es el que lleva más carga lógica ya que se encarga de mantener el estado de todos los clientes actualizados. Como ya se dijo anteriormente el servidor puede ser un servidor dedicado, como se aprecia en la Figura 5.7 "Arquitectura del proyecto con servidor dedicado", o estar alojado en uno de los jugadores, es decir, el Investigador participaría en el juego, como se puede ver en la Figura 5.8 "Arquitectura del proyecto sin servidor dedicado".

Desde el servidor se cargan los datos de configuración almacenados en archivos XML y se replican a los clientes. Cuando al servidor le llega un cambio de estado desde un cliente, el servidor valida ese cambio y lo transmite a los clientes a través de un ClientRpc. Ya que cada partida se puede configurar antes con los archivos XML nos pareció más apropiado que sea el ordenador del Investigador el que hiciese de servidor en lugar de utilizar un proveedor de DNS para montar el servidor en red.

Clientes

Por otro lado, los *clientes* son los dispositivos, puede ser un ordenador o un dispositivo móvil, solo necesita conexión a la red de los usuarios en los que se lanzo el juego.

Intercambio de Command y ClientRpc

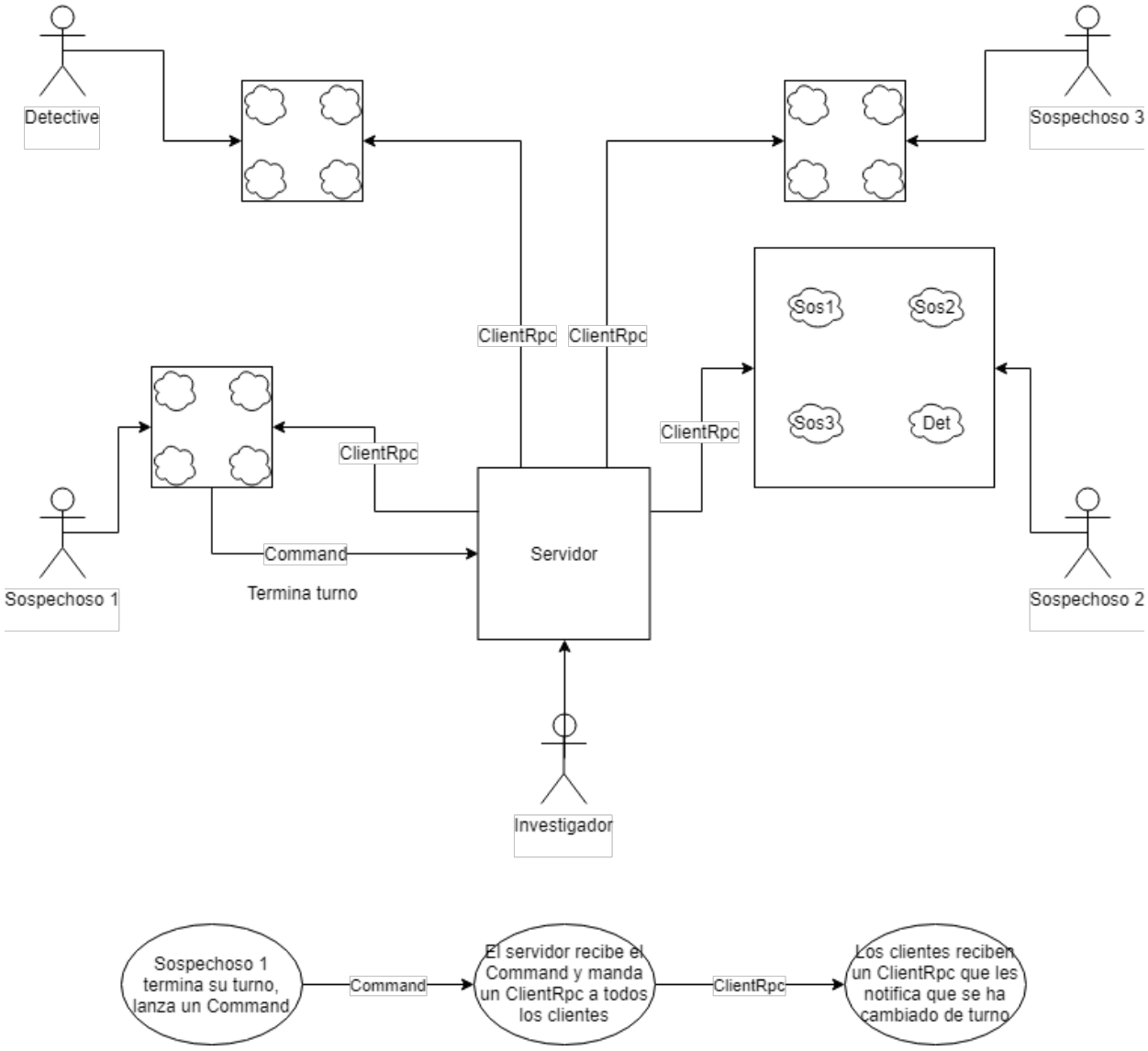

Figura 5.9: Intercambio de Command y ClientRpc.

Cada cliente tiene una copia del estado del juego actualizada por el servidor, cada vez que un jugador lleva a cabo una acción que modifique el estado del juego, por ejemplo moverse, responder una pregunta... el cliente manda un Command al servidor para avisarle que se ha modificado el estado de la partida, este intercambio de mensajes se ejemplifica en la figura 5.9 “Intercambio de Command y ClientRpc”.

