UNIVERSITAT DE BARCELONA

Trabajo de final de grado

GRADO EN INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

Restaurant code: Juego en Realidad Virtual para aprender a programar

Autor: Alex Fuentes Raventós

Director:	**Dra. Inmaculada Rodríguez Santiago**
Realizado en:	**Departamento de Matemáticas y Informática**

Barcelona, **20 de junio de 2021**

Abstract (English)

This project focuses on the creation of a Virtual Reality game that helps those who want to get started in the computer world to learn how to program.

Specifically, it is about improving and expanding the Cooking Code game, developed in a previous Degree Final Project. During our project, the already created scenario has been modified and it has been fixed existing errors but we have focused on creating a new mode, so as not only to be able to learn to read computer language but also to write it. In this new mode, the player will be a waiter, where they must go to the different tables of the restaurant collecting the orders. The main difficulty will be to translate these orders that you receive in natural language into a language similar to pseudo-code.

Abstract (Català)

Aquest projecte es focalitza en la creació d'un videojoc en Realitat Virtual que ajudi els principis de la programació a aquells que es vulguin introduir en el món de la informàtica.

En particular, es tracta de millorar i ampliar el joc de Cooking Code, desenvolupat en un Treball de FinaL de Grau anterior, en el qual s'ha modificat l'escenari ja creat, arreglat errors existents i incorporat un nou mode de joc per no solament poder aprender a llegir llenguatge informàtic, sinó també escriure'l. En aquest nou mode joc, el jugadro serà un cambrer, on haurà d'anar a les diferents taules del restaurant recollint les comandes. La principal dificultat del joc serà traduir les comandes que reps en llenguatge natural a un llenguatge semblant al pseudocodi.

Abstract (Castellano)

Este proyecto se focaliza en la creación de un juego en Realidad Virtual que ayude a aprender a programar a los que quieren inicializarse en el mundo de la informática.

En particular se trata de mejorar y ampliar el juego de Cooking Code, desarrollado en un Trabajo de Final de Grado anterior, en el cual se ha modificado el escenario ya creado, arreglado errores ya existentes y se ha incorporado un nuevo modo para no solamente poder aprender a leer el lenguaje informático, sino también a escribirlo. En este nuevo modo, el jugador será un camarero, donde deberá ir a las diferentes mesas del restaurante recogiendo los pedidos y la principal dificultad será traducir estos pedidos que recibes en lenguaje natural a un lenguaje parecido al pseudocódigo.

Índice

Índice de figuras

1. Introducción

La gran vocación por la Realidad Virtual y el interés por introducir nuevas metodologías en la enseñanza han sido las dos grandes razones de la realización de este Trabajo de Final de Grado.

El tipo de juego que se ha desarrollado es lo que se llama Serious Game: juegos que tienen como objetivo principal la formación, pero teniendo el entretenimiento como componente fundamental. Son juegos que se utilizan en ámbitos como la publicidad, la educación, la formación o la simulación. Es una nueva metodología que permite aprender mientras te diviertes.

Durante este proyecto, se ha utilizado la Realidad Virtual para desarrollar un juego que facilite a la gente introducirse en el mundo de la programación. Es una plataforma muy novedosa e ideal para utilizar en este trabajo ya que la inmersión a otro mundo y la interacción con objetos que proporciona al jugador forman una muy buena combinación con los Serious Games. Para la creación de este juego se han utilizado las Oculus Quest, de las que hablaremos posteriormente (ver sección 9, apartado Oculus Quest).

En este trabajo se propone el diseño y la implementación de un nuevo modo de juego en un Trabajo de Final de Grado creado por Iván Gómez, llamado Cooking Code (añadir link), centrado para niños de 12 a 16 años en el que tenías que leer pseudcódigo para así poder crear hamburguesas. Este nuevo modo de juego permitirá no solamente leer, sino también escribir el pseudcódigo y así dar un paso más para el entendimiento de la lógica de la programación.

En este nuevo juego, el jugador será un camarero que tendrá que recorrer el restaurante para así recoger los pedidos de los comensales. El problema que se encontrará este camarero es que el cocinero que prepara las hamburguesas sólo entiende pseudcódigo y por lo tanto, tendrá que traducir el lenguaje natural del pedido recibido a pseudcógio.

El juego está dividido en jornadas laborales o días, durante el cual el camarero irá recogiendo pedidos y los traducirá para así poderlos enviar a cocina. Cuanto más rápido traduzca el pedido a pseudcódigo, más puntos de experiencia ganará a la hora de entregarlo, los cuales serán importantes para desbloquear nuevos niveles al acabar el día.

Para poder realizar una correcta implementación del trabajo, he tenido que aplicar conocimientos adquiridos en diferentes asignaturas. *Algorítmica* y *Algorítmica Avanzada* me han ayudado mucho a la hora de implementar los algorítmos más importantes del proyecto y los patrones de diseño aprendidos en *Diseño de Software* han sido muy útiles para la creación del código con una buena estructura. Para la correcta

utilización de una base de datos he utilizado he aprovechado mi aprendizaje en *Ingeniería del Software* y *Porgramación* me ha proporcionado una experiencia previa de la creación de un juego. Finalmente, *Factores Humanos* ha sido clave para la etapa de diseño.

2. Objetivos

2.1. Objetivo general

En este Trabajo de Final de Grado hay dos objetivos definidos: El primer objetivo principal del proyecto es ampliar el juego de Cooking Code creado en un Trabajo de Final de Grado anterior[4], con la creación de un nuevo modo de juego que permita aprender a la gente a escribir e iniciarse en el lenguaje utilizado para la creación de cualquier aplicación y programa informático. Por lo tanto, jugando a Restaurant Code se pretende que el usuario obtenga conocimientos sobre el uso de las instrucciones básicas: secuenciales, condicionales e iterativas.

