Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

D.A.N.G.E.R. – Disaster Prevention

Videojuego educativo para la gestión de emergencias en Unity 3D: Interfaz de usuario, gestión y tratamiento de la información

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Autor: Adrián Sánchez Lavarias

Tutor: Teresa María Pellicer Armiñana

Segundo Tutor: Ramón Pascual Mollá Vayá

Curso 2020/2021

Resumen

En este TFG se ha realizado un **videojuego** de **simulación social** y estrategia en tiempo real en **Unity 3D** para ordenador con una posterior portabilidad a dispositivos móviles. En este videojuego nos hemos encargado de la **Interfaz de Usuario,** incluyendo los menús necesarios y los ajustes, entre otras tareas del desarrollo de videojuegos como son el diseño y la programación de las distintas mecánicas.

Además, se ha realizado una **base de datos** que aloje las decisiones del usuario y la muestre en forma de estadísticas con el fin de que el usuario pueda aprender a partir de sus propias acciones.

Por último, hemos creado una **página web** en la que tener fácil acceso a los distintos enlaces de interés sobre cómo actuar en medidas de emergencia y prevenirlas.

Este juego está realizado junto a Pablo Querol Ballester por lo que se comparten elementos de este Trabajo Fin de Grado con el suyo, como son el análisis de mercado, la investigación en materias de seguridad y el documento de diseño del videojuego (GDD) entre otros.

El objetivo de este trabajo y por lo que hemos decidido que fuera un videojuego es enseñar las medidas que deben tomarse en caso de emergencia en un entorno interactivo, seguro y divertido con tecnologías modernas.

El videojuego se ha dividido en 2 modos de juego principales, la creación de la sala o edificio y otro de estrategia en tiempo real con simulación.

En el primero se puede construir una sala o un conjunto de estas, colocando los distintos objetos de prevención de incendios y posteriormente comprobar que todo sea correcto según la normativa.

El segundo se compone por personajes con distintas personalidades y emociones a los que debemos darles ordenes con el fin de salvar al mayor número posible de la emergencia en la que se ven envueltos.

Pablo se ha encargado de las distintas acciones del jugador, de los NPCs que están controlados a través de inteligencia artificial y de los niveles que son generados de forma procedural.

Palabras clave: Interfaz de Usuario, base de datos, página web, Unity 3D, videojuego.

Summary

In this TFG we have develop a **social simulation** and real-time strategy **video game** in **Unity 3D** for computer with a subsequent portability to mobile devices.

In this game we have been in charge of the **User Interface** including the necessary menus and settings. Among other tasks of game development such as the design and programming of the different mechanics.

We have created a **database** to store the user's decisions and show them in the form of statistics so that the user can learn from his own actions.

Finally, we have created of a **web page** to have easy access to the different links of interest on how to act in emergency measures and how to prevent them.

This game will be realized together with Pablo Querol Ballester, so elements of this Final Degree Project are shared with his, such as market analysis, research on safety issues and the video game design document (GDD) among others.

The objective of this work and why we have decided to make it a video game is to teach the measures to be taken in case of emergency in an interactive, safe and fun environment with modern technologies.

The video game it's divided into 2 main game modes, the creation of the room or building and other real-time strategy with simulation.

In the first one we will be able to build a room or a set of these, placing the different fire prevention objects and checking that everything is correct according to the regulations.

In the second we will find characters with different personalities and emotions to whom we must give orders in order to save as many as possible of the emergency in which they are involved.

Pablo is in charge of the different actions of the player, of the NPCs that will be controlled through artificial intelligence and of the levels that will be procedurally generated.

Keywords: User Interface, Data Base, web page, Unity 3D, videogame.

Resum

En aquest TFG hem realitzat un **videojoc** de **simulació social** i estratègia en temps real en Unity 3D per a ordinador amb una posterior portabilitat a dispositius mòbils.

En aquest videojoc ens hem encarregat de la **User Interface** incloent els menús necessaris i els ajustos. Entre altres tasques del desenvolupament de videojocs com són el disseny i la programació de les diferents mecàniques.

A més hem realitzat una **base de dades** que allotge les decisions de l'usuari i la mostre en forma d'estadístiques amb la finalitat que l'usuari puga aprendre a partir de les seues pròpies accions.

Finalment, la creació d'una **pàgina web** en la qual tindre fàcil accés als diferents enllaços d'interés sobre com actuar en mesures d'emergència i comprevindre-les.

Aquest joc ho hem realitzat al costat de Pablo Querol Ballester pel que elements d'aquest Treball Fi de Grau es compartiran amb el seu, com són l'anàlisi de mercat, la investigació en matèries de seguretat i el document de disseny del videojoc (GDD) entre altres.

L'objectiu d'aquest treball i pel que hem decidit que fora un videojoc és ensenyar les mesures que han de prendre's en cas d'emergència en un entorn interactiu, segur i divertit.

El videojoc es divideix en 2 maneres de joc principals, la creació de la sala o edifici i un altre d'estratègia en temps real amb simulació.

En el primer podrem construir una sala o un conjunt d'aquestes, col·locant els diferents objectes preventores d'incendis i comprovant que tot siga correcte segons la normativa.

En el segon trobarem personatges amb diferents personalitats i emocions als quals hem de donar-los ordres amb la finalitat de salvar a la major gent possible de l'emergència en la qual es veuen embolicats.

Pablo es l'encarregat de les distintes accions del jugador, dels NPCs que estarán controlats a través d'intel·ligència artificial y dels nivells que seran generats proceduralment.

Paraules clau: Interfície d'Usuari, base de dades, pàgina web, Unity 3D, videojoc.

Tabla de contenidos

Contenido

Índice de figuras

1. Introducción

Hoy en día el sector de los videojuegos sigue creciendo y se ha convertido en una de las principales fuentes de entretenimiento de jóvenes y adultos. Siguiendo este interés, el desarrollo de videojuegos también ha evolucionado llegando a ser verdaderos proyectos profesionales de software con numerosos equipos trabajando en las distintas partes. No cabe olvidar que los videojuegos también son arte, se forman en gran medida gracias al dibujo o modelado y la música. Por lo que son la herramienta perfecta para transmitir ideas o enseñar en un entorno agradable y controlado [1].

Por ello, este Trabajo de Fin de Grado tiene como finalidad desarrollar un videojuego educativo que enseñe como actuar en caso de incendio y que sirva como base para posteriores ediciones. En las que enseñar medidas de actuación de otro tipo de emergencias.

El trabajo se ha realizado por Adrián Sánchez Lavarias junto con Pablo Querol Ballester por lo que se comparten tareas como el diseño, investigación y análisis de mercado.

Adrián se ocupará de desarrollar el nivel de creación de salas, la interfaz, los menús y una base de datos que almacene las acciones del jugador. Además de una página web, en la que poner información sobre cómo actuar en caso de incendio e información de la base de datos.

Nuestro objetivo es desarrollar un proyecto lo más modular posible para que futuros alumnos puedan usarlo como base para expandirlo, utilizando las funciones ya descritas y la página web para enseñar.

1.1 Motivación

La principal motivación de este proyecto es conseguir de una forma divertida y segura enseñar las medidas de actuación en caso de incendio a una clase o grupo de alumnos.

Al pensar en una forma de conseguir esto pensamos que lo mejor sería enseñarlo de una manera interactiva en la que cualquiera pueda participar, por lo que llegamos a la conclusión de que un juego educativo sería la mejor opción.

Esto garantiza una experiencia personal en la que los alumnos tendrán que participar, jugando en una simulación que les enseñará cómo actuar en caso de que ellos se vean en una situación similar. Creemos que esto será una gran mejora a la hora de actuar o recordar los conocimientos que se desean enseñar comparado con unas transparencias o un vídeo [2].

1.2 Objetivos

El principal objetivo es desarrollar un videojuego funcional y educativo sobre procedimiento en emergencias. Nos enfocaremos en sirva para enseñar en clases o reuniones. De manera que se consiga información más cercana e interactiva, ofreciendo un trato personal para cada alumno. El proyecto se compone en distintos módulos separados que se podrán reutilizar para expandir el proyecto por otros alumnos. Nos hemos encargado de realizar distintas partes del desarrollo:

- 1º modulo o nivel: Creación de salas
- Splash
- Menú principal
- Menú de ajustes
- Créditos

Además de estas tareas dentro del propio videojuego también usaremos herramientas donde expandir lo enseñado en él:

- Página web
- Base de datos con las acciones del jugador

1.3 Estructura

La memoria se estructura en seis capítulos principales, en el primero encontraremos una introducción, las motivaciones que nos han llevado a elegir este proyecto y nuestros objetivos a cumplir. En el segundo hablaremos del estado del arte y del análisis de mercado para una aplicación similar a la nuestra en el contexto actual.

En el tercero encontraremos una lista de las distintas herramientas que hemos utilizado a lo largo del proyecto y una justificación de por qué las hemos elegido. En el cuarto se expone el diseño tanto conceptual como estructural del proyecto.

En el quinto capitulo se lleva a cabo una explicación de los distintos sistemas que componen el primer modulo o nivel, las implementaciones de la base de datos y la aplicación web con el juego.

En el sexto se exponen las distintas conclusiones del proyecto una vez finalizado relacionándolas con los estudios cursados. También se planteará una serie de trabajos futuros con el fin de poder expandir el proyecto.

2. Estado del arte

Los videojuegos se crearon con la finalidad de entretener, pero con el tiempo se ha expandido a mucho más. Hoy en día podemos encontrar videojuegos que sirven para transmitir ideas, como herramienta para expresar un sentimiento o con la intención de enseñar un concepto.

Si añadimos que la tecnología para jugarlos ha evolucionado con el paso de los años hasta convertirse en un producto accesible por la mayoría de gente, como son los *smartphones* [3].

Por esto y al ser nuestro objetivo que se pueda utilizar para enseñar en clases de alumnos u oficinas, utilizaremos ordenadores o smartphones para jugar. Aprovechando la tecnología para llegar a todo el mundo y un medio divertido para transmitir la información.

Al ser estudiantes y apasionados de los videojuegos nos interesan especialmente los juegos educativos como una nueva forma de enseñar [1]. Hay que destacar que este género de videojuegos no es el más popular, pero cuanto más avanza la tecnología más formas aparecen de crear mecánicas interactivas, innovadoras e interesantes con las que aprender mientras juegas.

A la vanguardia de las tecnologías actuales encontramos la realidad virtual o VR, un entorno en el que poder interactuar como si estuvieras físicamente dentro del videojuego. Lo que abre la posibilidad de simular lecciones o clases reales en el futuro, cuando está tecnología se popularice.

2.1 Análisis de mercado

Analizamos el mercado de juegos y aplicaciones cuya temática se centra en la simulación de emergencias, se incluyen en este análisis no sólo la temática de las emergencias enfocadas a incendios, sino también emergencias médicas, o de carácter laboral, en las tres plataformas más accesibles actualmente: páginas web, dispositivos móviles y ordenadores.

- Al analizar el mercado de videojuegos de simulación de emergencias en la web encontramos una decena de aplicaciones creadas en Flash, estos videojuegos se componen principalmente por juegos de memorización, juegos en los que relacionar palabras o incluso preguntas tipo test con los conceptos a enseñar.

  Cabe destacar también que la tecnología de Flash Player ya no está soportada de forma nativa por los navegadores web y esta se encuentra obsoleta. Es posible descargar extensiones y programas, así como páginas web especializadas para poder reproducir este contenido, pero para el usuario promedio estos contenidos serán obsoletos e inaccesibles.

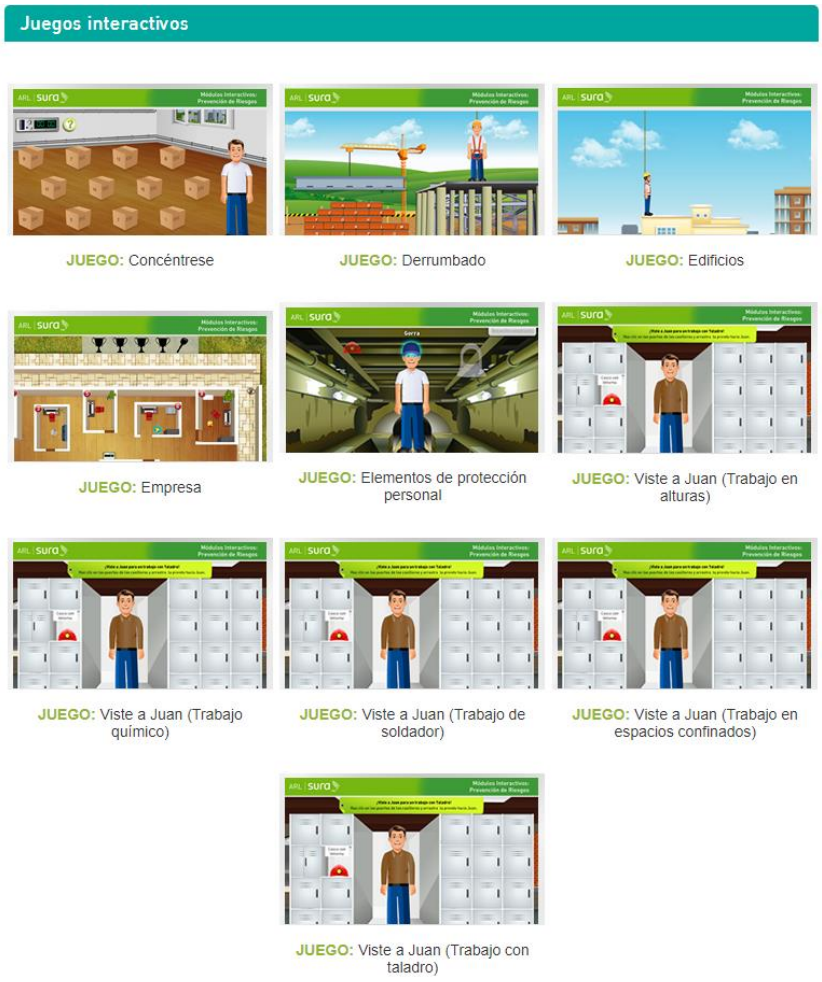


Figura 1. Fuente ARL Sura [1]

[1] https://www.arlsura.com/index.php/aprenda-jugando-independientes

- En el mercado móvil principalmente se encuentran aplicaciones que sirven como manual con la información en formato texto. Cabe destacar que también encontramos videojuegos que no siguen esta norma, como por ejemplo basados en secuencias animadas en las que debemos interactuar para completar la escena.