NetworkManager Unity

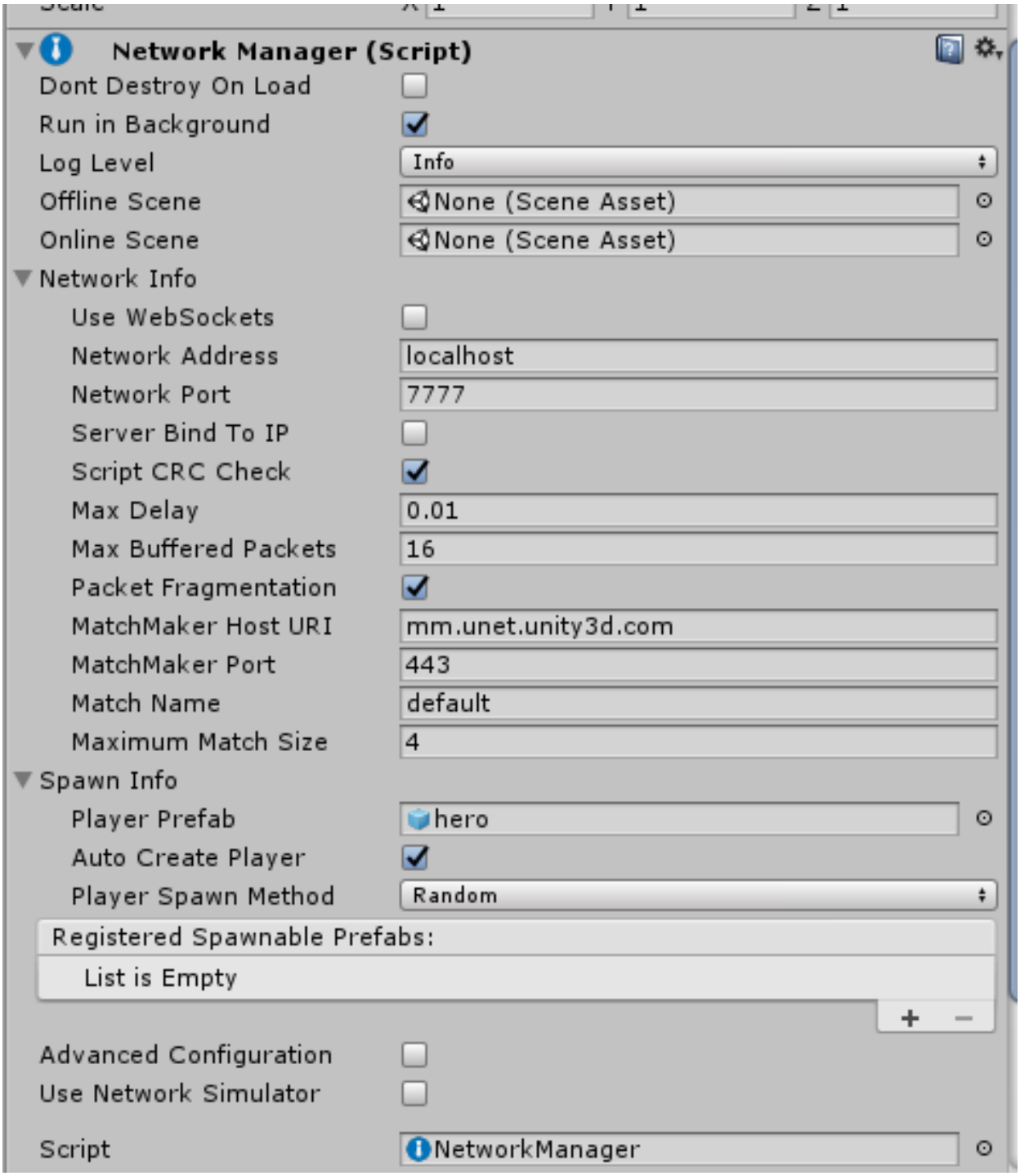

Figura 5.10: Componente NetworkManager

Para que los clientes puedan unirse a un servidor hemos utilizado el componente de Unity NetworkManager, se puede ver el aspecto que tiene en el inspector de Unity en la figura 5.10 “Componente NetworkManager”, que se encarga de definir que en dirección IP estará el servidor(localhost generalmente), el puerto en el que se alojara el servidor, cuantos jugadores podrán unirse, los prefabs de los jugadores que se unan a la partida, si la creación de los gameObject de los jugadores es automática cuando entran al servidor, el método de spawn (Random, RoundRobin)

5.3. Implementación

En esta parte de implementación se detallarán las diferentes tecnologías, lenguajes de programación y entornos de desarrollo usados.

Según vaya avanzando y mejorando el proyecto, los desarrolladores tendrán a su disposición un repositorio en GitHub, sistema de control de versiones que ya ha sido nombrado anteriormente, y donde futuros desarrolladores podrán descargar la herramientas e implementar sus propias mejoras. https://github.com/Narratech/MiceMaze

5.3.1. Lenguajes de programación

C#

Lenguaje de programación orientado a objetos, derivado del lenguaje C/C++. Se ha usado este lenguaje para el desarrollo de los *scripts* debido a la amplia comunidad existente en Unity y que Unity basa toda su documentación en C#.

XML

Es un meta-lenguaje que permite la creación de un lenguaje de marcado para almacenar información. Debido a la estructura proporcionada por las marcas los documentos XML son fácilmente legibles por bases de datos y permiten el intercambio de información entre aplicaciones.

5.3.2. Tecnologías

Unity

Herramienta que ya ha sido comentada en profundidad en el Capítulo 2.

Se ha utilizado tanto para el desarrollo de la herramienta como para los juegos de prueba.

Unity Multiplayer

Es una parte de Unity que permite crear juegos multijugador online de manera sencilla y rápida, esta orientado a juegos en tiempo real.

Para usar un sistema de turnos como el que buscábamos utilizamos dos tipos de funciones aportadas por esta tecnología, los Command y los ClientRpc que se encargan de organizar la información que se transmite por la red.

Microsoft Visual Studio

Entorno de desarrollo Integrado desarrollado por Microsoft. Soporta varios lenguajes entre los cuales C++, .NET, Java y C# entre otros.

Permite añadir librerías muy fácilmente con el gestor de paquetes NuGet.

Posee una versión de pago y otra gratuita llamada Community. Esta herramienta es la que usamos para la programación en C# de los diferentes scripts necesarios para el sistema y los juegos de demostración.

5.3.3. Desarrollo del juego

En esta sección contaremos el desarrollo del juego que hemos creado para poner a prueba inteligencias artificiales.

Este juego ha tenido dos versiones a lo largo de su desarrollo en la primera se jugaba a nivel local y en la segunda se podía jugar online.

La primera versión nos sirvió como una primera toma de contacto con Unity y C#, en la segunda versión con mas experiencia en la materia y una investigación en el multijugador de Unity la aplicación funcionaba en red.

A continuación explicaremos el diseño que han seguido las siguientes versiones:

- **Versión 1 - Multijugador local**

  La herramienta permite crear juegos de tablero con casillas, el tablero esta formado por las paredes y las casillas vacías por las que los sospechosos pueden desplazarse libremente siguiendo el sistema de turnos. Los jugadores solo pueden desplazarse una casilla de distancia.

  Esta versión acababa cuando cada jugador se había movido un numero determinado de veces.

  El procedimiento sería:

  1. El investigador crea los archivos de configuración
  2. Inicializamos el juego con el numero de sospechosos especificado en el archivo de configuración
  3. Empieza la primera ronda
  4. Se mueven todos los sospechosos, por turnos y de uno en uno
  5. Finaliza la ronda
  6. El juego continua hasta que se acaban las rondas

  Esta versión servia para comprender como mover los sospechosos y organizar los turnos en la versión posterior, la forma en la que estaba programado el sistema de turnos se descartó al introducir el factor online.