2.2. Objetivos específicos

Los objetivos principal se desglosa en los siguientes objetivos específicos:

- Realizar un análisis de la competencia: Permitirá saber desde que punto se parte con este proyecto, aquello que están ofreciendo los juegos actuales y aquello que está funcionando.
- Realizar un análisis del proyecto anterior: Se buscarán errores y posibles mejoras al juego anterior, para acabar de definir la base de la que se partirá para crear el nuevo modo de juego.
- Diseño y desarrollo del nuevo modo de juego: Parte fundamental del proceso, la incorporación de un nuevo modo de juego en el que disfrutes jugando, permitiendo aprender mientras te diviertes y en el que el jugador pueda ir subiendo de nivel.
- Diseño y desarrollo de un tutorial para el nuevo modo de juego: Se explicará al usuario las interacciones y el movimiento del jugador antes de empezar.
- Guardar información durante el juego. Para poder mejorar el juego en un futuro, se guardará información de las acciones que realiza el jugador y el tiempo que tarda en ejecutarlas.

3. Proyectos relacionados

Los proyectos que se muestran a continuación son los que han parecido más destacables de todos los que se han probado antes de diseñar y programar Restaurant Code. Se han extraído cosas muy interesantes de todos y cada uno de ellos, ya fuese por sus dinámicas, jugabilidad, o interfaces.

3.1. Basados en Realidad Virtual

Cooking Code

Cooking Code es un proyecto de Realidad Virtual desarrollado por Iván Gómez, centrado en conseguir un juego del estilo Serious Games, juegos en los que el principal objetivo es aprender algo de forma entretenida. El juego está ambientado en una hamburguesería y se basa en convertir al jugador en un cocinero, al cual no le llegan los pedidos de los clientes en lenguaje natural, sino en un lenguaje parecido al pseudocódigo. Añadiendo este hándicap a la hora de hacer las hamburguesas, se consigue que el usuario se dedique a aprender las bases de la lectura del lenguaje informático mientras se entretiene jugando. Como está centrado en el ámbito educacional, nos encontramos con una escena muy sencilla, donde el gran foco está en el aprendizaje de los jugadores y los jugadores no pueden moverse de detrás de la barra. Los jugadores consiguen diferentes logros con los cuales ir subiendo de nivel para así poder realizar pedidos más complejos y por lo tanto el juego consigue tener una curva de aprendizaje muy gradual.

Este es el juego que se ha continuado a lo largo de este proyecto.

Figura 1: Cooking Code (Fuente: propia)

Job Simulator

Job Simulator: The 2050 Archives es un juego de Realidad Virtual, desarrollado y publicado por Owlchemy Labs, donde acercan al jugador a cuatro trabajos del mundo real (oficinista, mecánico, chef y cajero) de una manera cómica. Esta parte cómica del juego se encuentra en que se estará rodeado en todo momento de robots que dirán concretamente lo que se tiene que hacer y durante esta tarea se puede hacer todo lo que se quiera, desde comer rosquillas mientras se trabaja, a fotocopiarse la mano poniéndola encima de la impresora.

Job Simulator es el modelo de referencia como juego de Realidad Virtual. Un juego donde no solamente

se busca conseguir el objetivo, sino donde el entretenimiento siempre estará presente. Además, su diseño intuitivo facilita a los usuarios al entendimiento de los trabajos a realizar, haciendo mucho más dinámico el juego desde un principio.

Figura 2: Oficina de Job Simulator (Fuente: https://www.roadtovr.com/)

Half-Life Alyx

Half-Life Alyx es un videojuego de disparos en primera persona desarrollado por Valve Corporation. Es la precuela de la serie de juegos de Half-Life desarrollada entre 1998 y 2007. Considerada por la mayoría de los críticos el mejor juego de Realidad Virtual conseguido hasta la fecha[5], este juego lo ponemos como modelo y como ejemplo por su excelencia. Desde la utilización de las interfaces de usuario hasta los gráficos, pasando por la jugabilidad y la historia, el juego está pulido en todos sus aspectos. Probando este juego se ve la capacidad, potencia y alcance que tiene la Realidad Virtual para contar historias. En todo momento se es el protagonista de esta historia, no sólo porque esta se basa en que el protagonista ha de salvar el mundo, sino por la posibilidad de hacer acciones entre batalla y batalla como jugar con los elementos del escenario y sus físicas.

Me gustaría decir que se ha utilizado este juego como referencia y como modelo, pero realizar lo que ha conseguido el equipo de Valve es algo inalcanzable en un Trabajo de Final de Grado. Este juego sirvió para valorar la Realidad Virtual y descubrir lo espectacular que puede llegar a ser un juego de Realidad Virtual.

Figura 3: Imagen de la vista del jugador en Half Life Alyx (Fuente: https://www.half-life.com/es/alyx/)

3.2. No basados en Realidad Virtual

Rabbids Coding

Rabbids Coding es un juego para el PC y móvil creado por Ubisoft, que utiliza los Rabbids[1] para enseñar las bases de la lógica de la programación.

El jugador tiene como objetivo limpiar una nave espacial invadida por Rabbids dando instrucciones básicas a un Rabbid con un dispositivo de control mental donde el objetivo será conseguir limpiar la nave con el menor número de instrucciones y dónde iremos desbloqueando nuevas instrucciones a medida que avancemos de nivel. Las instrucciones son muy básicos y las hemos de arrastrar desde un menú y ponerlas en orden.

El juego está pensado para que cualquier persona sin ningún tipo de conocimiento previo en programación pueda divertirse mientras tienen una experiencia educacional, ofreciéndo herramientas para poder disfrutar aprendiendo con las bases de la programación.