Figura 2. EMERGENCY HQ [2]

- Por último, en videojuegos de ordenador se pueden encontrar algunos proyectos en 3D creados que enseñan, mediante simulación de casos reales, que hacer en caso de incendio en una oficina u otras emergencias.

[2] https://play.google.com/store/apps/details?id=com.sgs.emhq.android&hl=es&gl=US

También encontramos juegos de estrategia como la saga de juegos EMERGENCY. Donde nos muestran una ciudad con distintas emergencias, incendios o gente herida. En la que debemos de gestionar a los personajes para solucionarlas:

Figura 3. EMERGENCY 20 [3]

Encontramos una cuarta referencia en el mercado que son los juegos de mesa. Generalmente los juegos de mesa orientados a la prevención y actuación en casos de emergencia, destinados a memorizar una serie de comportamientos de riesgo o preventivos y a trabajar la memoria. Pero no despiertan interés en los jugadores y todos los conceptos son representados como una situación abstracta que no aporta ningún grado de inmersión en la situación real. Podemos encontrar algunos ejemplos en la página web Información sobre Prevención de Riesgos Laborales[4].

[3] https://store.steampowered.com/app/735280/EMERGENCY_20/

[4] https://informacionprevencion.com/formacion/juegos-de-prevencion-de-riesgos-laborales/

- **PREVENCARD**

  Este juego de cartas tiene por objetivo inducir en la mente de los trabajadores la necesidad de identificar el riesgo, para poder prevenirlo y así aprender a protegerse.
  (Para jugar o ver más información haz Click aquí)

- **PREVENTE**

  En este juego de mesa, cada jugador deberá identificar correctamente los riesgos a que se expone en el trabajo y adoptar las medidas preventivas necesarias.
  (Para jugar o ver más información haz Click aquí)

- **9k1**

  En este juego deberás definir tu estrategia, cumplir tus metas, verificar si estás actuando correctamente y resolver las incidencias que te vayas encontrando, al final, conseguir cumplir tus objetivos de manera satisfactoria.
  (Para jugar o ver más información haz Click aquí)

- **ELIGE**

  Este juego trata de aventurarse hacia la toma de decisiones correctas. Ya que cada día tomamos multitud de decisiones, unas acertadas, otras no tanto. Si tuvieras la oportunidad de volver atrás cada vez que te equivocaras, si pudieras elegir, ¿qué harías?
  (Para jugar o ver más información haz Click aquí)

Figura 4. Información Prevención

2.2 Análisis del problema

Tras el estudio realizado en el estado del arte podemos obtener las siguientes conclusiones:

Entre los juegos web encontrados había una gran cantidad de juegos educativos, pero con mecánicas muy repetitivas y simples con muy poca interacción y basadas en memorizar la información, además de estar obsoletos e indisponibles para la mayoría del público.

Por otro lado, en los dispositivos móviles las mecánicas son algo más interactivas para involucrar al jugador en un procedimiento real en el que aprender jugando. Sin embargo, la cantidad de juegos educativos actualmente en el mercado móvil es bastante limitada, sobre todo para un juego tan amplio y de mecánicas variadas como el que se propone, por lo que podría considerarse que es un mercado con potencial.

Por último, los juegos comercializados en ordenador son muy reducidos, pero exploran mecánicas interactivas e inmersivas. La tecnología de realidad virtual puede ser una herramienta muy visual y que proporcione una mayor inmersión, pero actualmente sería inviable utilizar esta tecnología en cualquier sala para enseñar a varias decenas de personas, por lo que se ha decidido desarrollar un videojuego en formato estándar que pueda ser portable y no tenga unos requisitos exigentes para llegar al mayor número de personas posible.

Por estas razones hemos decidido centrarnos en las plataformas de ordenador y móvil. Cabe añadir que el campo de los videojuegos educativos todavía debe actualizarse y desarrollarse ampliamente para actualizarse a las nuevas tecnologías, proporcionando una formación mucho más completa e interactiva que llame la atención de las personas que participan en ella y les proporcione de unos conocimientos y habilidades duraderas en el tiempo.

3 Herramientas

A la hora de crear un proyecto, es de gran importancia seleccionar unas herramientas que se adecuen a nuestros objetivos. Y que nos ofrezcan una alta personalización y facilidad de uso a la hora de desarrollar nuestro proyecto. Para ello necesitaremos herramientas para crear el videojuego, para el tratamiento de la información y para la organización necesaria del proyecto.

3.1 Unity

Unity[5] es uno de los motores más utilizados en la creación de videojuegos actualmente. En Unity podemos crear juegos tanto en 2D como en 3D, por lo que sirve para crear múltiples estéticas y poder crear una gran gama de videojuegos.

La interfaz de Unity es sencilla y fácil de entender, por lo que se puede aprender rápidamente para crear proyectos profesionales.

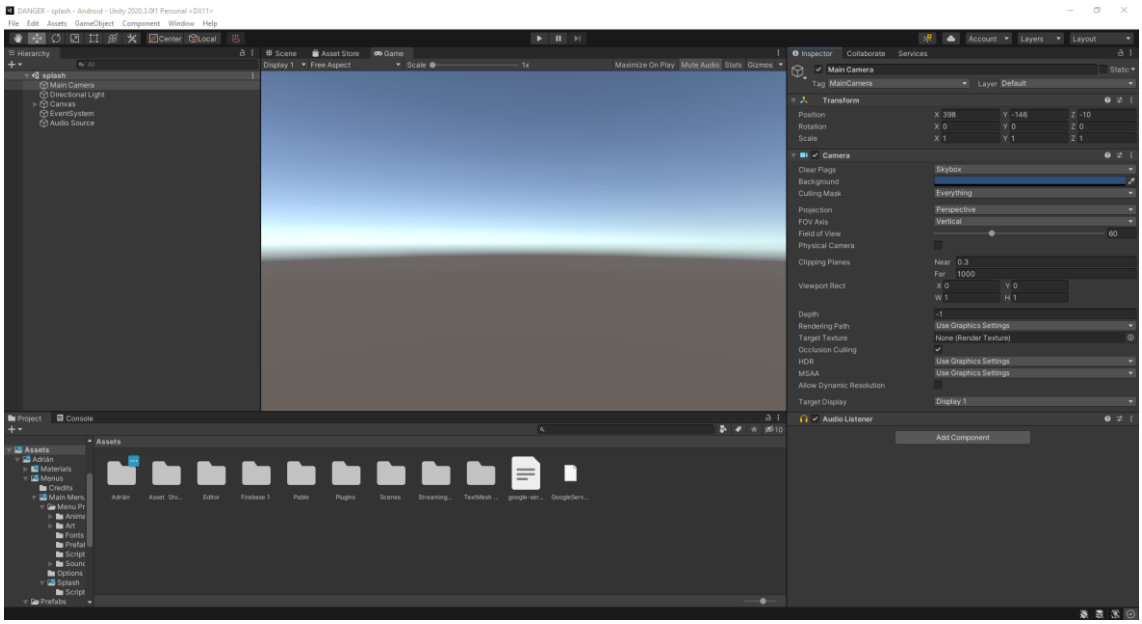


Figura 5. Interfaz de Unity

[5] https://unity.com/es

Este motor nos permite crear el juego una sola vez e implementarlo en más de 20 plataformas distintas, por lo que es ideal para videojuegos que salgan en diversos dispositivos. Soporta el lenguaje C# orientado a objetos y desarrollado por Microsoft, por lo que es ideal para gente que está empezando en la programación.

Unity cuenta con una tienda virtual llamada *asset store*[6] en la que existen multitud de *assets* que podemos añadir a nuestro proyecto fácilmente.

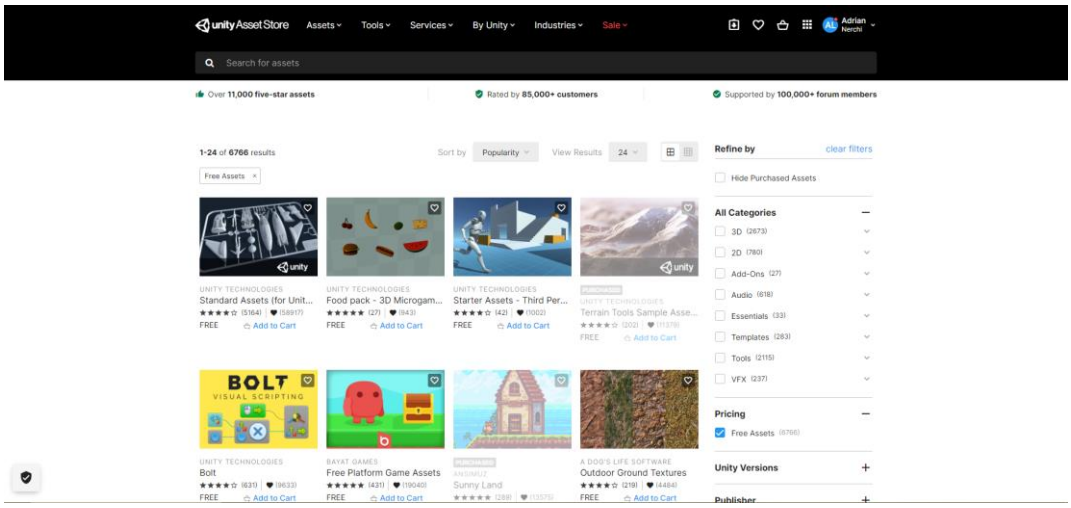


Figura 6. Asset Store

Además de los puntos anteriores, Unity ofrece un plan gratuito para estudiantes o particulares por lo que no nos extraña que sea uno de los motores más usados tanto profesional como personalmente.

Unity cuenta con un *hub* de versiones llamado Unity Hub[7] lo que nos permite administrar las distintas versiones de nuestros proyectos y sus extensiones.

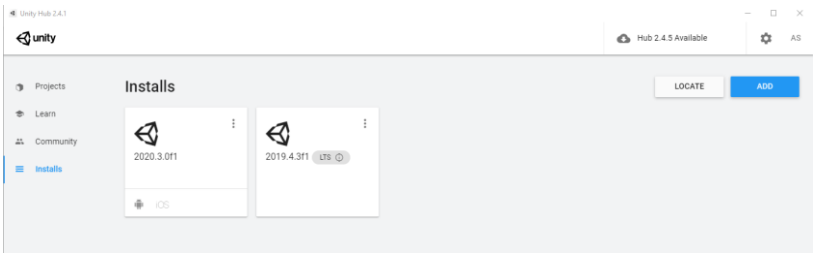


Figura 7. Unity Hub

[6] https://assetstore.unity.com/

[7] https://docs.unity3d.com/Manual/GettingStartedInstallingHub.html

Por estas razones hemos elegido Unity como motor de nuestro videojuego, ya que la portabilidad ha sido uno de nuestros principales objetivos para poder desplegar la aplicación en dispositivos móviles. Se adecua a nuestras necesidades como crear un videojuego 3D, los múltiples *assets* a nuestra disposición y ser una plataforma actual.

3.2 Visual Studio

Visual Studio[8] es un entorno de desarrollo creado por Microsoft que soporta múltiples lenguajes de programación y es uno de los editores de código más utilizados.

Además, tiene una extensión para programar más fácilmente en Unity por lo que el código será mucho más claro y nos sugerirá variables o funciones dependiendo de lo que escribamos en nuestro código.

Por último, nos permite hacer *debug* desde el editor y utilizar comandos Git lo que nos ha sido útil para el desarrollo.

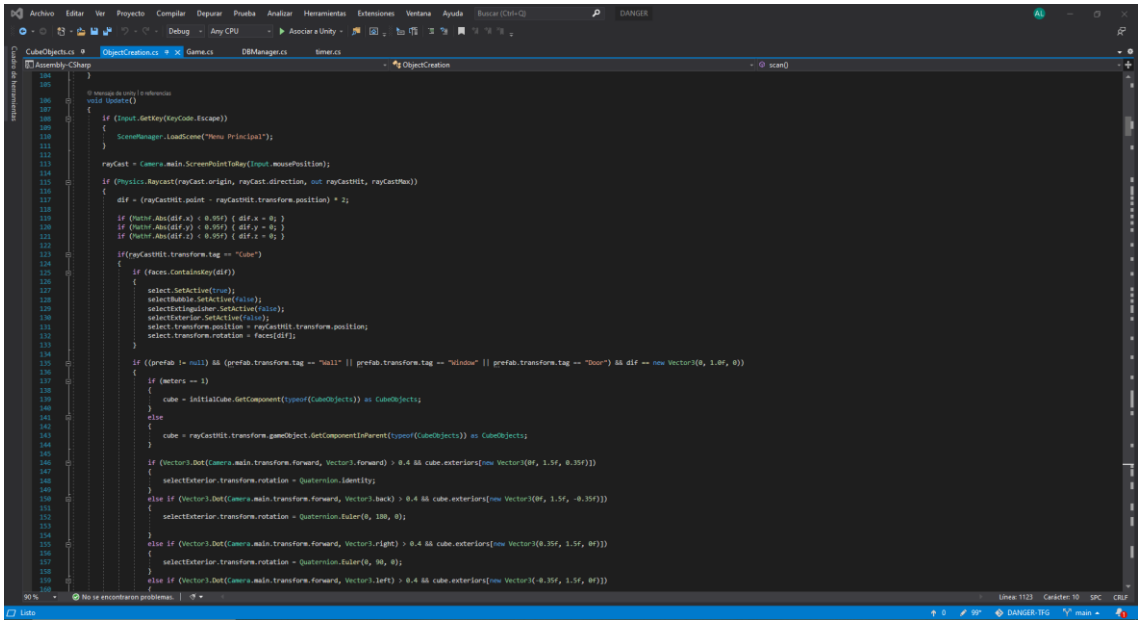


Figura 8. Interfaz de Visual Studio

[8] https://visualstudio.microsoft.com/es/

Es una elección indispensable si elegimos Unity ya que es uno de los mejores editores de código para programar en esta plataforma y se nos instalará junto con nuestra versión elegida en el Unity Hub.