- **Versión 2 - Multijugador Online**

En esta versión reconstruimos la anterior para que se pudiese jugar desde distintas maquinas a través de Internet, para ello utilizamos los Command y ClientRpc, son un tipo de funciones soportadas por Unity que facilita para la comunicación entre cliente y servidor. Con las funciones Command desde un cliente mandamos una función para que se ejecute solo en el servidor, y con las funciones ClientRpc se lanzan desde el servidor y se ejecutan en todos los clientes A parte del online también se incluyo la fase del interrogatorio y se enriqueció los elementos que componen la parte de simulación.

Esta versión acababa cuando termina la fase de interrogatorio. El procedimiento sería:

1. El investigador crea los archivos de configuración
2. El investigador arranca un servidor ya sea dedicado o no
3. Se unen los jugadores a partir de la dirección IP del servidor
4. Empieza la primera ronda
5. Se mueven todos los sospechosos, por turnos y de uno en uno
6. Finaliza la ronda
7. La fase continua hasta que se acaban las rondas
8. Una vez terminada la fase de simulación comienza la fase de interrogatorio
9. El detective va haciendo preguntas a los sospechosos
10. Una vez acabados los turnos de preguntas el detective declara los culpables
11. Se calculan las puntuaciones de los jugadores y se muestran

En el siguiente capitulo hablaremos de las pruebas que realizamos para comprobar las distintas funcionalidades de la aplicación.

Capítulo 6

Pruebas y resultados

Para probar nuestra plataforma DeductionLab, la utilizamos para implementar un pequeño juego llamado MiceMaze.

6.1. MiceMaze: Especificación

A lo largo del capítulo 4 especificamos diferentes requisitos relacionados con este juego de ejemplo por lo que en este apartado mostraremos las especificaciones que tenían las versiones desarrolladas para las pruebas con usuarios.

La versión que teníamos disponible para la primera de las dos pruebas con usuarios que realizamos presentaba un menú principal con una GUI para entrar en el juego, salir de este o entrar en la pantalla de créditos del juego. Una vez hemos saltado de esa pantalla principal tenemos una escena introductoria que te explica el funcionamiento del juego, esta pantalla te explica las puntuaciones y las diferentes fases que posee el juego.

Una vez hemos terminado con la introducción pasamos al lobby y cuando todos los usuarios están listos para jugar da inicio la primera fase del juego, la relacionada con el laberinto. En dicha fase los jugadores que son ratones pueden realizar un movimiento por turno y los movimientos abarcan una distancia de una baldosa, es decir, cada jugador puede moverse sólo a sus casillas adyacentes. Cada ratón tiene asignado mediante el color un sistema de puntuación diferente.

Todos los ratones ganaban 1 punto por realizar un movimiento excepto el ratón verde que ganaba 4 puntos. Además todos los ratones ganan 100 puntos por comerse el queseo, pero el ratón de color rojo en lugar de ganar 100 gana 200 puntos. El ratón amarillo además de esto gana puntos rompiendo puertas, al contrario que el ratón azul que pierde puntos por realizar esta acción.

En esta versión del juego teníamos un chat libre en la mitad izquierda de la pantalla, que se usaba para el interrogatorio pero que se podía usar en todo momento.

Durante la fase del interrogatorio el jugador que juega el rol del científico debe hacer preguntas a los otros jugadores mediante el chat libre del que antes hemos hablado para descubrir quien es el culpable. Una vez el científico crea saber quien es el culpable de comer el queso sólo tenía que pulsar un botón para declarar un culpable y el juego terminaba.

Para la segunda versión del juego que fue usada para la últimas pruebas con usuarios se realizaron ciertos cambios que decidimos a raíz de la información extraída de la anterior prueba con usuarios realizada. De estos cambios hablaremos en los siguientes apartados de este capítulo.

Figura 6.1: Portada de ejemplo para MiceMaze

6.2. Primera sesión de pruebas con usuarios reales

En el mes de Mayo reservamos un día para realizar pruebas con usuarios y recabar información que nos ayudara a mejorar la experiencia y defectos del juego. Los cuatro usuarios que participaron en la sesión eran personas pertenecientes al grupo de investigación y profesores de la facultad, es decir, usuarios "dentro del mundillo". La versión del juego que pudieron probar no era la definitiva, pero si que contaba con muchos aspectos de la versión final por lo que los usuarios podrían hacerse una idea bastante aproximada de como llegaría a ser el producto final.

Las pruebas las realizábamos en grupos de tres. Dos de ellos serían los ratones y otro el científico que tendría que adivinar quien de los otros dos se había comido el queso. Durante las partidas uno de nosotros observaba como jugaban y les explicaba las mecánicas del juego a los usuarios, además de resolver las dudas que pudieran tener durante el desarrollo de la partida. Algunas de estas partidas se realizaban completamente por grupos de tres formados por personas ajenas al proyecto. En otras partidas participábamos alguno de nosotros por falta de personas. El director del proyecto también pudo jugar para ver los avances que teníamos, a modo de reunión como las que solemos tener cada dos semanas.

Las pruebas se realizaron en un laboratorio de la facultad. Para el desarrollo de las mismas pasamos un ejecutable a tres ordenadores del laboratorio. Uno de ellos haría de servidor y los otros dos deberían conectarse a la IP de ese ordenador, esta IP se la proporcionábamos a los usuarios en un documento de Google Docs. Además en este documento una vez finalizadas las partidas cada usuario podía darnos feedback.

6.2.1. Resultados

Víctor:

Poder ver el laberinto molaría. El chat debería mantener el foco al pulsar enviar. Podrías usar habilidades para verificar si alguna de las casillas u objetos rotos han sido visitados por algún jugador. La sensación que da es que aunque mientan es muy difícil saber cual de ellos dice la verdad y el chat se convierte en algo irrelevante. Debería de haber alguna forma de comprobar, de forma parcial, si un jugador puede estar mintiendo sobre algún hecho (las habilidades podrían ayudar). El investigador también podría tener una puntuación, basada en su capacidad para resolver en tiempo, habilidades usadas, mensajes enviados, mentiras detectadas, etc.

En la versión en la que jugó Víctor la visión reducida del científico no estaba disponible por lo que no le permitíamos que viera el laberinto para que la fase del interrogatorio no fuera "falsea", es decir, pensamos que si el sabía quien se había comido el queso al mirar el laberinto los resultados que obtendríamos de la fase del interrogatorio no serían válidos. En conclusión, este problema de la visión ya fue solucionado para la versión final.

El usuario encontró dificuldad en el desarrollo del interrogatorio. Durante la prueba observamos que tenía dificultades para desarrollar preguntas a los otros usuarios a los que estaba interrogando. Debido a eso cambiamos el chat libre que tenía esa versión por unas preguntas predefinidas que se puede ir construyedo con los diferentes valores que aportan múltiples combobox. Esto se ha implementado tanto para las preguntas como para las respuestas.