En Rabbids Code se enseña utilizando simplemente fichas de puzzle con acciones que se deberán unir para realizar la acción completa. Es un nivel más básico de programación del que se ha utilizado en Restaurant code, donde no sólo se utilizan acciones sino también palabras para poder traducir el lenguaje natural. Este juego muestra lo interesante y divertido que puede ser aprender las bases de la programación si esta es enseñada de una manera entretenida y entendible.

[1]Rabbids: Son personajes ficticios, una especie de conejos antropomórficos.

Figura 4: Pantalla de un nivel de Rabbids Coding en Android (Fuente: https://www.trucosapple.com/)

Scratch

Scratch es un lenguaje de programación visual desarrollado por el Grupo Lifelong Kindergarten del MIT Media Lab. Está centrado en facilitar el aprendizaje, intentando que este sea lo más intuitivo posible. Se ha de tener en cuenta que Scratch no sólo sirve para la creación de programas, sino que con Scratch se puede llegar a crear mini videojuegos o incluso historias animadas, motivando así a hacer tus propias creaciones.

Respecto a su interfaces gráfica, se puede ver un panel lateral donde se encuentran piezas que pueden ir encajando. Estas piezas servirán para crear un puzzle que permitirá desarrollar diferentes secuencias que en conjunto crearan el programa.

Esta aplicación ha inspirado para crear la mecánica de juego en Restaurant Code, donde se tienen piezas con instrucciones que deberán encajar entre ellas para así poder realizar los pedidos que se ordenen hacer.

Figura 5: Ejemplo de un proyecto en Scratch (Fuente:https://ladiversiva.com/)

4. Análisis del proyecto

4.1. Requisitos funcionales

- UC1: El usuario se registra en el sistema.
- UC2: El usuario inicia sesión, recuperando su progreso anterior en el caso de haber jugado antriormente.
- UC3: El usuario escoge el modo de juego.
- UC4: El usuario recibe el pedido.
- UC5: El usuario realiza el pedido.
- UC6: El usuario comprueba si el pedido está bien realizado.
- UC7: El usuario recibe feedback del pedido.
- UC8: El usuario recibe feedback al terminar el día.
- UC9: El usuario pausa la partida
- UC10: El usuario visualiza la lista de logros.
- UC11: El usuario guarda la partida.
- UC12: El usuario sale del juego.
- UC13: El usuario modifica el volumen.

4.2. Casos de uso

En esta sección se muestra el diagrama de casos de uso, donde se ven las actividades que deberá realizar el usuario para llevar a cabo algún proceso y se describirá la acción de cada caso de uso.

Por cada caso de uso que se puede ver en el diagrama, hay una tabla que profundiza en el caso de uso, donde se ve de cada uno:

- Actor
- Descripcion
- Precondiciones
- Secuencia Principal
- Flujo Alternativo
- Postcondiciones

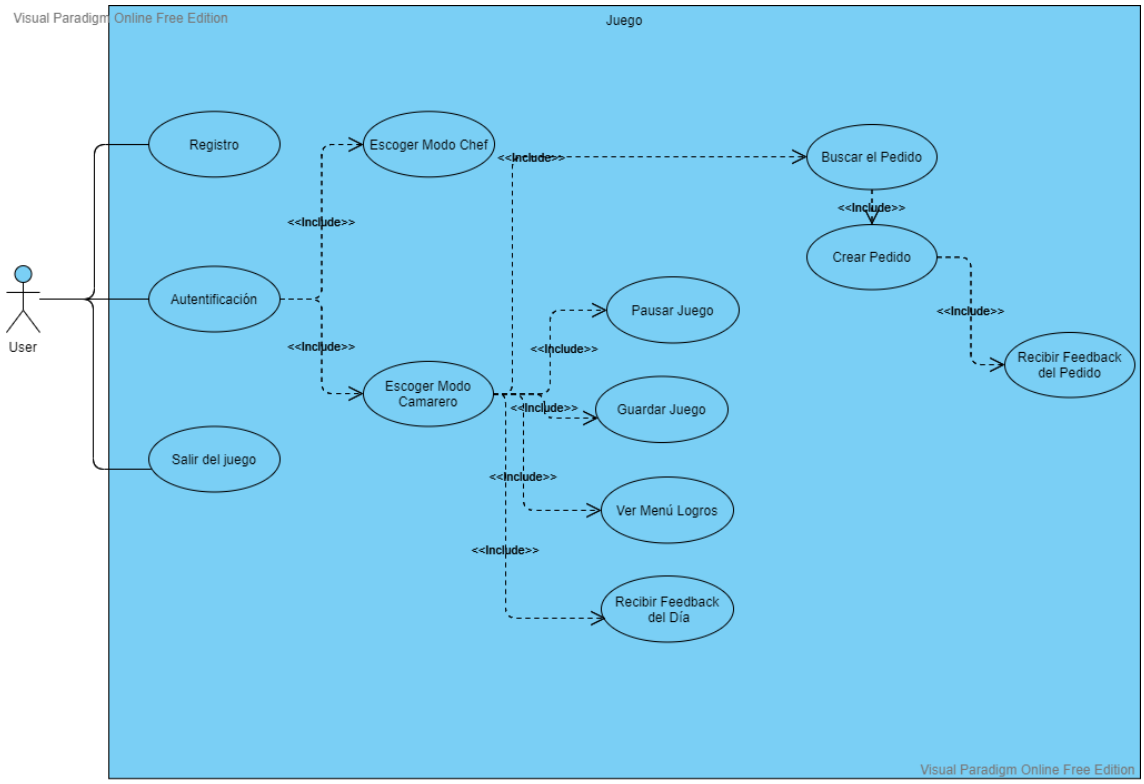

Figura 6: Diagrama de Casos de Uso (Fuente: Propia)