3.3 Firebase

Firebase[9] es una herramienta creada por Google para ayudar a desarrollar y mantener aplicaciones en dispositivos móvil. A su vez también podremos conectarlo a Unity y a nuestra aplicación web para gestionar los datos entre nuestras distintas plataformas.

En esta plataforma encontraremos distintos servicios como base de datos, almacenamiento en la nube, autenticación de usuarios, *hosting,* detección de errores y *testing* entre otros.

Además, contamos con servicios relativos al negocio como analíticas o monetización para monitorizar nuestra aplicación una vez lanzada. También podemos conectar Unity a Firebase para enviar información a la base de datos desde él.

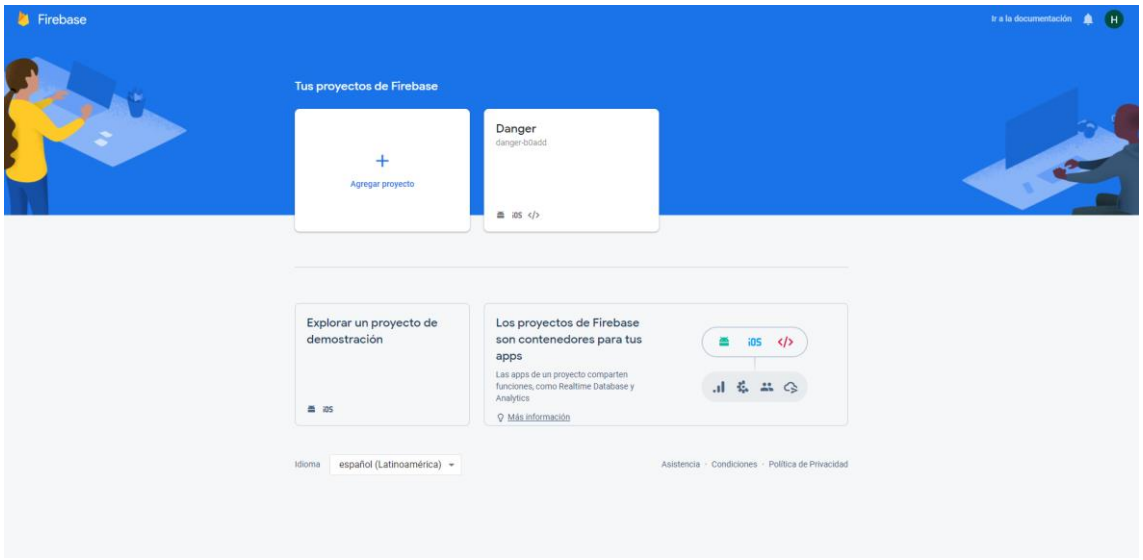


Figura 9. Interfaz de la consola de Firebase

[9] https://firebase.google.com/?hl=es

Por estas razones hemos decidido usar está plataforma para crear nuestra base de datos y para tener numerosas opciones de cara al futuro, como autenticación de usuarios y analíticas entre otros.

3.4 Angular

Angular[10] es un *framework* para el desarrollo de aplicaciones web creado por Google. Se basa en el patrón MVC (Modelo-vista-controlador) que se agrupan en componentes creando una aplicación web modular y eficiente.

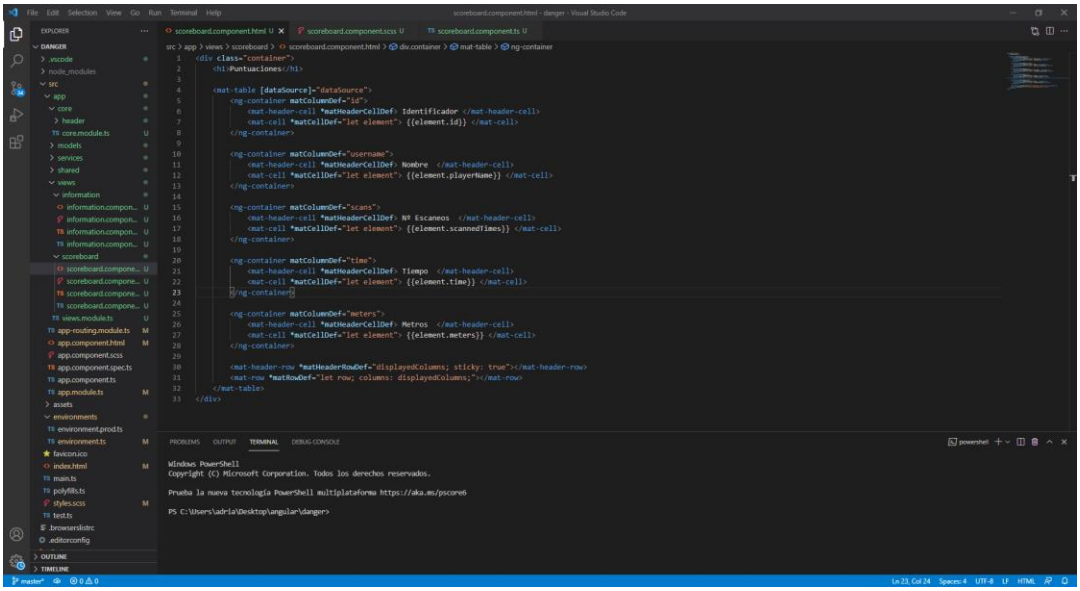


Figura 10. Interfaz Visual Studio Code

Usaremos Visual Studio Code[11] como editor de nuestro código. En nuestro caso conectaremos nuestro *front-end* con Firebase para recoger la información de la base de datos.

[10] https://angular.io/

[11] https://code.visualstudio.com/

3.5 GitHub

GitHub[12] es una herramienta que utiliza repositorios Git para crear versiones de control de cualquier proyecto.

En nuestro caso utilizaremos ramas para desarrollar las distintas partes del proyecto y ponerlas en común. Si el proyecto se ve alterado, podemos recuperar una versión anterior lo que es muy útil para solucionar errores en el proyecto.

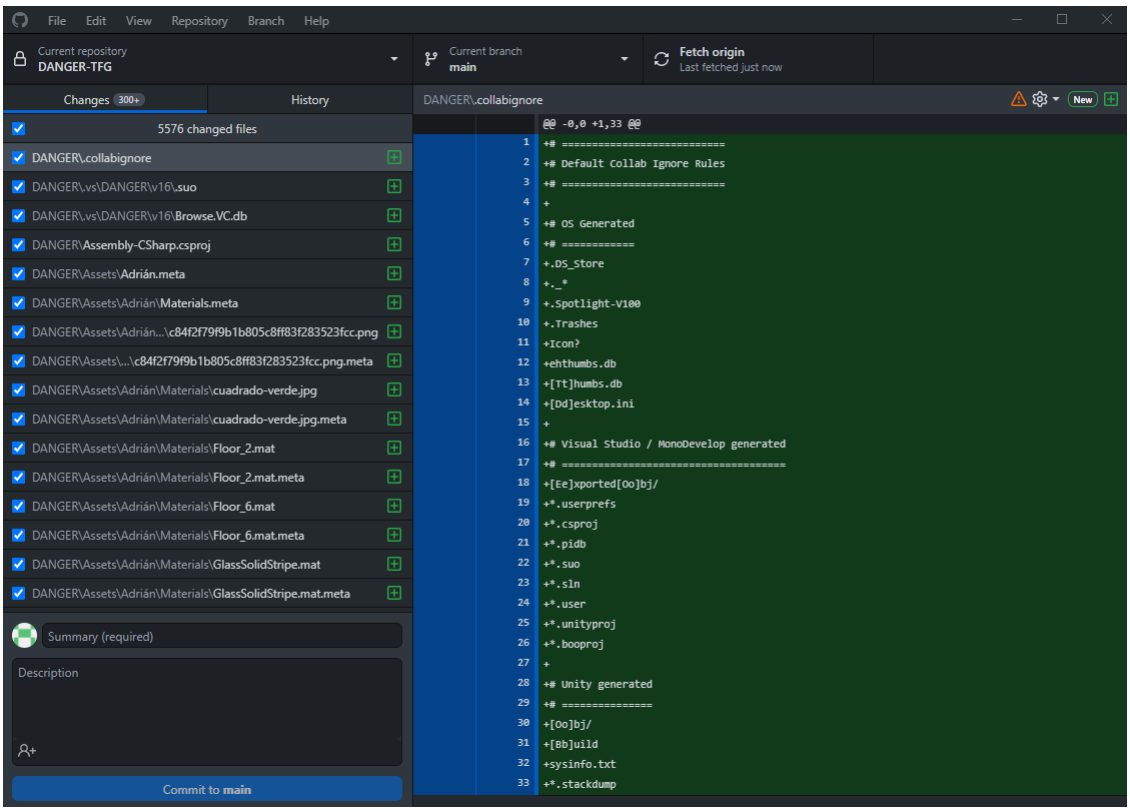


Figura 11. Interfaz GitHub

Por estas razones creemos que es una herramienta muy útil para llevar un proyecto entre varias personas con necesidades de organización y puesta en común de las diferentes tareas.

[12] https://github.com/

3.6 Trello

Trello[13] es un software con interfaz web y cliente para aplicaciones móvil creado para la administración de proyectos.

Nos permitirá gestionar nuestro proyecto creando tarjetas con las tareas a realizar, su estado, fecha de inicio, fecha de fin, responsable e incluso el color entre otros. Estas tarjetas las organizamos por estado para obtener una vista del proyecto completa en un solo panel.

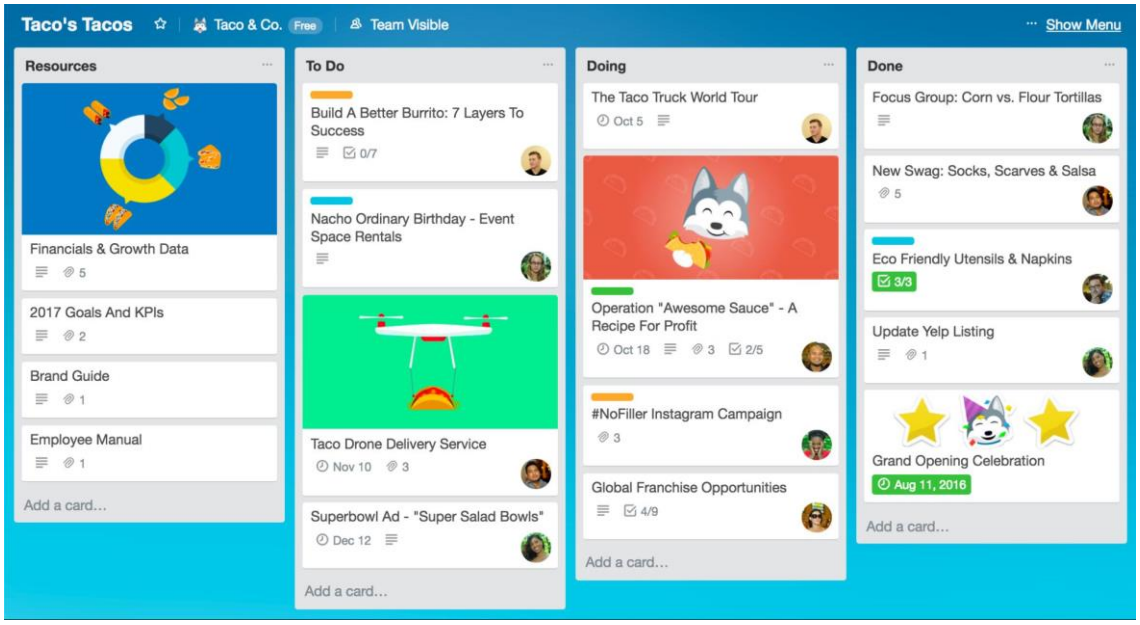


Figura 12. Interfaz de Trello

Elegimos Trello por la necesidad de controlar el proyecto día a día, su facilidad de uso y su extraordinaria personalización.

[13] https://trello.com/es

4 Diseño

En este apartado definiremos el proyecto de forma conceptual y estructuralmente.

Por un lado, observaremos las características de nuestro juego en el campo del diseño de videojuegos. Desarrolladas durante la fase de diseño en el *Game Design Document* (GDD) que encontramos en el Anexo 1.

Respecto al diseño estructural explicaremos las distintas partes que forman un proyecto Unity, llamadas escenas, en las que nuestro videojuego se divide en seis.

4.1 Diseño conceptual

Danger es un videojuego en 3D con dos niveles principales, en uno de ellos encontramos un juego del género de construcción y *puzzle*. Mientras que el segundo pertenece al género de la estrategia en tiempo real o RTS (*Real Time Strategy*) [4].

Enfocándonos en el primero, el jugador tendrá un espacio abierto y un bloque que sirva como base para comenzar a construir una o varias salas. El jugador podrá elegir distintos materiales con los que crear el suelo, además de poder colocar paredes, puertas, ventanas, mesas y extintores. El jugador tendrá completa libertad para crear la sala, mientras que dispondrá en pantalla de la cantidad de metros totales, el número de puertas, ventanas y extintores que se han colocado.

Con esta información el jugador tendrá que, dependiendo del número de metros total, aproximarse a la cantidad mínima de estos objetos que debe haber. Cuando crea que las salas están correctamente creadas, el jugador podrá pulsar el botón “*Scan*” para escanear las salas.