En cuanto a la puntuación lo apuntamos para el futuro. Ya se había hablado de ello en algunas reuniones, pero pensamos que realizar un justo sistema de puntuación no era la principal de nuestro trabajo.

Juan José:

El funcionamiento online está bastante bien. Explicaría mejor la función de cada ratón y como gana puntos cada uno, cuál es su función dentro del juego y cómo debe actuar. Faltaría mejorar las puertas que ha roto cada uno, poder ver las de los demás y donde acaba cada ratón (aunque esto se que no se ha implementado todavía). Mejoraría el chat final para hacerlo más fácil y entendible, quizás haciendo preguntas por turnos o de alguna otra forma. Por lo general el funcionamiento es correcto.

Este usuario encontró difícil el hecho de saber que función tiene cada ratón durante la partida. Debido a esto en cada visión del juego que tiene cada jugador aparece un texto bien visible que te indica con que actividades a realizar durante la partida puedes ganar o perder puntos. También veía difícil el chat del interrogatorio, la solución a esto ya la hemos comentado anteriormente.

Alfredo:

Esta bien es sencillo y fácil de entender, aunque es difícil saber en función de que se gana o pierde puntuación. Estaría bien el poder jugar viendo al otro personaje y las cosas que rompe, o al menos que pudiese haber dos modos de juego, uno ciego en el que como ahora solo ves tus movimientos y otro en el que pudiese ver como se mueve el resto de personajes. Estaría bien el que al final del juego las preguntas fuesen como en otros juegos en plan preguntas de tipo si o no y por turnos.

Este usuario tenía dificultad para saber como se gana y pierde puntos en el desarrollo de la partida, la solución a este problema ya ha sido planteada anteriormente en este mismo capítulo.

Al igual que Víctor, pensaba que la fase del interrogatorio sería más fácil con preguntas generadas que con un chat libre como teníamos en ese momento. Esto fue solucionado para la versión final.

En cuanto a que haya dos modos de juego, uno con niebla y otro sin ella, eso estará disponible en la configuración del proyecto en el archivo XML.

Javi:

Quizá el científico que te explica las cosas debería empezar con algún aviso de que él es el tutorial, para que los jugadores novatos no se lo salten. En el menú, la casita que me indica quién soy yo es un icono tradicionalmente asignado al host de la partida. Puede ser interesante resaltar de alguna otra manera (resaltar con algún color, ponerlo el primero de la lista..) quién es el jugador y dejar la casita para el que sea host. Sería interesante que la posición del queso fuera aleatoria en cada partida (igual la posición inicial de los ratones también). Si es un laberinto quizá esté bien que los ratones vean el recorrido pero no las puertas ni el queso. Es muy interesante que cada ratón tenga sus propios objetivos, quizá el laberinto debiera dar alguna opción más a aquellos ratones que NO quieran coger el queso y prefieran alargar la partida. (si se pueden romper puertas, por qué no repararla para que los demás pierdan tiempo) Si el científico tiene que ser un jugador, debería tener algo más de interactividad que la de ser un espectador durante las partidas. Pensad que la opción de que un espectador ajeno a la partida pueda ser el host puede terminar con ese espectador aburrido y que se vaya, dejando a los jugadores tirados.

Este usuario piensa que la escena introductoria que hay en el juego antes de llegar al lobby no parece un tutorial. Este problema se solventa con la explicación que tiene cada jugador durante la partida, dicha solución ya fue explicada anteriormente.

Veía poco intuitivos o liosos los iconos que hay en el lobby para saber que jugadores eres tú. Este problema al ser mucho menor que otros que encontramos durante estas pruebas con usuarios fue ignorado.

Además de estos fallos que experimentó mientras interactuaba con el software también nos recomendó incluir algunas nuevas opciones para el juego como que la posición del queso, esto es posible hacerlo mediante el archivo de configuración XML. Añadir la opción de reparar puertas, propuesta a la que dejamos la puerta abierta para desarrollar en un futuro. Que el científico pudiera hacer más durante la primera fase del juego, esto fue ignorado pues desde el principio del proyecto pensamos que el científico solo tendría poder en la segunda fase del desarrollo del juego aunque esto pudiera hacer que su rol fuera más aburrido que el de resto de jugadores.

6.3. Segunda sesión de pruebas con usuarios reales

A parte de las pruebas con usuarios relatadas antes, el 30 de mayo también realizamos pruebas con una versión más avanzada en la que ya se habían llevado a cabo los cambios pertinentes después de nuestra primera experiencia con usuarios.

En esta ocasión los dos usuarios no estaban relacionados con el mundo de la informática. Eran dos chicas que cursan su último año en el grado de publicidad y relaciones públicas en la UCM. Estos fueron sus impresiones.

6.3.1. Resultados

Laura:

Me gustaría que, tras comerse el queso, el jugador tenga más de un turno extra; de tal manera que para el científico es más complicado averiguar qué ratón ha sido el culpable.

Además, al empezar el juego ya se sabe qué ratón va a ganar, siendo una buena opción que en algunas ocasiones se bloquee alguno de los turnos, siendo así más equilibrada la partida.

Me gusta la opción del chat al final de la partida, ya que la forma de contestar de los jugadores puede condicionar a la decisión del científico.

Relacionado con los turnos extras. En la demo que probaron el juego tenía un máximo de 20 movimientos y al ser dos jugadores cada uno tenía 10 movimientos. Esta exigencia no es un problema pues simplemente hay que modificar la variable que tiene asignada el total de turnos por partida.

En cuanto a "bloquear turnos" es una opción que vamos a tener en cuenta, pero más que bloquear turnos hemos pensado en proporcionar a los jugadores un botón para saltarse el turno. Además de esto el usuario piensa que es fácil saber quien se ha comido el queso, pero esto es debido a la disposición del tablero. Hay esquinas del laberinto desde las que se tarda menos en llegar al queso pero esto no llega a ser un problema del juego sino de como se configure el aspecto del laberinto mediante el XML.

En las anteriores pruebas con usuarios, estos preferían un sistema de interrogatorio en el que las preguntas no fueran muy libres y parece que hemos acertado con este cambio pues a Laura le convence el sistema de preguntas predefinidas que hemos implementado.

Noelia:

No es complicado que el científico adivine qué ratón llega antes. Él no debería ver en su pantalla donde están los ratones mientras juegan.

Podrían moverse con la flecha del ordenador también.

Al igual que Laura, este usuario piensa que es fácil que el jugador que hace de científico adivine quien se ha comido el queso con el simple hecho de saber donde empieza cada ratón. Nos ofrece como solución que el científico no pueda ver donde comienza cada ratón.