Name	UC-1 Registro en el sistema
Actor	Usuario
Descripción	El usuario quiere registrarse en el sistema, tiene que introducir usuario y contraseña
Precondiciones	No hay
Secuencia principal	1. El usuario entra en el juego 2. Selecciona la opción de registrarse. 3. El juego muestra un formulario para registrarse. 4. El usuario introduce nombre de usuario y contraseña. 5. El sistema comprueba que no exista ese nombre de usuario. 6. El sistema registra el usuario en la base de datos.
Flujo Alternativo	No hay
Postcondiciones	El juego inicia con el usuario ya registrado

Nombre	UC-2 Inicio de sesión
Actor	Usuario
Descripción	El usuario quiere iniciar sesión e introduce usuario y contraseña
Precondiciones	El usuario ha de estar registrado en el sistema
Secuencia principal	1. El usuario entra en el juego 2. Selecciona la opción de iniciar sesión. 3. El sistema muestra un formulario para iniciar sesión. 4. El usuario introduce nombre de usuario y contraseña. 5. El sistema comprueba que exista ese usuario en la base de datos.
Flujo alternativo	5. *a)* Si el usuario no existe o la contraseña es errónea, el sistema muestra un mensaje y vuelve a enseñar el menú de inicio de sesión.
Postcondiciones	El juego se inicia con el usuario introducido

Name	UC-3 Salir del juego
Actor	Usuario
Descripción	El usuario desea salir del juego
Precondiciones	No hay
Secuencia principal	1. El usuario pausa el juego 2. El sistema muestra el menú de pausa. 3. El usuario selecciona la opción de guardar y el sistema guarda el prograso 4. El sistema termina la partida.
Flujo alternativo	1. El usuario inicia la aplicación 2. El sistema muestra el menú principal 3. El usuario selecciona la opción de salir 4. El el sistema termina la partida
Postcondiciones	No hay

Name	UC-4 Escoger modo de juego
Actor	Usuario
Descripción	El usuario decide qué modo de juego quiere jugar
Precondiciones	El usuario ha iniciado sesión (UC2)
Secuencia principal	1. El usuario escoge el modo de juego camarero. 2. El sistema muestra el nivel.
Flujo Alternativo	No hay
Postcondiciones	El usuario puede jugar e ir subiendo niveles

Name	UC-5 Buscar el pedido
Actor	Usuario
Descripción	El usuario va a la mesa donde le reclaman para recoger el pedido
Precondiciones	El usuario ha escogido el modo de juego (UC4)
Secuencia principal	1. El usuario busca la mesa donde se le solicita. 2. El usuario va a la mesa. 3. El sistema muestra el pedido 4. El usuario recoge el pedido.
Flujo Alternativo	No hay
Postcondiciones	El usuario puede crear el pedido

Name	UC-6 Crear el pedido
Actor	Usuario
Descripción	El usuario crea el pedido solicitado
Precondiciones	El usuario ha escogido el modo de juego 'Waiter'(UC4)
Secuencia principal	1. El usuario va al espacio donde puede crear el pedido 2. El usuario utiliza los objetos a su alrededor para crear el pedido.
Flujo alternativo	2. *a*) Si el tiempo del día finaliza, se eliminan los paneles colocados y se visualiza el feedback.
Postcondiciones	El usuario puede entregar el pedido en el momento que crea que esté correcto

Name	UC-7 Indicar pedido realizado
Actor	Usuario
Descripción	El usuario indica que ya ha realizado el pedido
Precondiciones	El usuario ha escogido el modo de juego 'Waiter'(UC4)
Secuencia principal	1. El usuario pulsa el botón para comprobar el pedido 2. El sistema muestra por pantalla el feedback del pedido
Flujo alternativo	2. *a*) Si el pedido no esta bien realizado, volveremos a intentar crear el pedido
Postcondiciones	El usuario puede ir a la siguiente mesa a recoger un nuevo pedido

Name	UC-8 Pausa el juego
Actor	Usuario
Descripción	El usuario pausa el juego
Precindiciones	El usuario ha escogido el modo de juego (UC4)
Secuencia principal	1. El usuario pulsa el botón para pausar el juego . 2. El sistema muestra el menú de pausa.
Flujo Alternativo	No hay
Postcondiciones	El tiempo del día y del pedido se detiene hasta que decida el usuario

Name	UC-9 Mostrar menú de juego
Actor	Usuario
Descripción	El usuario ve la lista de logros.
Precindiciones	El usuario ha pausado el juego (UC8).
Secuencia principal	1. El usuario pulsa la opción del menú de pausa para visualizar los logros. 2. El sistema muestra los logros conseguidos.
Flujo Alternativo	No hay
Postcondiciones	El usuario visualiza el menú de logros hasta querer reanudar la partida

Name	UC-10 Recibir el feedback del día
Actor	Usuario
Descripción	El usuario ve el feedback del día y su evolución
Precindiciones	El día se ha acabado
Secuencia principal	1. El sistema detecta que el día se ha acabado. 2. El sistema finaliza el pedido actual. 3. El sistema muestra el feedback por pantalla que leerá el usuario.
Flujo Alternativo	No hay
Postcondiciones	El usuario visualiza el menú de logros hasta querer reanudar la partida.

4.3. Grupos de Usuarios

El grupo de usuarios al que va enfocado este juego es a estudiantes entre 12 y 18 años. Dentro de este rango de edad se puede encontrar dos subgrupos: aquellos que no han tenido contacto con el mundo de la programación y aquellos que ya tienen experiencia.