Esto puede dar pistas al jugador durante unos segundos si la disposición de las habitaciones no es correcta o un cartel que le indica al jugador que la sala es segura y ha resuelto el nivel. A este nivel se accederá desde el menú principal como podemos ver en el diagrama de flujo:

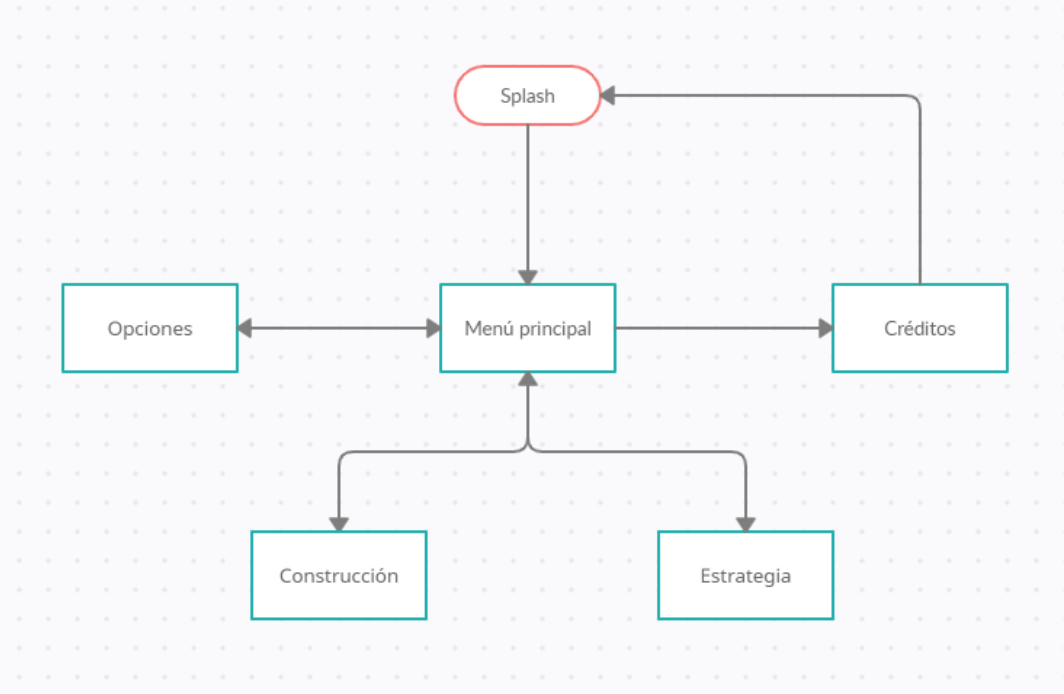


Figura 13. Diagrama de flujo de las escenas

4.2 Diseño estructural

El diseño estructural del primer nivel o módulo lo podemos dividir en cuatro sistemas: control del *User Interface*, control de la cámara, gestión de objetos y puntuación. Con estos 4 sistemas compondremos lo que el jugador verá y podrá realizar en la escena. Además de estos sistemas nuestro proyecto se compondrá por 6 escenas:

- Splash:
  Cinemática introductoria, se puede esperar o saltar con la tecla espacio o escape. Sirve como primer contacto con el jugador, por lo que hemos creado una escena móvil con música para introducir al jugador a nuestro juego.

- Menú principal:
  Una vez finalizado el Splash nos encontraremos en el menú principal, desde aquí podemos acceder a las demás escenas.

- Opciones:
  Esta escena servirá para que el jugador pueda cambiar entre pantalla completa o ventana, modificar el volumen, los gráficos del videojuego o la resolución a la que se muestra. Con el botón “back” volveremos al menú principal.

- Créditos:
  En esta escena hemos colocado las distintas tareas de las que nos hemos encargado cada uno en el proyecto en un fondo negro con música. Esta escena conecta con el Splash para que la navegación sea completa.

- Construcción:
  Este es nuestro primer nivel jugable, mantiene la música de las anteriores escenas, se compone por un entorno interactivo en el que construir salas de cualquier forma y tamaño.

- Estrategia:
  Para nuestro segundo nivel, contamos con un número de salas generadas de forma procedural, donde encontraremos un modo de juego de estrategia en tiempo real.

Figura 14. Escena Menú Principal

Figura 15. Escena Opciones

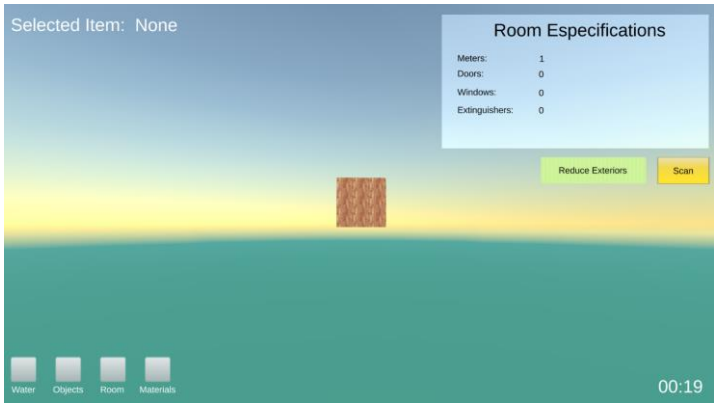


Figura 16. Escena Construcción

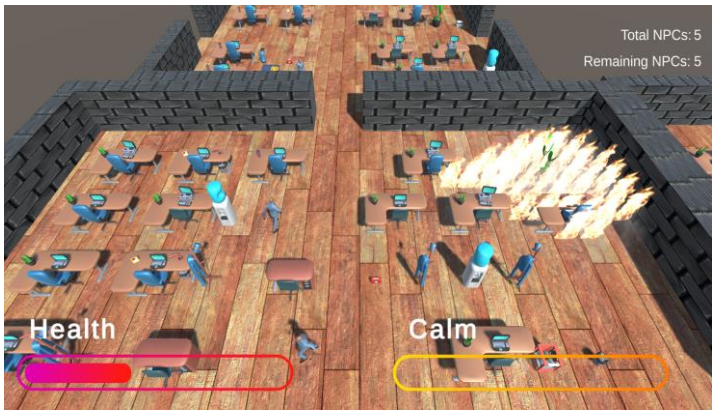


Figura 17. Escena Estrategia

Figura 18. Escena Créditos

Figura 19. Escena Splash

5 Implementación

El sistema de control de la interfaz de usuario controlará los botones de la escena y su comportamiento dando *feedback* al jugador cuando el juego se ejecute.

El control de la cámara permitirá al jugador moverse, girar y hacer *zoom* por la escena a libertad para poder utilizar el sistema de colocación de objetos correctamente.

En el sistema de colocación de objetos gestionaremos los distintos objetos que puede usar el jugador, conectándose al sistema de control de la interfaz para saber el objeto seleccionado. Nos permitirá colocar este objeto seleccionado en la escena con las reglas de colocación correspondientes para cada objeto.

Por último, el sistema de puntuación utilizará la información recabada en el sistema de colocación de objetos para transformarla en una entrada de la base de datos y enviarla. Unity cuenta con numerosos tutoriales con los que aprender desde cero a utilizar esta herramienta [5].

5.1 Control de la interfaz de usuario

El sistema de control de la interfaz de usuario se compone por 4 clases o scripts que añadiremos a nuestros botones principales que al pulsarlos mostrarán las distintas páginas asociadas a cada botón. Las páginas contienen a su vez botones con los que seleccionar el objeto o material a colocar por el jugador.

Con estos scripts garantizaremos un correcto funcionamiento de todos los componentes:

- Items Button:

  Este script lo añadiremos a todos los componentes botón que utilizaremos en la escena, se encargará de manejar los eventos de *clic*, entrada del ratón y salida del ratón. Como también de activar la página correspondiente a cada botón principal.

Cuando se inicie la escena con el botón activo, guardaremos la imagen del botón en una variable llamada *background* y suscribiremos este botón al objeto *item group.*
Unity nos permite añadir el objeto *item group* directamente desde la pestaña inspector de la interfaz de Unity.

Además, Unity maneja los distintos eventos del ratón que trataremos en esta clase y enviaremos al *item group* para su correcta resolución.

Por último, definimos un método *Select* y otro *Deselect* para conectar con otros componentes de Unity si fuera necesario de manera sencilla y desde la interfaz de Unity.

- Items Group:

En esta clase manejaremos los botones suscritos anteriormente desde la clase *item button* por lo que tendremos una lista de objetos compuesta por ellos junto con sus scripts y una lista de las páginas a activar.
Existen tres estados para los botones *Idle* como posición inicial, *Hover* como indicador de que es seleccionable cuando el ratón está sobre el objeto o *Active* cuando el objeto es clicado y se selecciona.
Cuando se activa un evento del ratón este llamará a la función correspondiente de su *Item Group,* cambiando entre los estados descritos anteriormente y activando la página relacionada con el botón, por lo que cambia en tiempo real cuando el juego está en ejecución.
Por último, encontramos un método *Reset* para devolver los objetos no activos a su estado inicial o *idle.*

- Items Page:

Este script se encargará de manejar las distintas páginas activadas en la anterior clase. Por ello se asociará con un *item button* y con un *item group* de manera que tengamos comunicados los botones con su correspondiente página dependiendo del estado de este.

Cuando se pulse un botón perteneciente a una página se enviará al sistema de colocación de objetos el nombre del material o *item* seleccionado.
Además, manejaremos el evento *Exit Hover* con el que la página se desactivará cuando nos encontremos en el estado *hover* y salgamos de este.
Guardaremos en una variable la imagen de fondo, crearemos los métodos *Select* y *Deselect* para expandir más esta clase si fuera necesario.

- Timer:

  En esta clase contaremos los segundos y minutos pasado mientras el jugador intenta resolver el nivel. Conectaremos el resultado de este script a un objeto de la interfaz para su visualización.
  En la esquina inferior derecha se puede observar un objeto de tipo texto que nos servirá como contador del tiempo transcurrido hasta resolver el nivel y que usaremos más tarde para el sistema de puntuación.

Por último, encontramos distintos elementos de la interfaz conectados directamente al sistema de colocación de objetos donde se obtiene la información necesaria a mostrar. En este apartado encontraremos el objeto seleccionado, los metros totales construidos, el número de puertas, ventanas y extintores. Para que el jugador pueda controlar adecuadamente el estado del juego, completándolo de manera clara y satisfactoria.

5.2 Control de la cámara

Este sistema está formado por dos funciones que llamaremos desde el sistema de colocación de objetos dependiendo del *input* del jugador.

Si el jugador pulsa la tecla *Left Control* mientas mantiene el clic izquierdo del ratón seguido de movimiento en cualquier dirección, la cámara se moverá por la escena siguiendo este movimiento. Esto se consigue guardando en una variable los ejes del ratón en forma de vector, multiplicados por el vector de la posición de la cámara y un parámetro que determinará la sensibilidad con la que se realizará este movimiento.

Del mismo modo, si pulsamos ratón derecho en vez del izquierdo ese movimiento se convertirá en rotación para la cámara.

En el caso de girar la rueda del ratón se acercará o alejará la cámara en la escena, esto se consigue guardando el valor de la rueda del ratón y multiplicándolo por un parámetro de sensibilidad que más tarde utilizaremos.

- Camera Recolocation:

  En esta función se asigna la nueva posición a la cámara principal, por lo que calcularemos un parámetro *position* con los datos recogidos anteriormente en el sistema de colocación de objetos y en la función *Rotation* para asignarlos a la cámara.
  Por último, indicaremos que la cámara se direccione hacia el bloque inicial.

- Rotation:

  En este caso pasaremos los ejes del ratón en forma de vector por un filtro que nos devolverá el valor resultante si está entre los parámetros *nearZoom* y *farZoom*, o en caso contrario siendo estos el mínimo y máximo que puede valer para el eje z. Hacemos lo mismo para el eje y, pero entre -89 y 89 ya que es un conjunto de valores adecuado para la visualización de la escena.
  Posteriormente transformaremos en ángulos el vector formado y lo asignaremos a un parámetro *Quaternion* utilizado en la función *Camera Recolocation* que llamamos al finalizar este proceso.

5.3 Gestión de objetos

Este es el sistema principal del videojuego ya que se encarga de las mecánicas principales y se conecta a todos los demás sistemas para dar o recibir información. La principal tecnología que utilizaremos en este apartado son los *Raycast,* una característica de Unity muy útil para nuestro propósito.

- Raycast:

  Utilizando el sistema de físicas de Unity [6] generaremos un rayo partiendo desde la posición del ratón en la escena y lo ampliaremos en dirección a la que está mirando la cámara. Este rayo colisionará con cualquier objeto en su camino que tenga un componente *collider*, dándonos la posibilidad de acceder a él desde el código.

  Al tener el objeto seleccionado podremos acceder a su posición, rotación, tag y componentes que utilizaremos colocar cada objeto.

- Prefabs:

  Como necesitaremos crear varias instancias de un mismo objeto con las mismas propiedades, Unity nos permite crear *prefabs* que servirán como plantillas para instanciar objetos desde el código. De manera que con un solo *game object* o una colección de estos en forma de *prefab*, podemos crear numerosos objetos que utilizar en nuestra escena.
  Estos *prefabs* pueden ser creados por el usuario o añadidos desde un *asset* externo a nuestro proyecto. Siendo fácilmente modificado para ajustarse a las necesidades de nuestro sistema.

  En nuestro caso creamos *prefabs* propios para crear un prototipo, que más tarde cambiamos por *assets* externos que le diera más credibilidad y color a la sala.
  Actualmente contamos con ocho tipos distintos de bloque, una puerta, una mesa, una ventana y un extintor. Esto puede ser fácilmente ampliable con materiales u objetos externos del proyecto para crear todo tipo de objetos o texturas con los que extender el proyecto.