Además a este usuario no le convence el sistema de movimiento y cree que deberíamos incluir la posibilidad de moverse mediante las flechas de dirección del teclado. Esta opción no creo que lleguemos a implementarla pues pensamos que es mucho mas intuitivo pinchar con el ratón en la casilla a la que quieres desplazarte.

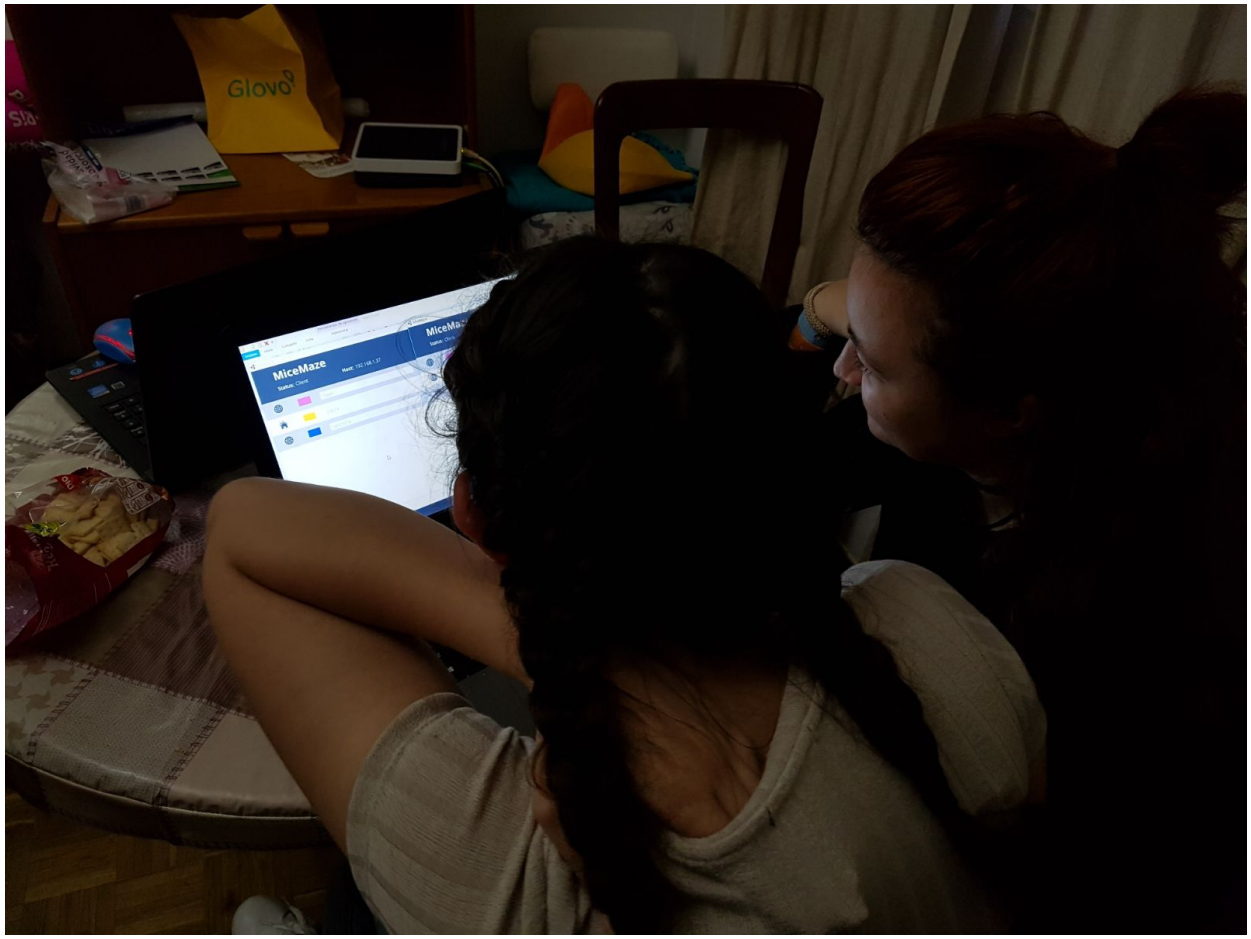

Figura 6.2: Usuarios en el lobby de la aplicación

Figura 6.3: Usuarios jugando en la primera fase del juego

6.4. Discusión sobre los resultados

De estos resultados podemos deducir que el juego convence pero que se puede mejorar, pues la mayoría de defectos (quitando uno o dos) que han encontrado los usuarios durante las pruebas no son muy graves más allá de mejorar aspectos visuales.

Y como información que podemos sacar de las respuestas obtenidas de los usuarios, nosotros como desarrolladores hemos observado que gran parte de los usuarios probaban el juego rápidamente ya fuera porque tenían prisa en acabar la prueba para marcharse a sus quehaceres o porque el juego puede llegar a ser aburrido. Esto último siempre lo hemos tenido en cuenta, pero pensamos que es bastante potente y que simplemente para hacer que sea más divertido es necesario añadir mecánicas más entretenidas para los usuarios.

Capítulo 7

Conclusiones

Como se ha establecido desde el Capítulo 1, el propósito de este proyecto es el desarrollo de una plataforma sobre Unity que sirva como herramienta para crear e investigar en videojuegos multijugador de deducción, pudiendo experimentar con jugadores humanos e incluso pudiendo conectar bots en un futuro.

Durante este año hemos podido aprender mucho acerca de Unity, el entorno de desarrollo más utilizado entre los proyectos de desarrollo independiente, y sobre todo de su sistema de juego en línea, ya comprendemos la arquitectura que crea Unity y como se intercambian eventos a través de los Command y ClientRpc.

El seguimiento constante de las reuniones cada dos semanas nos ha ayudado a valorar qué objetivos se estaban cumpliendo y cuales seguían estando lejos por alcanzar. Las sesiones de pruebas con usuarios, aunque hayan sido breves y sin contar con un gran número de sujetos, nos aportaron buenas ideas para el futuro, muchas de ellas formando parte del trabajo futuro que ha sido identificado para sucesivas versiones de la plataforma.

7.1. Objetivos alcanzados

En cuanto a los objetivos propuestos en la introducción, tenemos que detallar que se han alcanzado los siguientes:

1. Estudio del mercado de los juegos del género de la lógica y la deducción. Sabíamos que queríamos abordar el género de los juegos de deducción por lo que buscamos diferentes juegos de los que pudiéramos estudiar sus planteamientos y mecánicas. En un principio, durante las primeras reuniones, se realizaban ejercicios de "brainstorming". Muchas de estas ideas fueron tomando fuerza según veíamos que eran implementadas en juegos del género que habían tenido éxito en el mercado. Algunos de estos juegos son juegos de mesa como *cluedo*, otros de videoconsolas como *L.A Noire* (Team Bondi, 2011).