4.3.1. Subgrupo de usuarios 1: No iniciados en la programación

Tienen un conocimiento avanzado de tecnología, y eso junto a sus grandes dotes de adaptación, les permite controlar diferentes dispositivos y plataformas. Suelen tener inquietudes y pensamientos muy diversos, desde temas sociales hasta más recreativos, como series o películas ...siendo un grupo bastante heterogéneo. Comparten, por otra parte, una atracción muy grande a nuevas formas de entretenimiento, encontrándose abiertos a cualquier novedad.

No han tenido experiencias previas con la programación y nunca han mostrado interés por descubrir ese mundo, ya sea porque no saben que existe esa posibilidad o porque lo que se les ha enseñado no ha captado su atención.

4.3.2. Subgrupo de usuarios 2: Iniciados en la programación

Tienen un uso de la tecnología muy avanzado, conociendo y controlando el funcionamiento de los dispositivos que utilizan. A diferencia del otro grupo, nos encontramos con jóvenes que muestran mucho interés por el apartado tecnológico, con el mundo de los videojuegos y su desarrollo, siendo un grupo bastante más interesado en juego como Restaurant Code. Suelen estar bien informados sobre las nuevas tecnologías, conociendo ya seguramente las gafas de realidad virtual mas en profundidad.

Han tenido experiencias previas con la programación, ya sea por simple el simple conocimiento de su existencia o incluso por haber probado algunos ejercicios relacionados. Además, también es un grupo con mucha experiencia en el mundo de los videojuegos lo que significa una gran rapidez para adaptarse rápido a nuevas mecánicas

4.4. Personas

Cuando ya se tienen definidos los grupos de usuario, se crean personajes ficticios a partir de una descripción y una imagen. Las personas son arquetipos de usuarios que permiten ponernos en su piel y así tenerlo en cuenta a lo largo del proceso de creación del juego. A continuación se presentan dos personas, cada una de un subgrupo de usuarios.

Diego

Diego es un joven de 16 años que está cursando 1o de Bachillerato en el centro de Barcelona. Le gusta mucho leer durante su tiempo libre, tanto novelas en libros físicos, como noticias y libros de temas sociales en el móvil, tablet o ebook, dispositivos que domina con facilidad. Suele estar muy pendiente de los últimos portátiles ya que quiere que su nueva adquisición sea un portátil de gama media, que pese poco para poder llevarlo de un sitio a otro sin dificultad.

Su pasatiempo favorito es quedar con sus amigos y entrena dos días a la semana a baloncesto, lo que le permite distraerse después de largas horas de estudio. Su objetivo es estudiar Ciencias Ambientales, para así poder trabajar y ayudar en un futuro en el mundo de la sostenibilidad, tema por el que está muy pendiente.

Figura 7: Diego (Fuente: https://pixabay.com/es/photos)

Raquel

Raquel es una chica de 14 años que está cursando 3o de la eso en las afueras de Barcelona. Durante su tiempo libre le gusta buscar nuevos juegos cooperativos con los que distraerse con sus amigos. La mayoría de sus inquietudes están relacionadas con el mundo de la tecnología y además ha empezado a desarrollar un interés por el mundo de la programación, el cual está descubriendo a partir de la asignatura de Tecnología que está cursando en el colegio, donde ya está utilizando Scratch para hacer pequeños programas.

Su sueño es ser directiva de una gran empresa y por lo tanto tiene claro que quiere estudiar Ingeniería Informática cuando se gradúe, para así estar un paso más cerca de su objetivo.

Figura 8: Raquel (Fuente: https://www.freepik.com/free-photo)

5. Diseño del juego

En este apartado se introduce la idea principal del juego, la organización que se ha seguido durante el proceso de diseño y los objetos que se han utilizado a lo largo del juego y que se han creado para conseguir una experiencia única.

5.1. Idea Principal

La idea principal del juego corresponde con la que se introduce en el proyecto anterior: facilitar una nueva forma de aprender las bases de programación de una manera entretenida e innovadora. Además se ofrece a los usuarios la opción de guardar su progreso con la utilización de una base de datos, dándoles la posibilidad de que mejoren a su propio ritmo y ofreciéndoles motivaciones como el desbloqueo de diferentes tipos de logros.

El objetivo de el modo creado durante el proyecto, el modo "Waiter", será escribir en un lenguaje parecido al pseudcódigo los pedidos que se han ido a recoger a las mesas de los comensales y que se han recibido en lenguaje natural. Durante todo este proceso, el jugador deberá estar recorriendo el restaurante e interaccionar con objetos.

El jugador empieza por el nivel más básico, donde el pseudocódigo es relativamente sencillo y los ingredientes a utilizar son muy limitados. A medida que el usuario vaya aprendiendo, y por lo tanto mejorando, su nivel irá subiendo, lo que significará instrucciones más complicadas como condicionales y desbloqueo de ingredientes como la lechuga o el ketchup.

El jugador también irá recibiendo experiencia a medida que entregue pedidos realizados con éxito. Dependiendo del tiempo que tarde en crear un pedido recibirá más o menos experiencia. Si el jugador tarda menos de 1 minuto en realizar el pedido, recibirá 100 puntos de experiencia, en caso contrario, recibirá 25 puntos de experiencia.

En el juego hay cuatro niveles: básico, condicional simple, condicional compuesto e iterativo. Para subir de nivel el jugador tendrá que cumplir dos condiciones:

- La cantidad de pedidos que se hayan realizado de los diferentes tipos. Es decir, para desbloquear el segundo nivel, en el que podremos escribir instrucciones condicionales simples, has de haber realizado diez pedidos básicos. El desbloqueo de los siguientes niveles funciona de la misma manera. El desbloqueo del condicional compuesto sucederá con diez pedidos condicionales simples realizados y el último nivel, donde las instrucciones iterativas estarán desbloqueadas, lo podremos alcanzar al realizar diez pedidos condicionales compuestos.