- Tags:

  Cada *game object* en Unity tiene una propiedad *tag* o etiqueta. Lo que es muy útil para clasificar los distintos objetos y poder tratarlos por grupos dentro del sistema de gestión de objetos de forma sencilla.

- Dictionary:

  Estructura de datos compuesta por una *key* o clave y un *value* o valor. Los usaremos para relacionar una posición con un ángulo concreto en el caso de colocar un bloque o para relacionar una posición con un booleano o un *game object* en otras ocasiones.

Con estas herramientas a nuestra disposición hemos creado el sistema de gestión de objetos, cada objeto guarda información sobre su tipo en el *tag,* su posición y rotación en la escena.

Pero los bloques guardan tanto la anterior información como si tiene o no los distintos objetos que se pueden colocar y su posición relativa respecto al bloque.

Por lo que cada vez que creemos o eliminemos un objeto, se llamará al bloque en cuestión para modificar o consultar esta información.

También guardaremos el número de instancias de cada objeto y el objeto seleccionado en cada momento para enviárselo a la interfaz de usuario y que podamos mostrarle esta información al jugador.

Por otro lado, tenemos los selectores que son *quads* o planos que se ajustan en tamaño dependiendo del objeto seleccionado y la dirección. Existen cuatro tipos de selector distinto junto con su contraparte en nuestro proyecto. Se coloreará de verde el área donde el jugador podrá colocar el *item* seleccionado, esta área podrá ocupar toda una cara de un bloque, un rectángulo en la cara superior pegado a una arista para colocar los exteriores (paredes, ventanas y puertas) de la sala, un rectángulo centrado en las paredes para colocar un extintor o una burbuja que flota por encima del objeto.

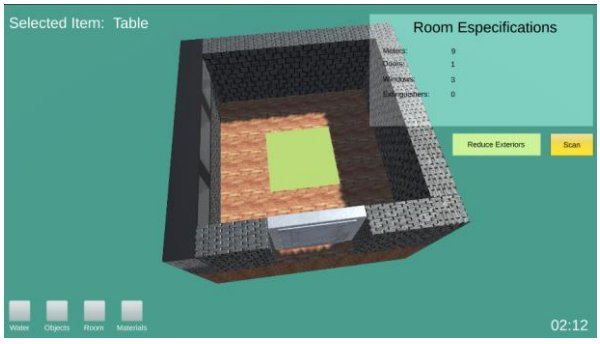


Figura 20. Selector Bloque

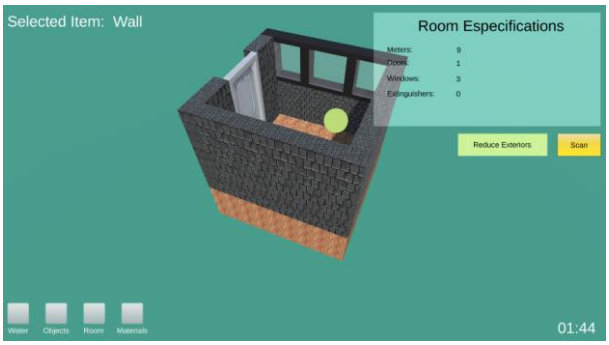


Figura 21. Selector Burbuja

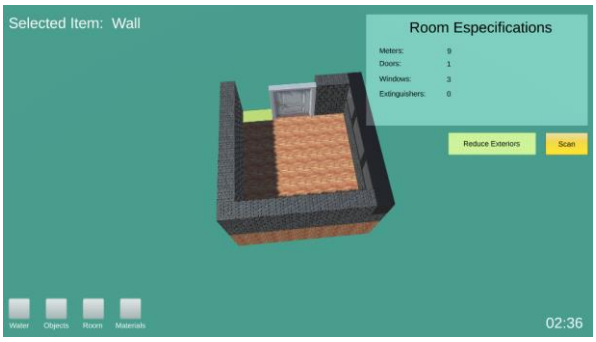


Figura 22. Selector Exterior

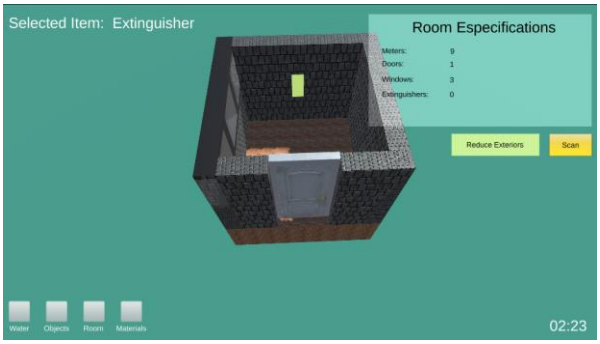


Figura 23. Selector Extintor

A la hora de crear la sala hay ciertas reglas dependiendo de cada objeto:

-Las puertas no pueden ir en las esquinas de la sala.
-Solo puede haber un extintor por bloque.
-Solo puede haber una mesa por bloque.
-Cada 15 metros totales debe haber un extintor [7]
-El máximo de objetos exteriores por bloque es de 4.
-Los bloques deben estar en posición “y” igual a cero
-El bloque inicial no se puede eliminar.

Cuando el jugador intenta poner un objeto que infringe alguna de estas reglas, el selector pasará de ser verde a rojo por unos segundos. Indicando al jugador de forma visual y entendible que no puede poner ese objeto en ese lugar.

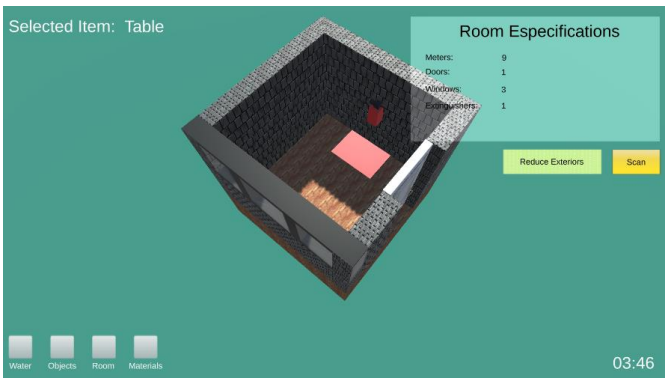


Figura 24. Selector Rojo Bloque

El sistema de gestión de objetos está inspirado por el conocido videojuego *minecraft* [5], la mecánica de poner y quitar bloques u objetos concretamente, con la intención de que sea más intuitivo e interesante para el jugador. Gracias a los selectores el jugador tendrá *feedback* directo sobre dónde y cómo poner el objeto seleccionado en cuestión, para después pulsar clic izquierdo para eliminar un objeto o clic derecho para colocarlo.

En el caso de crear un objeto comprobaremos que hay un selector activado, que estamos apuntando a un objeto y que tenemos un *item* seleccionado. Una vez hecho esto filtramos por el *tag* y accedemos a la información del bloque en el que pondremos el objeto. Después comprobaremos en qué dirección está mirando el jugador y si en esa dirección cumple con todas las reglas o infringe alguna llamando al bloque en cuestión. Si cumple todas las reglas se colocará en la posición calculada para esa dirección y se guardará la información de que existe en ese bloque y en las salas totales. Para ello se instanciará el *prefab* preparado anteriormente, creando un clon de este y colocándolo en la jerarquía correspondiente. Por último, actualizaremos el contador de objetos de ese tipo.

Para facilitar la visión del jugador y una construcción de la sala agradable, hemos implementado un botón con el que poder reducir los objetos exteriores de manera que el jugador pueda optar a una mejor visión con solo un clic. Por ello los ítems exteriores como paredes, ventanas y puertas tienen dos posiciones en vez de solamente una. Dependiendo de si el botón está activo podremos colocar estos objetos en su tamaño normal o en una escala de 0.5 en el eje “y” reduciendo su altura. Para este propósito existen dos diccionarios, uno para objetos exteriores y otro para los extintores, que relacionan una posición concreta de la escena con un objeto concreto. Esto nos permite transformar esos objetos fácilmente y eliminarlos por posición cuando sea el caso.

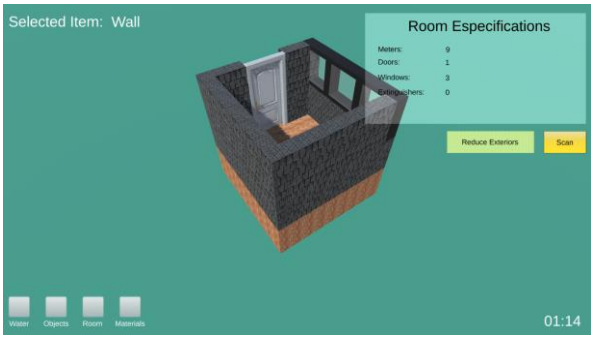


Figura 25. Habitación sin reducir

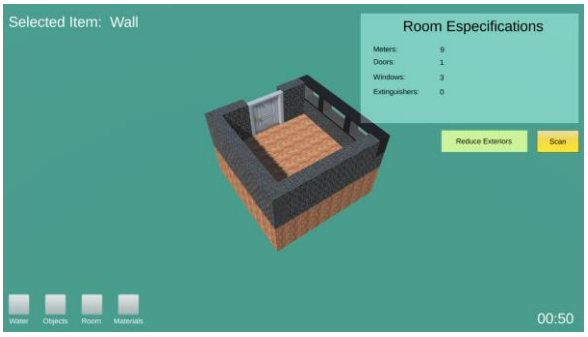


Figura 26. Habitación reducida

En el caso de eliminar un objeto, lo primero que haremos será comprobar que hay un objeto al que estamos apuntando y después filtrará por etiqueta para tratarlo según el tipo de objeto que sea.

Una vez hecho esto accedemos al bloque correspondiente donde está el objeto y comprobamos su posición para eliminarlo correctamente de la información perteneciente al bloque y también de algún diccionario si fuera necesario. Después se actualiza la cuenta de objetos y se destruye el objeto en cuestión.

5.4 Puntuaciones

El sistema de puntuaciones se compone principalmente de tres clases y también usaremos información recopilada por el sistema de gestión de objetos. Usaremos las clases *timer, Game* y *DBManager:*

- Timer:

  En esta clase calcularemos y daremos formato a el tiempo transcurrido desde que se inició el juego hasta que se resuelva el nivel. Una vez terminado el tiempo se detendrá y se guardará para ser utilizado más adelante.

- Game:

  Este script sirve para crear un objeto de tipo *Game* con sus constructores. Lo rellenaremos con la información recabada en el sistema de gestión de objetos: metros totales, número de veces pulsado el botón "*Scan*" y el tiempo recogido de la clase *timer.*
  Estos parámetros son los que indicarán nuestra "puntuación" que se guardará en la base de datos.

- DB Manager:

  En la clase siguiente inicializaremos la base de datos de Firebase, conectada a nuestro proyecto y a la aplicación web. Una vez lista, enviaremos la información pasada por la clase *Game* a la base de datos almacenándola para su uso posterior.

5.5 Firebase

Firebase es una plataforma sencilla, intuitiva y con mucha documentación útil [8]. Crearemos un nuevo proyecto en la consola llamado Danger, dejaremos habilitadas las *Analytics* de Google y seleccionaremos una cuenta nueva. Una vez creado el proyecto, accederemos al apartado *realtime database* donde seleccionaremos "Crear una base de datos", después elegiremos Europa en el desplegable en las opciones y pondremos las reglas a true para permitir acciones de lectura y escritura en la base de datos.

Necesitamos instalar las extensiones para Android y iOS que ya instalamos junto con la versión, pero es posible descargarlos más tarde desde el Unity Hub.

El próximo paso es configurar el *package name* de Unity, navegaremos a:

File > Build Settings > Android > Project Settings > Player > Package Name

Cambiaremos este nombre por “com.TFG.DANGER” que es el nombre único de nuestro proyecto en Firebase.

Una vez hecho esto, registraremos nuestra aplicación en Android, iOS y Web App con el nombre del paquete anterior. Al registrar la aplicación nos permitirá descargar un archivo de configuración llamado “google-services.json” que colocaremos en la carpeta “*Assets*” de nuestro proyecto Unity.

Con estos pasos la conexión con Firebase ya estaría creada, pero todavía no tenemos acceso a la *realtime database* desde Unity.

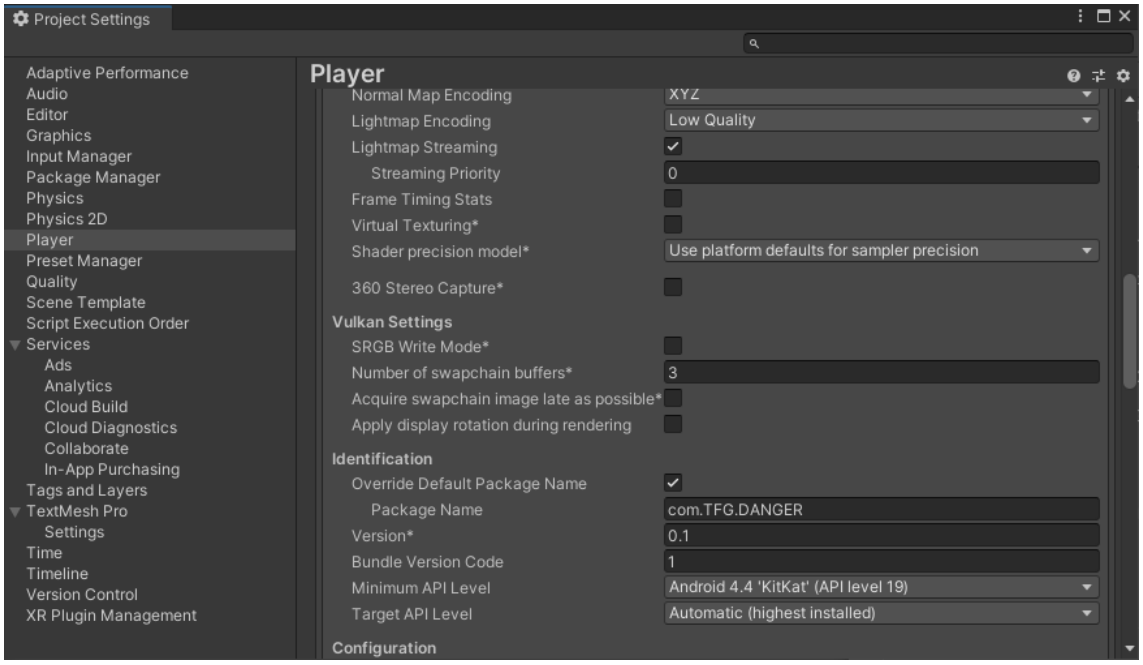


Figura 27. Configuración de proyecto

Primero configuraremos la *realtime database* para aceptar peticiones de lectura-escritura, entonces descargamos el SDK de Firebase para Unity. Utilizaremos la versión dotNet4 y dentro de esta carpeta importaremos el archivo “*FirebaseDatabase*” a nuestra carpeta “*Assets*” del proyecto.