2. Estudio de las plataformas software que permitan experimentar con jugadores humanos y también con *bots* programados mediante IA. Esta revisión no resultó muy fructífera, porque apenas se han encotnrado herramientas para poner a prueba las capacidades de los jugadores de engañar o de descubrir mentiras. Por eso hemos llegado a la conclusión de que es un campo poco explotado en cuanto a aplicaciones software o por lo menos no es un fenómeno muy visible en el mercado del entretenimiento digital.

3. Desarrollo completo (especificación, análisis, diseño, implementación y pruebas) de una plataforma para la creación e investigación de este género de videojuegos, utilizando una metodologías y unas herramientas adecuadas para un proceso de desarrollo lo más sencillo y eficaz posible. La elección de una metodología fue fácil. Los primeros días no teníamos nada claro qué tipo de proceso de desarrollo podíamos seguir, pero pronto identificamos que lo apropiado era seguir una aproximación ágil, que nos exigía revisar cada pocas semanas los requisitos y la funcionalidad desarrollada con los clientes, representados por nuestro director y codirector. De alguna manera esta metodología nos permitió avanzar cada poco tiempo, aunque no fueran grandes incrementos. Este problema de avanzar cada dos semanas, en muchas iteraciones, pero hacerlo muy poco, en incrementos pequeños, se puede achacar a nuestro prácticamente nulo conocimiento inicial del entorno de desarrollo Unity.

A la hora de escoger las herramientas necesarias para desarrollar el proyecto también fue todo muy rápido pues el director nos recomendó unas opciones que nos parecieron sencillas y adecuadas para las necesidades del proyecto. Es cierto que también por inexperiencia hemos tenido continuos problemas con GitHub, solapando unos *commits* con otros y cuestiones así, pero nunca han sido tan graves como para perjudicar al desarrollo.

En paralelo a la especificación del proyecto realizamos un proceso de aprendizaje del entorno de desarrollo, Unity, y de su lenguaje de programación, C#. Realizamos varios tutoriales que ofrecían en el sitio web oficial, consistentes en pequeños juegos, algunos más sencillo y otros más complejos. La consulta de documentación oficial ha sido clave durante todo el proceso, aunque aquella relativa a Unity Multiplayer consideramos que es insuficiente y su modelo difícil de entender para los inexpertos, lo que nos hizo dedicar cerca de dos meses sólo para implementar el sistema de turnos en línea del juego.

4. Creación de un videojuego multijugador de deducción de ejemplo que ilustre el funcionamiento y el potencial de la plataforma anterior. La primera versión tanto de DeductionLab como de MiceMaze se diferenciaban con las finales en que no tenían funcionalidad multijugador y por lo tanto todos los jugadores jugaban desde el mismo ejecutable. Estábamos convencidos de que sería sencillo transformar esa versión local en la versión online, pero no fue así y hubo que descartar todo lo implementado, movimientos de los personajes por el tablero y demás mecánicas del juego, y reimplementarlo según el modelo de multijugador de Unity. Esto paralizó durante mucho tiempo el proyecto, aunque finalmente pudo retomarse el ritmo. La fase de simulación tiene las mecánicas básicas implementadas: movimiento de personajes, interacción con objetos como las paredes, y un sistema de visión donde se obtienen percepciones parciales del entorno. La fase de los interrogatorios, que cuenta con una interfaz que permite formular preguntas a partir de los personajes y otros elementos del escenario, también se implementó cuando ya estaba muy avanzado el proyecto.

5. Experimentación con el citado videojuego de ejemplo en una sesión de pruebas con usuarios reales, y validación por tanto de la utilidad de la plataforma desarrollada. La primera sesión de pruebas se realizaó cuando había varias mecánicas implementadas, pero no todas. La información obtenida, a pesar de que no provenía de un gran número de usuarios, resultó de mucha utilidad para hacer cambios y mejoras de cara a la versión final, que fue utilizada en la segunda y última sesión de prueba.

7.2. Trabajo futuro

Aunque podemos considerar tanto a *DeductionLab* como a *MiceMaze* primeras versiones estables, todavía hay mucho margen de mejora y hay objetivos por cumplir como parte del proyecto global de ayuda a la creación e investigación en videojuegos en el que participa este software. Concretamente estas son las líneas de trabajo futuro que se abren a continuación:

- Añadir objetos en los escenarios que requieran de una interacción más compleja, incluso colaboración entre varios personajes, lo que potenciaría las alianzas entre ellos.

- Añadir la posibilidad de realizar acciones positivas en la fase de simulación, tales como reparar o corregir lo que haya hecho otro personaje o incluso los errores cometidos por uno mismo. Estas acciones positivas podrían recibir recompensa en materia de puntuación siempre que sea reconocida la autoría de las mismas.

- Añadir más variaciones al sistema de visión que permita ver a los personajes a través de una superficie que no te permita reconocerlos del todo, por ejemplo.

- Añadir variantes en el conocimiento inicial de los personajes, por ejemplo no conociendo el escenario previamente y teniendo que explorarlo y aprender sus características según se está explorando.

- Añadir algún tipo de sistema económico que permita a los personajes intercambiar "dinero" (por ejemplo, la puntuación obtenida) en la fase de simulación a cambio de apoyos en la fase de interrogatorio.

- Desarrollar e integrar más juegos de ejemplo e incluso algún bot, aunque sea básico, que actúe como jugador virtual para estos juegos, como el propio *MiceMaze*.

- Mejorar la documentación, la organización y el estilo del código del proyecto que está publicado en GitHub, para aumentar los estándares de calidad y la optimización de la implementación.

- Mejorar el apartado artístico audiovisual y la experiencia de usuario de la aplicación, ya que se ha recurrido a interfaces muy simples y se han combinado recursos gratuitos de diferentes estilos, por falta de experiencia y medios para elaborar una aplicación mucho más atractiva para jugadores y también para diseñadores e investigadores del videojuego.

Aportaciones individuales de los autores

Víctor García Rodríguez

Al comienzo del proyecto, al no tener un alto conocimiento de Unity, aprovechamos para investigar y especificar requisitos. En este inicio de proyecto ya teníamos material para ir escribiendo una parte de la memoria. De esto me fui encargando yo en esta etapa inicial mientras mi compañero comenzaba con la implementación de la fase de simulación.

Mis aportaciones a la memoria abarcan casi todos los capítulos, a excepción del Capítulo 5 del que se encargó por completo mi compañero. Yo me ocupé de redactar el Capítulo 6 entero, mientras que el resto de capítulos han sido realizados a la par.

Como ya se ha comentado, durante el desarrollo del proyecto teníamos planificadas tres "reuniones generales" en las que nos reuníamos todos los participantes de Narratech Laboratories para mostrar los avances de nuestros proyectos (Trabajos de Fin de Grado, de Fin de Máster y hasta Tesis Doctorales). En la primera de estas reuniones, en la que aún no teníamos ninguna demo, me encargué de mostrar lo que queríamos hacer con *MiceMaze* usando un *mockup* hecho con Balsamiq.