- La cantidad de puntos de experiencia del jugador. Para poder subir el primer nivel el usuario ha de tener 325 puntos de experiencia. Para poder subir al siguiente nivel el jugador deberá tener 1500 puntos de experiencia y para el siguiente deberá tener 3000 puntos de experiencia. Finalmente, para alcanzar el último nivel, el jugador deberá tener un mínimo de 10.000 puntos de experiencia.

El jugador irá recibiendo pedidos a medida que recurra el día .La jornada de trabajo durará algo más durante el primer nivel ya que se ha de tener en cuenta no sólo la curva de aprendizaje del pseudcódigo, sino también la de las mecánicas del juego. Al acabar el día, el pedido actual desaparecerá y se mostrará un

feedback por pantalla donde podremos ver si hemos desbloqueado algún nivel o ingrediente y la cantidad de experiencia conseguida ese día.

Figura 9: Comienzo del tutorial del modo Waiter (Fuente: propia)

5.2. Game Design Canvas Framework

Realizar un juego tanto educativo como entretenido no es una tarea sencilla. Por eso se utilizará un *framework*[2] llamado Game Design Canvas (GDC) que nos ayudará a organizar el proceso de diseño y evaluar posibles puntos débiles de este diseño. Los elementos que encontramos son los siguientes:

- Planificación: El primer paso es ver de que recursos disponemos para el desarrollo del juego. Se ha de definir: el género, el presupuesto, el plazo para desarrollar el juego, el personal disponible, el target de nuestro juego y los dispositivos los cuales son necesarios para hacer funcionar el juego.
- Meta de diseño: Las emociones que se quieren conseguir cuando el jugador juega. En este caso es el entretenimiento y el interés por la programación.
- (1) Deseos/Necesidades: En este punto se recogen los deseos y necesidades que los jugadores pueden satisfacer cuando juegan. En este caso, el usuario desea introducirse al mundo de la programación y busca una manera distinta de aprender. Puede progresar de una manera dinámica y entretenida.
- (2) Objetivos: Encontramos tres tipos:
  - De corto alcance: Actividad cíclica central. Es el conjunto de acciones que el jugador realiza durante una sesión de juego. En este caso, traducir el lenguaje natural a pseudcódigo para así realizar el pedido correctamente.
  - De medio alcance: Elementos que los jugadores acumulan para preparar los objetivos de largo alcance. Por ejemplo, hacer los pedidos necesarios durante un día para así desbloquear un logro.

[2]Framework: Entorno de trabajo

- De largo alcance: El objetivo al cual se enfrenta el jugador, tras el cual el juego habrá acabado. En este caso, será desbloquear todos los logros.

- (3) Sistema de Progresión: El sistema requiere de mecanismos que hagan sentir al usuario que avanza y progresa. Los jugadores percibirán dicho avance a través de este sistema. Es uno de los mecanismos cruciales de fidelización. En este proyecto el progreso del jugador se contempla a través de las instrucciones e ingredientes que va desbloqueando cuando sube de nivel. También se ve dependiendo de la cantidad de logros que haya desbloqueado. El personaje no evoluciona, es la habilidad de la persona que lo controla la que irá mejorando.

- (4) Desafíos: Los obstáculos que experimentan los jugadores deben de estar siempre un poco por encima de su habilidad actual. El diseño debe incluir una curva de dificultad ascendente, gradual y que contenga momentos donde se pueda hacer patente el progreso realizado. En el juego la curva de dificultad reside en el desbloqueo de nuevas instrucciones e ingredientes y en el aumento de dificultad de los pedidos. Nunca se desbloqueará más de un ingrediente o un tipo de instrucción a la vez, para así hacer la curva de dificultad más gradual.

- (5) Sistema de recompensas: A través de este, los jugadores pueden visualizar su progreso. Los elementos del sistema suelen ser insignias, logros, títulos honoríficos o cualquier elemento coleccionable. En el modo de juego creado, el sistema de recompensas funciona a través de tres maneras distintas:

  - Los comentarios recibidos después de realizar un pedido bien hecho. Estas son recompensas momentáneas que animan al jugador a seguir mejorando.

  - Los puntos de experiencia recibidos al completar un pedido. Estos puntos de experiencia dependerán de la cantidad de tiempo dedicado para realizar el pedido.

  - Insignias que puede ver el jugador en todo momento en el menú de logros, que representan los logros desbloqueados.

- Narrativa: En este apartado se describe el contexto en el que sucede la experiencia diseñada. En este caso, un nuevo cocinero ha sido incorporado en el restaurante donde hemos trabajado desde siempre. Pero este cocinero tiene una característica única... Sólo entiende el pseudcódigo! Tendremos que aprender a traducir a pseudcódigo todos los pedidos recibidos por los comensales en lenguaje natural para que así nuestro cocinero sea capaz de hacerlos.

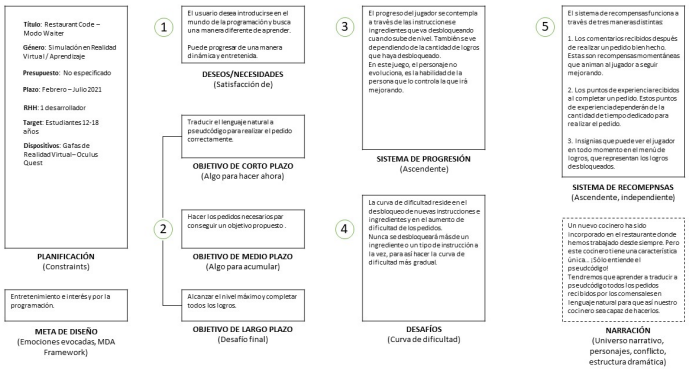

Figura 10: Game Design Canvas (Fuente: propia)

5.3. Objetos del juego

En este apartado se explica los diferentes objetos que componen el juego, su utilización y su funcionamiento.