Para enviar información a la base de datos usaremos la clase “DB Manager”. En ella crearemos una referencia a la base de datos de Firebase desde la que obtendremos una instancia. En la consola en el apartado *realtime database* podemos encontrar el

enlace proporcionado por Firebase. Una vez hecho esto podemos enviar un objeto, en nuestro caso *Game*, con una ID única y recibirlo en la base de datos en tiempo real.

Por último, configuraremos Firebase para conectarlo con Angular que será la herramienta con la que creemos nuestra aplicación web. Nos dirigimos a los ajustes del proyecto, en el apartado “Tus apps” seleccionaremos aplicación web con el símbolo </>. Después registramos la aplicación poniendo un nombre, en nuestro caso Danger.

5.6 Aplicación Web

Para el desarrollo de nuestra aplicación web utilizaremos Angular junto con Visual Studio. Utilizaremos la documentación de Angular como guía y apoyo para crear nuestra aplicación web [9]. Instalaremos Visual Studio Code como IDE de desarrollo, Nodejs[14] como entorno de ejecución para JavaScript y por último el CLI de Angular[15] con el comando:

```
npm install -g @angular/cli
```

Empezaremos creando el proyecto desde la herramienta Angular CLI, para ello abrimos una consola msdos donde tenemos instalado y enlazado Angular CLI. Nos situamos en el directorio donde deseamos crear el proyecto “danger” y ejecutamos el comando:

```
ng new danger
```

En las opciones presentadas elegiremos que sí a añadir Angular *routing* y scss como formato de las hojas de estilo.

Una vez completado, podemos arrancar el proyecto ejecutando en la consola el siguiente comando, siempre dentro del directorio creado:

```
ng serve
```

Angular compilará el código fuente de nuestro proyecto, creando un servidor local alojado en la dirección: http://localhost:4200/

[14] https://nodejs.org/es/

[15] https://angular.io/cli

Para finalizar la creación del proyecto, instalaremos “Material” que está perfectamente integrado en Angular y nos permitirá elegir una paleta de colores predefinida o elegir una *custom* con el siguiente comando:

```
ng add @angular/material
```

Ahora formaremos la estructura de la aplicación, creando los módulos generales llamados *shared, core, models, services y views*. En estos módulos organizaremos los componentes creados posteriormente dependiendo de su tipo o uso. Para ello ejecutamos el siguiente comando con el nombre de cada módulo:

```
ng generate module shared
```

El primer componente que crearemos será el *header* en *core* con el comando:

```
ng generate component core/header
```

En los componentes utilizaremos las etiquetas de material además de las HTML por lo que necesitaremos importar las dependencias oportunas a los diferentes módulos. Por otro lado, en estos módulos podemos declarar los componentes y exportarlos para su uso en otros módulos de la aplicación. Ahora construimos el *header* a nuestro gusto y añadimos su etiqueta a la página general de la aplicación “app.component.html” para añadir este componente.

Por otro lado, crearemos los componentes *information* y *scoreboard* dentro de *views* ya que serán las dos principales vistas de nuestra aplicación. Para el componente *scoreboard* necesitaremos un servicio que se encargue de recoger y tratar la información de la base de datos, así que lo creamos en el módulo *services*. El componente *scoreboard* estará formado por una tabla con la información de la base de datos.

Por último, crearemos un modelo *Game*, que servirá para tratar la información recopilada por la base de datos y enviarla adecuadamente al componente *scoreboard* para su correcta visualización.

Una vez nuestra aplicación web está lista debemos conectarla a la base de datos de Firebase:

Al utilizar Angular podemos usar npm, por lo que instalaremos el SDK directamente desde la consola escribiendo:

npm install --save firebase.

Esta acción nos solicitará acceso a nuestra cuenta de Firebase que debemos permitir para continuar.

Una vez instalado copiaremos en el archivo *environment* la configuración para Firebase, importaremos a nuestro servicio las dependencias necesarias e inicializamos Firebase en nuestra aplicación web.

Por último, con la aplicación Firebase que acabamos de inicializar creamos una referencia a la base de datos que utilizaremos para recibir información. También crearemos una función llamada desde el componente *scoreboard* que a su vez utilice la referencia a la base de datos, recuperé la información y la devuelva al componente.

Figura 28. Vista Puntuaciones

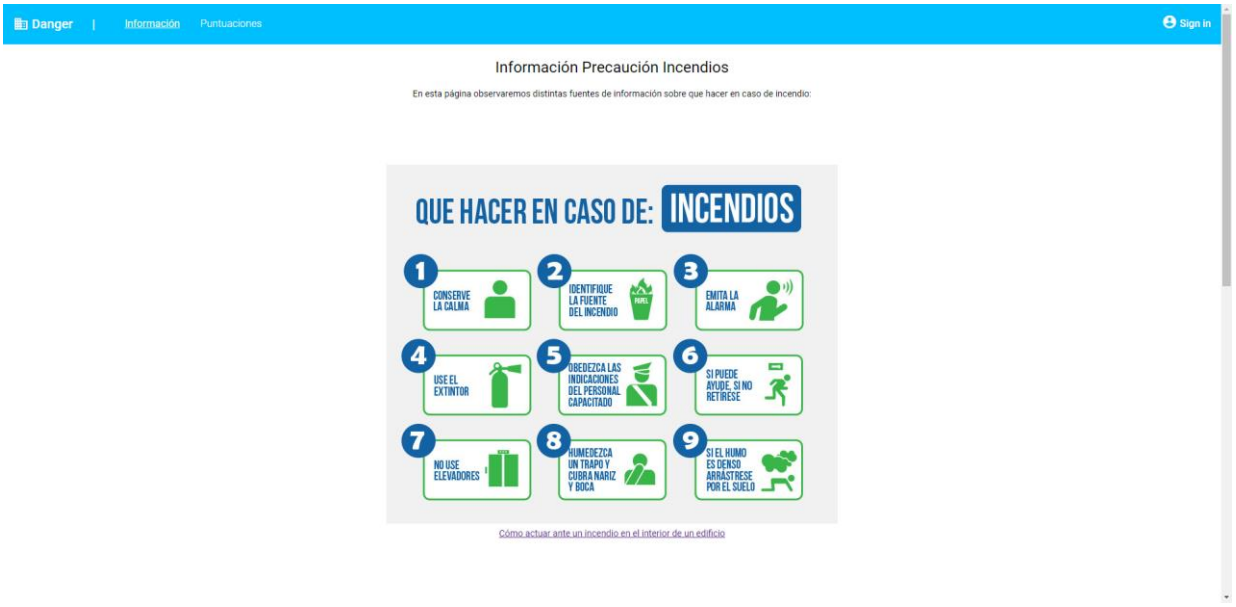


Figura 29. Vista Información

6 Conclusiones

El resultado de este trabajo ha sido muy satisfactorio, hemos creado un videojuego educativo funcional y completamente original junto con las escenas correspondientes para ser un producto de software completo y terminado. Conectado por medio de una base de datos a una aplicación web donde expandir la información. El videojuego cuenta con un *splash*, un menú principal, un menú de opciones y créditos.

Pese a que estamos contentos con el resultado y es completamente funcional, nos hubiera gustado trabajar más en la aplicación web y su implementación con Firebase. Ya que este último no lo habíamos utilizado anteriormente y nos ofrece múltiples servicios entre otros gestionar usuarios o alojar la aplicación. También nos hubiera gustado personalizar más la web y conectarla a un *back-end* creado con Spring.

Este ha sido nuestro segundo proyecto en el sector de los videojuegos utilizando el motor de Unity, junto con el lenguaje c#, por lo que la experiencia y soltura conseguida con este trabajo es muy satisfactoria tanto personal como profesionalmente.

En los últimos meses hemos aprendido a utilizar Angular para desarrollar aplicaciones web en el lenguaje TypeScript lo que nos ha servido para ampliar gratamente el conocimiento de desarrollo de páginas web que teníamos. Al principio aprender Angular fue complicado porque es una forma nueva de crear páginas web, nosotros teníamos conocimientos de HTML, CSS y JavaScript principalmente donde se divide una página en vista, estilo y controlador.

En Angular toda nuestra aplicación está compuesta por componentes individuales con su propia vista, modelo y controlador que se agrupan a la vez en módulos dependiendo de su uso o tipo de componente. Además, TypeScript utiliza numerosas etiquetas que son realmente útiles, pero dificultan comenzar de cero en este lenguaje.

Inicialmente tuvimos problemas para sincronizar los distintos componentes, ya que al estar formado por numerosas partes debes importar, exportar y declarar tanto componentes como paquetes entre los distintos módulos correctamente para que funcione la aplicación. Para resolverlo rehicimos los *imports*, importando los paquetes, declarando los componentes y exportándolos desde el correspondiente módulo.

Por último, hemos conseguido más experiencia en la gestión de proyectos y control de versiones gracias a utilizar Trello y GitHub con un equipo.

6.1 Relación con los estudios cursados

Al trabajar con varias tecnologías con propósitos tan distintos, muchos conocimientos que obtuvimos durante los estudios han aportado a que hoy seamos capaces de crear proyectos de software de esta complejidad.

Existen muchas asignaturas que nos han ayudado a conseguir estos objetivos como las clases introductorias a la programación, pero destacaremos las siguientes:

- Interfaces persona computador para la creación de las distintas interfaces del proyecto junto con Desarrollo centrado en el usuario y Desarrollo web donde expandiríamos estos conocimientos con desarrollo web.

- Gestión de proyectos e Ingeniería del software para la correcta organización de un proyecto, tanto externa como internamente.

- Introducción a la programación de videojuegos por ser el primer contacto con Unity, donde aprendimos a trabajar con esta herramienta y ganamos experiencia con el lenguaje C#.

Por último, nos han sido especialmente útiles las siguientes competencias transversales:

- “Planificación y gestión del tiempo” y “Trabajo en equipo y liderazgo” a la hora de desarrollar y gestionar en equipo el proyecto a lo largo de los meses de realización.

- “Innovación, creatividad y emprendimiento” al crear un proyecto original con múltiples herramientas de vanguardia y “Análisis y resolución de problemas“ para crear soluciones efectivas a los objetivos de nuestro proyecto.

6.2 Trabajo futuro

Al comenzar este proyecto se planteo de manera que fuera lo más modular posible para su reutilización posterior por otros alumnos. Respecto a esto creemos que el proyecto es muy ampliable de cara a nuevos modos de juego o añadir nuevas características a los ya creados.

Para expandir el nivel de construcción podemos crear un nuevo tipo de objeto con un botón, le añadimos los scripts de UI explicados anteriormente y creamos una nueva página a enlazar con este botón mediante los scripts.

También es sencillo añadir nuevos objetos dentro de las páginas, creando igualmente un nuevo botón con las funcionalidades ya implementadas y otorgándole un nuevo nombre. Por último, añadiríamos un *asset* de la tienda o uno creado por nosotros y lo ajustaríamos dependiendo de su tamaño para que se coloque correctamente sobre el bloque u objeto. Podemos modificar los *assets* actuales con nuevos materiales o texturas, otorgándoles una nueva estética y creando un nuevo objeto.

Por otro lado, es totalmente viable crear nuevas temáticas o modos de juego de incendios u otro tipo de emergencia utilizando este proyecto como base. Creemos que los modos de juego de construcción y de estrategia se pueden aplicar a multitud de escenarios para enseñar.

Podríamos expandir la aplicación web con la ayuda de Firebase que nos pone a disposición diferentes servicios como por ejemplo gestión de usuarios. Lo que sería muy útil para ver las puntuaciones de una persona particular o optar a otras funciones dentro de la aplicación web.

La página web al estar construida con Angular es completamente modular por lo que se podrían añadir funciones fácilmente y nos abre la posibilidad de construir un intermediario con Spring fácilmente, un *back-end* que actúe de mediador entre el *front-end* y los servicios de Firebase.

7 Agradecimientos

A Irene y Víctor por apoyarnos durante estos meses y ser *beta testers* excelentes, aportando a este proyecto una visión externa, creatividad y *feedback*.

A Pablo por ser un compañero ejemplar y una fuente de ideas muy sensatas.

A Ramón y Teresa por darnos la oportunidad de realizar este proyecto.

Bibliografía

[1] Raph Koster. Theory of Fun for Game Design. 2 diciembre 2013.

[2] Video Games in Education
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.543.5729&rep=rep1&type=pdf

[3] Minecraft as a Creative Tool: A Case Study
https://www.igi-global.com/article/minecraft-as-a-creative-tool/116516

[4] Tracy Fullerton. Game Design Workshop: A Playcentric Approach to Creating Innovative Games, Fourth Edition. 22 agosto 2018.