Con respecto a la creación del juego de ejemplo, me encargué de realizar la interfaz gráfica de usuario y los menús, así como el sistema de creación de preguntas y respuestas de la fase del interrogatorio.

Utilizando un recurso de la Unity Asset Store implementé una escena introductoria en la que si pulsas sobre el personaje que aparece en pantalla, este te explica las mecánicas del juego mostrando bocadillos con texto. Esto parecía necesario para que el jugador no empezara muy perdido el juego y, de hecho, en la primera sesión de pruebas con usuarios reales, la mayoría de los usuarios coincidía en que era difícil saber cual era la misión asignada a cada uno de ellos.

Cuando empezamos con la segunda versión de *MiceMaze*, la versión multijugador, mi compañero y yo nos encaramos de manera conjunta de pasar de la versión local a la versión en línea. Esto nos llevo unas cuantas semanas debido a nuestro poco conocimiento acerca del entorno de desarrollo de Unity. En una de las reuniones de seguimiento que realizábamos cada dos semanas, llegamos a la conclusión de que era necesario un *lobby* para organizar de una manera más eficiente las partidas. Para ello al principio usé lo que me ofrecía Unity, pero al mostrar una interfaz gráfica bastante pobre, decidimos usar otra que obtuvimos también en la tienda oficial de Unity. Una vez retocada para que se adecuara a nuestras necesidades, la incluimos en el proyecto.

De la primera fase de *DeductionLab* se encargó mi compañero, pero yo añadí el sistema de puntuaciones y el temporizador para los turnos de los jugadores. Desde un principio teníamos pensado incluir la opción de "no realizar ningún movimiento" durante tu turno pero no fue hasta la segunda sesión de pruebas con usuarios que conseguimos sacar tiempo para implementarlo, coincidiendo con el hecho de que uno de los usuarios de prueba había mencionado que estaría bien incluir esa opción.

De programar la fase del interrogatorio me encargué al completo. En un principio constaba de un chat libre en el que los jugadores podían formular sus preguntas y respuestas. Había dos versiones y la diferencia entre ambas era el botón de declarar culpable que poseía la versión del jugador que tenía el rol de científico en *MiceMaze*. Tras discutirlo en varias reuniones de seguimiento y con los datos recabados durante la primera sesión con usuarios reales, decidimos cambiar el chat libre e implementar un sistema de preguntas y respuestas que estuviera definido y no diera libertad a los jugadores más allá de la combinación de posibles elementos que se permiten usar para elaborar una respuesta. Para esta nueva interfaz pensé que sería intuitivo añadir múltiples *comboboxes* para cada una de las opciones y combinar los valores de estos para que finalmente los jugadores (incluso virtuales, es decir, bots) pudieran formular sus preguntas y respuestas.

Jesús Menéndez Montejo

Mientras que mi compañero comenzaba a trabajar en la memoria con las investigaciones y especificaciones de requisitos, yo me encargué de la primera versión del escenario, el laberinto de *MiceMaze*. En una primera instancia el laberinto era estático, es decir, siempre se creaba con la misma forma. Tras la primera reunión concluimos que sería mas interesante si el laberinto se pudiese construir de manera dinámica en base a un fichero XML. Esta fue la segunda versión, cuyos laberintos estaban formados únicamente por baldosas y paredes exteriores.

Sobre este laberinto vacío incluí la primera versión de los personajes, ratones sospechosos en *MiceMaze*, y sus controladores. Al no disponer de arte el modelo fue un simple cubo. Con este modelo se implementó la primera versión del movimiento, que permitía mover al personaje directamente a cualquier punto del laberinto con un solo clic de ratón. En las versiones posteriores se restringió el movimiento del personaje a las casillas vecinas, siempre y cuando estuviesen vacías, ya que en este punto ya había añadido las paredes interiores. Una vez definido el movimiento creé la animación correspondiente para los sospechosos, que hasta ahora se transportaban al destino instantáneamente, sin animación.

Una vez creada la funcionalidad básica de los sospechosos, creé el primer sistema de turnos. Este sistema se tuvo que descartar al implementar el sistema multijugador en línea de Unity, ya que era una implementación pensada para funcionar únicamente en local.

Lo siguiente fue la creación de la comida a encontrar en el laberinto (el queso). Para este modelo utilicé un recurso gratuito de la Unity Asset Store. A parte de la creación del modelo incluí una acción para que el sospechoso interactuase (se comiese) la comida al situarse en la misma casilla.

Pasar a la versión multijugador nos costó varias semanas hasta entender bien como implementarlo. Una vez entendido lo primero que implementamos fue el sistema de turnos. Una vez ya con capacidades de juego en red, incluí las paredes que rompibles (los shojis), de apariencia mas fina. Una vez creado el modelo añadí otra acción a los sospechosos para que al entrar en contacto con una de estas paredes, esta se destruyera y permitiera el paso a través. Al contrario que las paredes normales, las paredes rompibles si permiten que te desplaces a su casilla.

En este punto, al igual que pasaba con el sistema de turnos tuve que modificar la implementación de las acciones "romper pared" y "comerse la comida" ya que, aunque el cambio se apreciase en el cliente que realizase la acción, no tenia efecto en el resto de clientes.

De manera simultanea cada vez que añadía un nuevo elemento al laberinto, modificaba el script que cargaba dinámicamente el laberinto para añadirlo y enriquecía el fichero XML de configuración del mismo.

Ya en las etapas finales del desarrollo incluí un sistema de visión entre los sospechosos distinto al de "visión total". De esta forma los sospechosos no eran capaces de verse entre ellos a no ser que se encontrasen en linea recta y sin paredes en medio, las paredes rotas, la comida y otros sospechosos no bloquean la visión, solo paredes y paredes rompibles que aun no hayan sido rotas. Este cambio me llevo a modificar de nuevo las acciones "romper pared" y "comerse la comida" para que la desaparición de la comida o la destrucción de la pared solo se conociese cuando el sospechoso viese ese elemento. De esta manera un sospechoso no tiene porque llegar a saber si se ha interactuado con uno de estos elementos, sin este sistema los sospechosos eran capaces de ver en todo momento al resto de sospechosos y sus movimientos, conociendo así quien era el responsable de cada acto.

En cuanto a la memoria me he encargado esencialmente del Capítulo 5 y de alguna sección como la dedicada a los retos del Capítulo 2.

Referencias

Assembly, C. (2014). *Alien isolation.* Multiplataforma.

BeautiFun Games. (2012). *Nihilumbra.* [Multiplataforma].