5.3.1. Paneles

Los objetos presentados a continuación son los paneles utilizados para realizar los pedidos en el nuevo modo de juego creado. Estos paneles estarán colocados encima de dos mesas, boca arriba para que así sea fácil leer su contenido y estarán ordenados dependiendo del nivel en el que se desbloquean cada uno.

Paneles Básicos

Paneles que sólo representan un ingrediente (ver Figura 11). Encontramos cuatro paneles básicos de nivel 1: BottomBreadPanel, TopBreadPanel, CheesePanel y MeatPanel. A medida que se consiguen logros y subamos de nivel, conseguiremos dos nuevos paneles básicos: LettucePanel y KetchupPanel. Estos paneles están divididos en dos partes: Acción a realizar, la cual siempre será "PUT", e ingrediente a colocar: Carne, Lechuga, Queso...

(Esto en implementación –¿El panel básico se compone sólo por un Panel3D. Además este panel básico será de color verde, para así seguir con los colores utilizados en el otro modo de juego.)

Figura 11: Visión del editor de Unity de un panel básico. (Fuente: propia)

Panel Condicional

Paneles que como dice su título, expresan una sentencia de programación condicional. Indican si el ingrediente colocado en el interior de la condición se colocará en la hamburguesa. Hay dos tipos de paneles condicionales:

- Panel condicional simple. Sólo contiene una condición y si esta no se cumple simplemente se pasará al siguiente ingrediente.

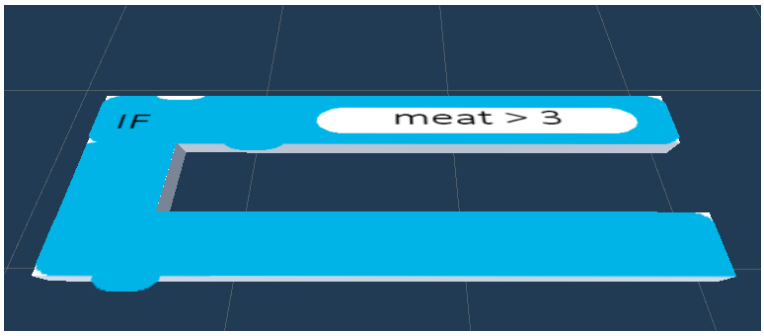

Figura 12: Visión del editor de Unity de un panel condicional simple (Fuente: propia)

- Panel condicional compuesto. El compuesto también contiene una condición, pero además contiene el ingrediente que se deberá añadir a la hamburguesa si esa condición no se cumple.

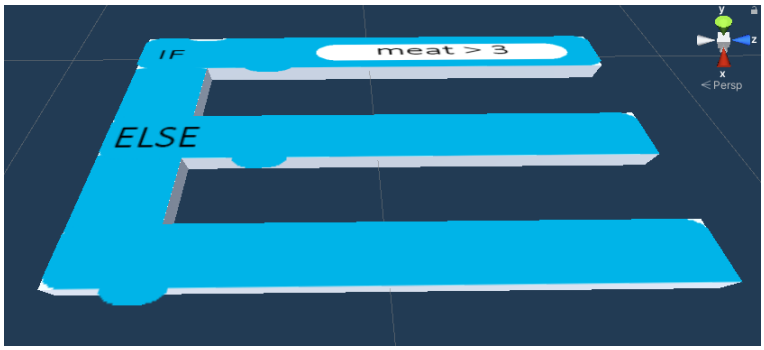

Figura 13: Visión del editor de Unity de un panel condicional compuesto (Fuente: propia)

Panel Iterativo

Cuando el jugador haya subido de nivel y haya aprendido a a utilizar todos los paneles anteriores, se desbloquearán los paneles iterativos. Estos paneles indican la cantidad de veces que se ha de poner un ingrediente.

Implementacion –¿El panel iterativo será exactamente igual al panel condicional básico, cambiando únicamente el IF por el FOR.

Figura 14: Visión del editor de Unity de un panel iterativo (Fuente: propia)

5.3.2. Pizarra

La pizarra será donde se colocarán los paneles, uno debajo del otro, para así mantener un orden vertical en las instrucciones que deberá seguir el cocinero para hacer la hamburguesa.

5.3.3. Botón de confirmación

Este botón se pulsará cuando se crea que se tienen todas las instrucciones colocadas en orden, para así confirmar el pedido creado.

5.3.4. Pantalla

Pantalla que mostrará el pedido que hay que realizar después de que se haya ido a buscar a la mesa. Esto nos facilita recordar en todo momento el pedido actual y así no tener que volver que preguntar a nuestro cliente. La pantalla también nos mostrará los diferentes feedbacks que necesitaremos a lo largo del día.

5.4. Desarrollo del juego y mecánicas

El jugador tendrá el rol de camarero en una hamburguesería y tendrá que enviar los pedidos recibidos a cocina. El juego se desarrolla en jornadas de trabajo durante la que iremos recibiendo pedidos y el objetivo del jugador deberá ser realizar la máxima cantidad de pedidos posibles a lo largo de la jornada.

Primero, el jugador tendrá que ir a la mesa donde es solicitado y preguntar por el pedido deseado. Este pedido es recibido en lenguaje natural por el jugador y lo tendrá que traducir a pseudocódigo antes de poder enviarlo a cocina. Cada pedido no tiene tiempo límite, sólo tiene límite la jornada de trabajo.

La interacción principal será agarrar y soltar. El jugador cogerá uno de los paneles y lo soltará al llegar a la pizarra, donde este se situará justo debajo del ingrediente puesto anteriormente. Finalmente el jugador pulsará el botón para comprobar que el pedido está bien escrito. Cabe la posibilidad de que el pedido no sea correcto y por lo tanto lo tendrá que volver a intentar.