[5] Tutoriales de Unity oficiales
https://learn.unity.com/

[6] Documentación de Unity 3D
https://docs.unity3d.com/Manual/UnityManual.html

[7] Documento Básico Seguridad en caso de incendio
https://www.codigotecnico.org/pdf/Documentos/SI/DBSI.pdf

[8] Documentación de Firebase
https://firebase.google.com/docs?authuser=0

[9] Documentación de Angular
https://angular.io/docs

Anexo 1. GDD

GAME DESIGN DOCUMENT

D.A.N.G.E.R

Adrián Sánchez Lavarias

Pablo Querol Ballester

SECCIÓN I: Visión General Del Proyecto

1. **Nombre del juego**

   D.A.N.G.E.R - Disaster Prevention

2. **Personal de equipo**

   A continuación, definiremos el personal al completo que forma parte del videojuego Danger, con sus respectivos correos para contactar con ellos.

3. **Desarrolladores:**

   - Adrián Sánchez Lavarias (email: adsanla@inf.upv.es)
   - Pablo Querol Ballester (email: pabqueba@inf.upv.es)

4. **Resumen ejecutivo**

   1. **Concepto general**

      Danger es un juego con propósito pedagógico con el que aprender normativas y procedimientos aplicables a la prevención de incendios. Formado por 2 modos de juego, en el primero podrás construir salas y ponerlas a prueba para comprobar si cumplen con la normativa. Mientras que en el segundo podrás ponerte en la piel de las personas que se encuentran en un incendio y guiarlas para que logren escapar antes de que les alcance el fuego.

   2. **El Gancho**

      En Danger podrás desatar tu imaginación creando distintas salas o desarrollar tus propias estrategias para salvar la vida de las personas que han quedado atrapadas.

   3. **Género y extensión del juego**

      Género: Construcción y estrategia en tiempo real (RTS)

      Extensión: Dispone de 2 modos de juego.

   4. **Estilo visual**

3D en 3ª persona.

5. **Motor**
   Unity y Visual Studio

5. Núcleo del Gameplay

Mezclará los géneros de construcción, puzzle y estrategia en tiempo real en tercera persona, enfatizando en la toma de decisiones a lo largo del videojuego. Las plataformas principales son PC y dispositivos móviles.

Es un juego pensado para un solo jugador, sin distintos niveles de dificultad. Enfocado en la temática principal de incendios, puede ser rejugado para explorar diferentes situaciones.

6. Características de juego

1. Innovaciones del Gameplay

El objetivo es hacer un juego interactivo en 3D con el fin de enseñar la normativa y metodología en caso de incendio. Para ello se han creado 2 modos de juego de los géneros construcción con puzzle y estrategia en tiempo real. Lo que permite desarrollar la creatividad del usuario y colocarlo en una simulación de una situación real.

2. Otras Características que hará este juego mejor que a otros del mercado

El factor diferenciador del juego es la dualidad de modos de juego que desarrollan la creatividad, la capacidad de estrategia y reacción del jugador. Ambos modos de juegos, además, fomentan la rejugabilidad del título.

7. Alcance de proyecto

1. **Descripción de niveles**

    1. ***Creación***

        En el modo de juego “build” encontramos un nivel con un solo bloque base desde donde el jugador comenzará a crear el conjunto de salas deseado. Estás salas tendrán unos requisitos dependiendo del tamaño total de las mismas que deben cumplirse para finalizar el nivel.

    2. ***Estrategia***

        Los niveles del modo “escape” se generan de forma procedural, por lo que cada vez que se ejecuta el juego es diferente. Sin embargo, todos comparten que son salas conexas entre sí con forma rectangular pobladas con mobiliario de oficina.

8. **Público objetivo**

    Danger está dirigido a todo tipo de público (PEGI 7). Esto es debido a que el jugador deberá afrontar tomas de decisiones en un entorno controlado y creativo donde aprender.

9. **Plataformas**

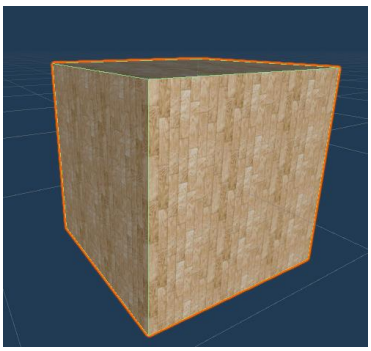
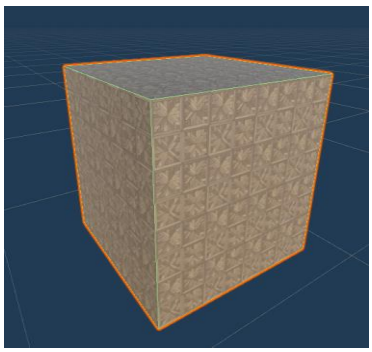
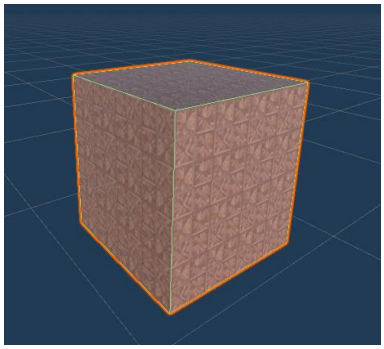
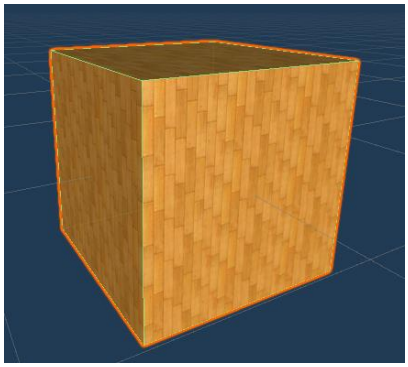
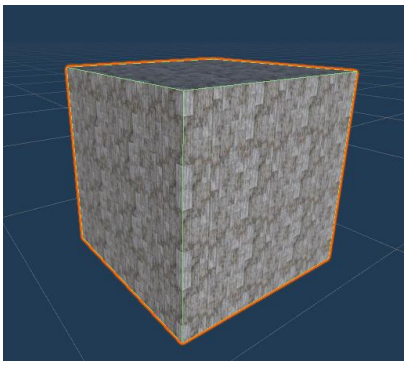
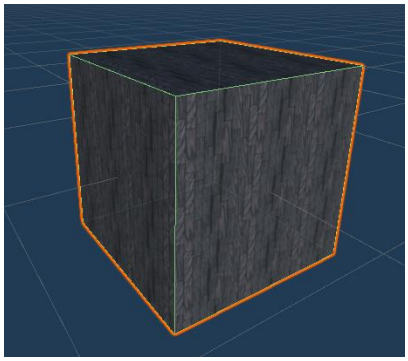
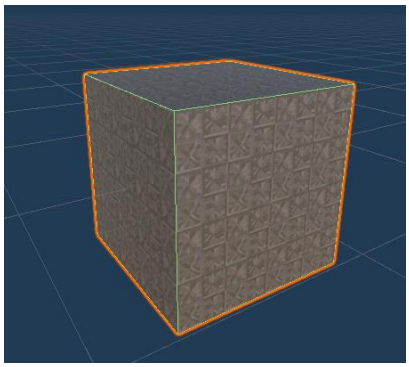
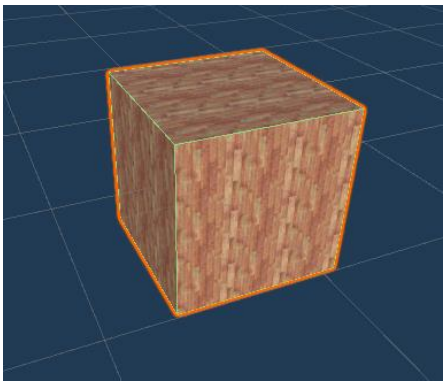
    PC

SECCIÓN II: Assets

1. **Modo construcción**

A continuación, se detallarán los distintos objectos o assets utilizados para el modo construcción a lo largo del proyecto:

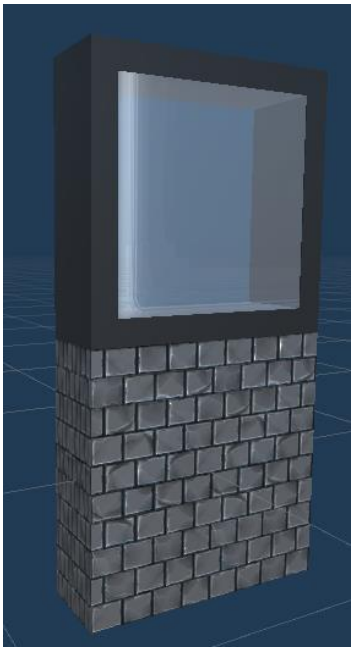
- ***Bloques:***

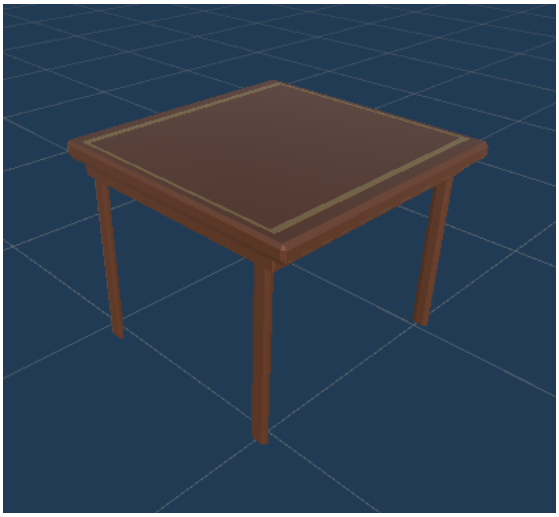
- ***Puerta:***

- ***Muro:***

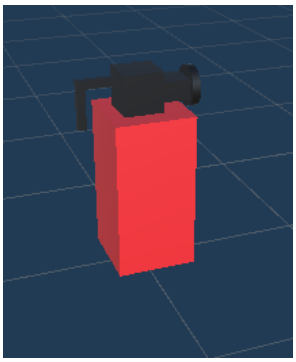
- ***Ventana:***



- ***Mesa:***



- ***Extintor:***

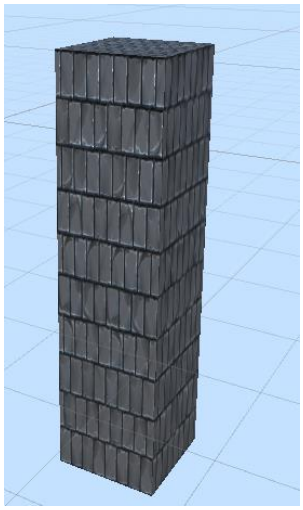


2. Modo estrategia (escape)

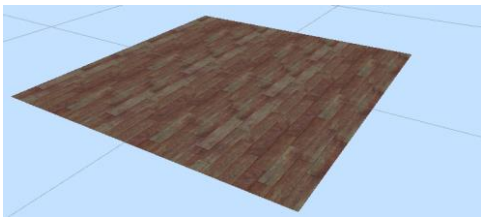
A continuación, se detallarán los distintos objetos o *assets* utilizados para el modo de estrategia (escape) a lo largo del proyecto:

- ***NPC:***

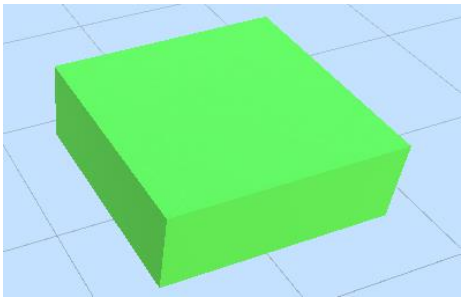
- ***Pared:***



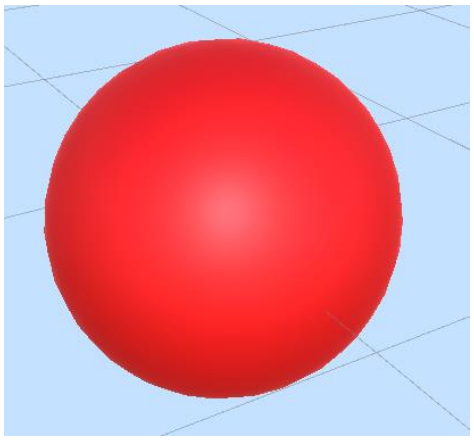
- ***Suelo:***



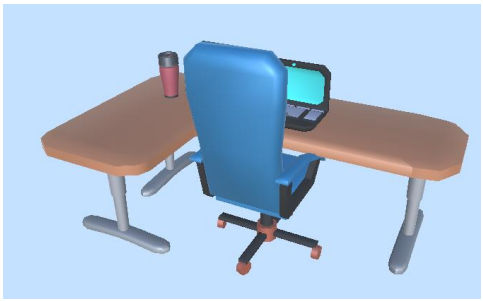
- ***Salida:***



- ***Localizador jugador:***

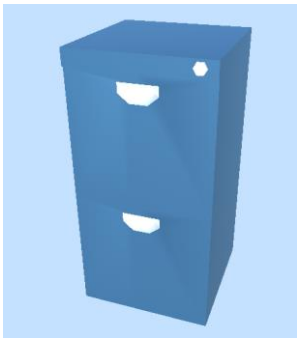


- ***Mesas:***



- ***Papelera:***

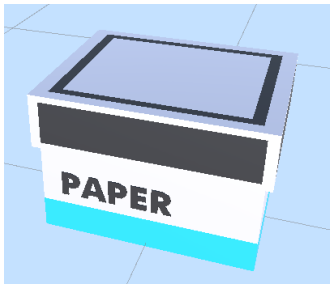
- ***Armario:***



- ***Carro de limpieza:***

- ***Extintor:***

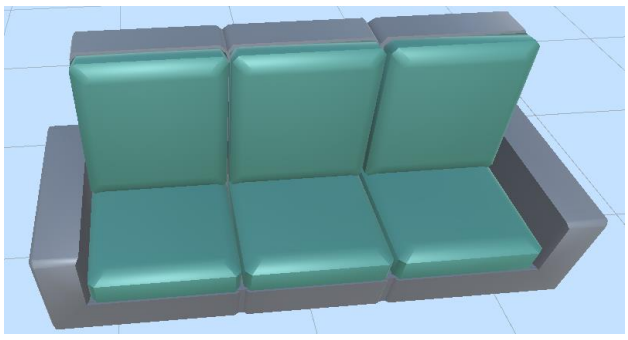
- ***Caja de papeles:***



- ***Planta:***

- ***Máquina de agua:***

- ***Sofá:***



SECCIÓN III: Controles

En este apartado se tratará de profundizar en detalle en todo lo referido a los controles que usaremos, para así poder realizar las distintas acciones que estarán definidas dentro del videojuego.