Blizzard Entertainment. (2014). *Hearthstone: Heroes of warcraft.* [Microsoft Windows, macOS, iOS, Android].

Bondi, T. (2011). *L.a. noire.* Multiplataforma.

Dimitry Davidoff, Philippe des Pallières y Hervé Marly. (2001). *Los hombres lobo de castronegro.* [Juego de mesa].

Faidutti, B. (1996). *Misterio de la abadia.* Juego de mesa.

Guerrero, J. C. D. (2007). *El club de los martes.* Juego de mesa.

Hoffman, H. (2015). *Aboard the looking glass.* [Microsoft Windows].

Niantic Inc. (2016). *Pokemon go.* [Android, iOS].

Nide, N., y Takata, S. (s.f.). Tracing werewolf game by using extended BDI model. En (Vol. 100).

Obsidian Entertainment. (2015). *Pillars of eternity.* [Microsoft Windows, OS X, Linux].

Pratt, A. E. (1949). *Cluedo.* Juego de mesa.

Quantic Dream. (2018). *Detroit: Become human.* [PlayStation 4].

Raymond Edwards, G. G., Suzanne Goldberg. (1900). *Sherlock holmes detective asesor.* Juego de mesa.

Team Bondi. (2011). *L.a noire.* [Multiplataforma].

Álvarez, N., y Peinado, F. (2015). Modelling suspicion as a game mechanism for designing a computer-played investigation character. En *Actas II Congreso de la Sociedad Española para las Ciencias del Videojuego (Barcelona, España, Junio 2015)* (Vol. 1394). CEUR-WS. Descargado de `http://ceur-ws.org/Vol-1394/`

Apéndice A

Title, Abstract and Keywords

Title

Development of a Platform for Creation and Research on Deduction Multiplayer Video Games

Abstract

Game development is a complext task that requires abstraction capability, analysis and meticulousness. Nowadays, games design is an academic discipline being imparted and researched in universities around the world. In the case of multiplayer deduction games, these tasks are challenging.

This project aims to simplify the creation and investigation of this type of games through a new assistant tool for professional designers and academics, that will allow them in the future to connect bots, also called virtual players, through its API.

The main goal is the development of this platform in Unity, DeductionLab, which must be highly configurable. Another goal is the creation of an example game called MiceMaze, that shows the platform's potencial to perform experiments in a comfortable and fast way.

In this work we study the games market, both digitals and analogical, from genres of detective, logic and deduction. For the development of the software we have used an agile methodology inspired in Scrum, with follow-up meetings with all the team each two weeks, as well as adecuated tools both to implement and to manage the project.

To validate the game we have realized two test sessions with real users. This feedback allows us to revise the focus of our work and above all, improve the tool performance. Our academic year was divided in three main milestones to show the work we did, and to be discussed and evaluated by the members of the team.

As a conclusion, we obtained the first stable version of this platform, free and open-source, that will help everyone interested in the research on deduction multiplayer games.

Keywords

Game design, Study of games, Development of Videogames, Entertainment Computing, Person-Computer Interaction, Software engineering, Agile methodologies, Negotiation, Argumentation, Logic, Detectives.

Apéndice B

Arte conceptual

Una estudiante de la UCM, Laura, se encargó de realizar el arte para MiceMaze. Dicho arte consiste en el diseño de los personajes y una portada de ejemplo para el juego. Estos diseños siguieron una serie de hitos que estaban enteramente relacionados con las especificaciones iniciales del juego de ahí que haya diferencias con respecto a la versión final. Dichos hitos era:

- Diseño de una portada para el juego que muestre el título del juego, el laberinto y a todos los jugadores (Figura B.1).
- Diseño para el personaje científico. Debe tener una bata de color morado debido a que es el color que tiene asignado en el juego (Figura B.2).
- Un diseño para cada ratón. Cada uno de ellos tiene un color distinto y una apariencia que haga evidente su rol en el juego, juguetón, glotón, malo y chivato (Figuras B.3 - B.6).

  En los ratones se puede apreciar un cambio respecto a la versión final. Ya que en un principio teníamos pensados esos roles y que estuvieran asignados al color, pero finalmente se descartó que los roles estuvieran asignados al color y el rol pasó a ser aleatorio, y también se descartó el rol del chivato que fue sustituido por el de bueno.

Figura B.1: Partada de ejemplo para MiceMaze

Figura B.2: Diseño del científico

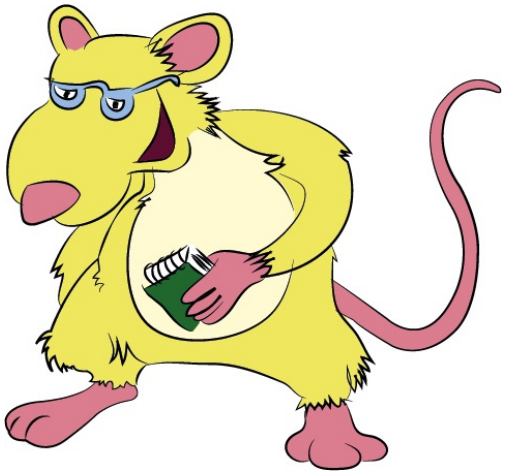

Figura B.3: Diseño del ratón chivato (Amarillo)

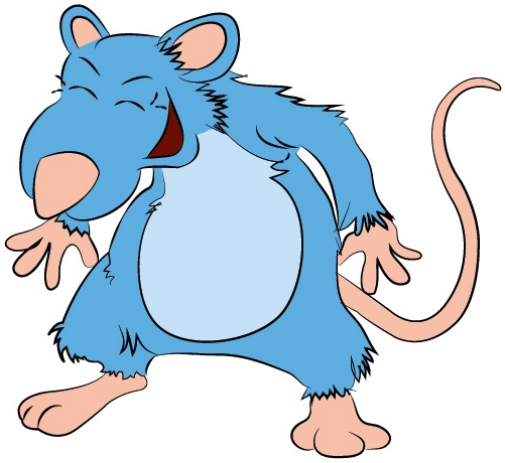

Figura B.4: Diseño del ratón pillo (azul)

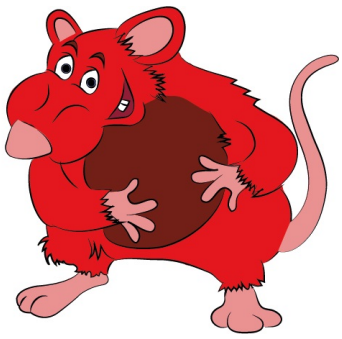

Figura B.5: Diseño del ratón glotón (rojo)

Figura B.6: Diseño del ratón malo (verde)

Apéndice C

Repositorio del proyecto

Enlace al repositorio del proyecto:

https://github.com/Narratech/MiceMaze