Tenemos la intención de lanzar nuestro videojuego en PC y dispositivos móviles, sin embargo los controles de la versión de móvil no están implementados todavía.

Órdenes de Ratón/Teclado de PC

A continuación definiremos los controles básicos que manejaremos a lo largo del videojuego. Algunas de las siguientes configuraciones, como son por ejemplo el caso del desplazamiento de la cámara y los personajes.

Teclas por defecto para el modo de juego construcción (Build)

- Desplazamiento de la cámara: Control Izq + Click Izquierdo.
- Rotación de la cámara: Control Izq + Click Derecho.
- Zoom de la cámara: Rueda del ratón.
- Colocar un objeto: Click Derecho.
- Eliminar un objeto: Click Izquierdo.

Teclas por defecto para el modo de juego de estrategia (Escape)

- Desplazamiento de la cámara: teclas WASD
- Seleccionar personaje: Click izquierdo .
- Seleccionar destino:Click izquierdo .
- Aumentar velocidad del movimiento: Shift .
- Abrir/cerrar mapa: tecla M

Menú de Acceso

- Acceso al menú: Escape.

SECCIÓN IV: Interfaz

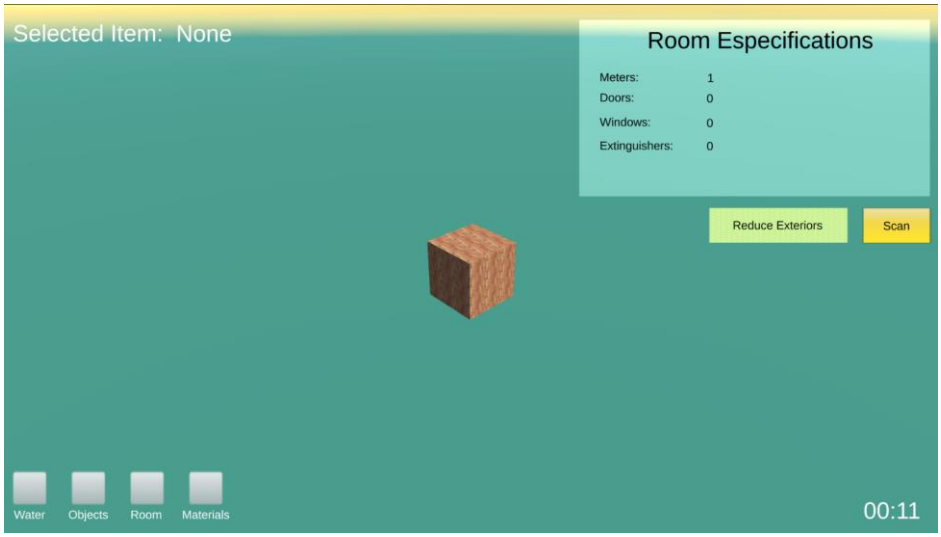
En esta sección concretamos la interfaz del jugador en detalle, especificaremos los distintos menús y cómo moverse entre ellos. Dentro de esta parte encontraremos el menú principal con sus distintas opciones, el HUD y el inventario.

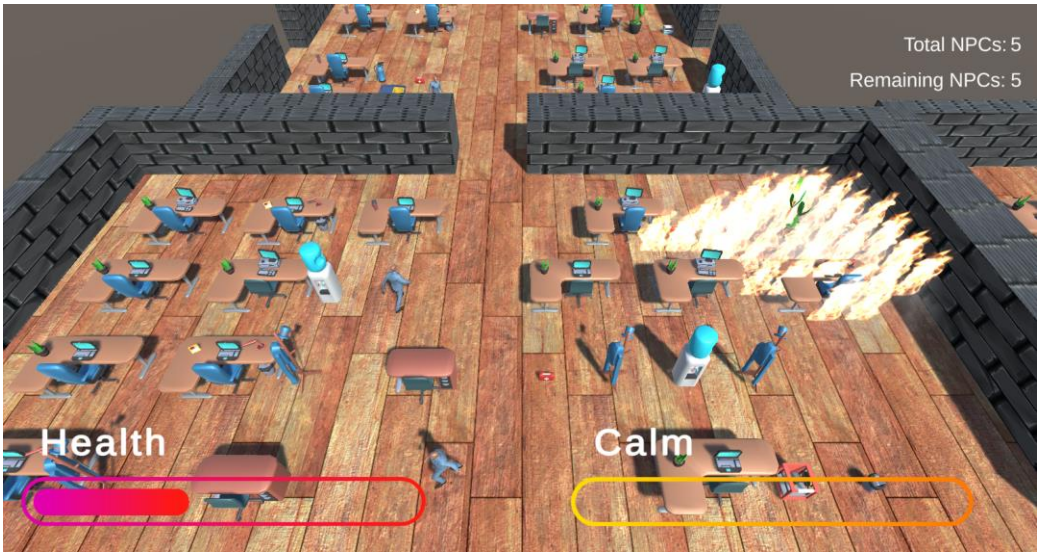
Cámara

Respecto a la cámara tendremos una vista en 3ª persona controlable por el jugador, de manera que el la sitúe en la dirección y ángulo que quiera en cada momento.

Vista estándar

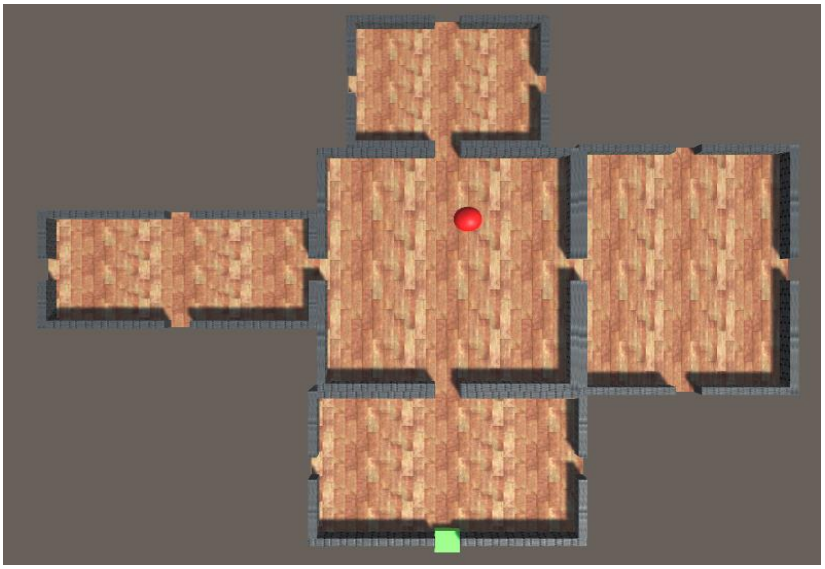
De manera estándar la cámara se situará en una posición concreta.





Vistas Alternativas

En el modo de estrategia (Escape) disponemos de una vista alternativa en forma de mapa, desde esta vista es la única donde pueden verse la posición actual del jugador en el escenario (punto rojo) y la salida del edificio donde tienen que llevar los personajes para ganar (cuadrado verde)



Opciones controlables por el Jugador

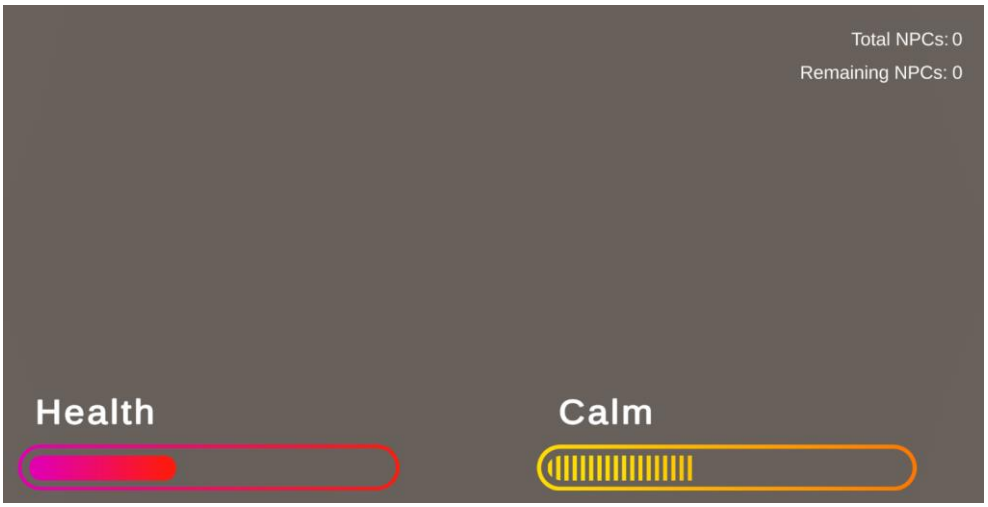
Utilizando el ratón el jugador podrá mover la cámara a voluntad.

HUD

Vista del jugador

- Modo de construcción

- Modo de estrategia



Información de estado
En la vista del jugador podemos observar los distintos estados e información que le damos al jugador:

- ***Modo escape***

  - Salud

Visible en la esquina inferior izquierda del HUD y administrada mediante una barra de color gradiente entre rosa y rojo. Cuando el fuego te alcanza al personaje el tamaño de la barra disminuye. Si te quedas sin salud pierdes el juego.

  - ***Estrés***

Visible en la esquina inferior izquierda del HUD y administrada mediante una barra de color gradiente entre amarillo y naranja. Cuando el personaje detecta fuego o humo este nivel aumenta. Cuando el personaje lleva un tiempo sin ver elementos peligrosos comienza a disminuir gradualmente. Cuanto más alto es este nivel mayor será la velocidad del personaje y menor será el tiempo de espera del personaje antes de pasar al modo autónomo. También aparecerá un texto sobre la barra que se intercambiará entre: “Calm”, “Stress” y “Pánic” según vaya aumentando o disminuyendo el nivel de estrés.

  - NPCs totales y restantes

Contadores visibles en la esquina superior derecha con el número de NPCs restantes y totales.

- Modo build

  - Selected Item

En la esquina superior izquierda encontraremos este elemento que nos indicará qué objeto concreto tiene el jugador seleccionado en cada momento.

○ Room Especifications

Este elemento se encuentra en la esquina superior derecha, se compone por un panel con la información relativa a todas las salas creadas por el jugador en la escena.

○ Item selectors

Conjunto de botones donde los diferentes objetos se clasifican, al pulsar estos botones abrimos sus páginas correspondientes para poder seleccionar el objeto deseado.

Menús

- **Menú principal**

- **Menús de opciones**

SECCIÓN V: Planificación

Planificación			
Tareas	**Encargado**	**Comienzo**	**Final**
Desarrollo			
Diseño	Pablo y Adrián	01/06/21	06/06/21
Concepto	Pablo y Adrián	01/06/21	03/06/21
Level Mechanics	Pablo y Adrián	03/06/21	06/06/21
Art	Pablo y Adrián	06/06/21	20/08/21
Modo estrategia	Pablo	06/06/21	20/08/21
Modo construcción	Adrián	06/06/21	16/08/21
UI	Adrián	06/06/21	20/08/21
Implementación	Pablo y Adrián	08/06/21	31/08/21
Modo estrategia	Pablo	08/06/21	31/08/21
Modo construcción	Adrián	08/06/21	31/06/21
Audio	Pablo y Adrián	01/08/21	31/08/21
Sound Design	Pablo y Adrián	01/08/21	31/08/21
MVP	Pablo y Adrián	15/08/21	25/08/21
Testing			
Test Plan	Pablo y Adrián	29/08/21	29/09/21
Beta Testing	Pablo y Adrián	29/08/21	03/09/21
Fase de lanzamiento			
Listo para su uso	Pablo y Adrián	04/09/21	04/09/21

Anexo 2. Recursos

Asset Store:

- Polygon dining room
  https://assetstore.unity.com/packages/3d/props/interior/polygon-dining-room-199435

- Stylized Bricks Materials
  https://assetstore.unity.com/packages/2d/textures-materials/stylized-bricks-materials-184484

- Minimalist Cartoon UI Pack
  https://assetstore.unity.com/packages/2d/gui/icons/minimalist-cartoon-ui-pack-123705

- Texture Glass Transparent Window
  https://assetstore.unity.com/packages/2d/textures-materials/texture-glass-transparent-window-182052

- Classic Interior Door Pack 1
  https://assetstore.unity.com/packages/3d/props/interior/classic-interior-door-pack-1-118744

- Wood Pattern Material
  https://assetstore.unity.com/packages/2d/textures-materials/wood/wood-pattern-material-170794

- Floor materials pack v.1
  https://assetstore.unity.com/packages/2d/textures-materials/floors/floor-materials-pack-v-1-140435

- Yughues Free Wooden Floor Materials
  https://assetstore.unity.com/packages/2d/textures-materials/wood/yughues-free-wooden-floor-materials-13213

Audio:

- Angry bull – Dark Fantasy Studio
  http://darkfantasystudio.com/

- Treasure – Dark Fantasy Studio
  http://darkfantasystudio.com/

Fuentes:

- GoodDog Plain
  https://www.1001fonts.com/gooddog-plain-font.html

- Barkentina 1
  https://www.dafont.com/es/barkentina-1.font

Multimedia:

- New York city day - live wallpaper
  https://www.youtube.com/watch?v=rOw-DNPpYuQ