5.4.1. Feedbacks

El jugador recibirá diferentes feebacks a los largo del juego. Estos feedbacks se clasifican dependiendo de si se reciben de manera escrita o no. Dar un buen feedback es muy importante ya que así el jugador puede aprender de su error y corregirlo en futuras ocasiones y también le pueden alentar y animar, haciendo más entretenida su experiencia.

Escritos

Hay dos momentos en los que el jugador recibe feedback escrito:

1. Cuando envía el pedido. En ese momento, sale por pantalla el feedback del pedido. Si el pedido no está bien realizado, se muestra por pantalla un mensaje que da pistas al jugador sobre lo que puede haber hecho mal. Si el pedido se ha realizado correctamente, se muestra por pantalla un mensaje para dar ánimo al jugador.
2. Cuando se acaba el día. En el momento en el que la jornada laboral finaliza, se muestra por pantalla un mensaje informando sobre el progreso obtenido del jugador y la experiencia ganada ese día.

No escritos

Para no tener que mirar siempre la pantalla, se ha creído necesario crear feedbacks no escritos. La mayoría de ellos serán recibidos al pulsar el botón de entrega de pedido. Si el pedido no está bien creado, el botón se pondrá en rojo y una animación de humo saldrá de él. Además los controles sujetos en las manos

vibrarán. Por otro lado, si el pedido está bien creado el botón se pondrá en verde y un audio de aplausos y felicitaciones será reproducido.

El único feedback no escrito y que no tiene que ver con la entrega de pedidos es el gorro que tiene la pantalla. Este gorro cambiará de color cada vez que se suba de nivel.

5.4.2. Helpers

Por último, para hacer más fácil el entendimiento del entorno, se han añadido tres ayudas visuales a lo largo del primer nivel. Estas son:

- Mensaje inicial: En el momento que se entra al juego se verá un mensaje flotando en el escenario que indicará el botón a pulsar del mando para así activar el primer pedido del día.
- Flecha de ayuda: Para señalar a donde se ha de dirigir el jugador después de haber recibido el primer pedido por parte del cliente. Esta flecha se activará sólo en el primer pedido que se realice y desaparecerá al entregar este pedido correctamente.
- Texto de ayuda del botón: Encima del botón de confirmación que se utiliza para entregar los pedidos habrá un texto que indicará que se tiene que pulsar y la manera de hacerlo para así entregar el pedido realizado. Este texto estará activo durante todo el primer nivel y se desactivará cuando se supere durante el resto del juego.

5.5. Sistema de Logros

Es muy importante mantener la ambición por progresar en todo aquel que está aprendiendo. Por eso, se ha ampliado y modificado el sistema de logros en el modo de juego ya existente y se ha utilizado ese modelo actualizado para el nuevo modo de juego.

Para conseguir los logros, se guarda la cantidad de:

1. Pedidos totales realizados.
2. Pedidos básicos realizados: Pedidos desbloqueados desde el principio.
3. Pedidos condicionales básicos realizados: Pedidos desbloqueados en el nivel 2.
4. Pedidos condicionales compuestos realizados: Pedidos desbloqueados en el nivel 3.
5. Pedidos condicionales totales realizados.
6. Pedidos iterativos realizados: Pedidos desbloqueados en el nivel 4.

El jugador podrá conseguir:

Logros de los pedidos totales:

- 1 pedido completado.

- 10 pedidos completados.
- 25 pedidos completados.
- 50 pedidos completados.
- 100 pedidos completados.

Logros de los pedidos básicos:

- 1 pedido básico completado.
- 10 pedidos básicos completados.
- 25 pedidos básicos completados.
- 50 pedidos básicos completados

Logros de los pedidos condicionales básicos:

- 1 pedido condicional básico completado.
- 10 pedidos condicionales básicos completados.
- 30 pedidos condicionales básicos completados.

Logros de los pedidos condicionales compuestos:

- 1 pedido condicional compuesto.
- 10 pedidos condicionales compuestos.
- 30 pedidos condicionales compuestos.

Logros de los pedidos condicionales completados:

- 1 pedido condicional completado.
- 10 pedidos condicionales completado.
- 25 pedidos condicionales completado.
- 50 pedidos condicionales completado.

Logros de los pedidos iterativos completados:

- 1 pedido iterativo completado.
- 10 pedidos iterativos completados.
- 30 pedidos iterativos completados.

5.6. Distribución del espacio

Durante el proceso de diseño del juego, se ha de tener en cuenta la distribución del espacio para así tratar de evitar movimientos innecesarios o que puedan causar confusión. Esto gana más importancia si tenemos en cuenta que la Realidad Virtual nos puede producir mareos. Por lo tanto, hemos de eliminar cualquier movimiento que sea innecesario pero sin reducir el nivel de entretenimiento.

Para sumar realismo a la escena y reducir la sensación de agobio por estar en lo que podría parecer un espacio cerrado, se ha añadido una zona de parking exterior que el jugador podrá ver a través de las ventanas del restaurante.

El espacio de jugabilidad se divide en dos zonas: la zona del restaurante y la zona de pedidos las cuales están comunicados por una puerta que siempre se mantendrá abierta.

5.6.1. Distribución de la zona restaurante

Esta es la sala donde se recogen los pedidos y donde se encuentran las mesas con los clientes. Está divida en dos líneas de mesas también hay una barra de bar para darle realismo.

Se ha hecho que todas las mesas situadas en ambas filas estén a la misma altura, porque teniendo en cuenta que los pedidos se pueden recibir desde cualquier lado de las mesa, se consigue que sólo sea necesario el pasillo que comunica con la zona de pedidos. Así limitamos el movimiento necesario de los jugadores y reducimos la posibilidad de mareo.

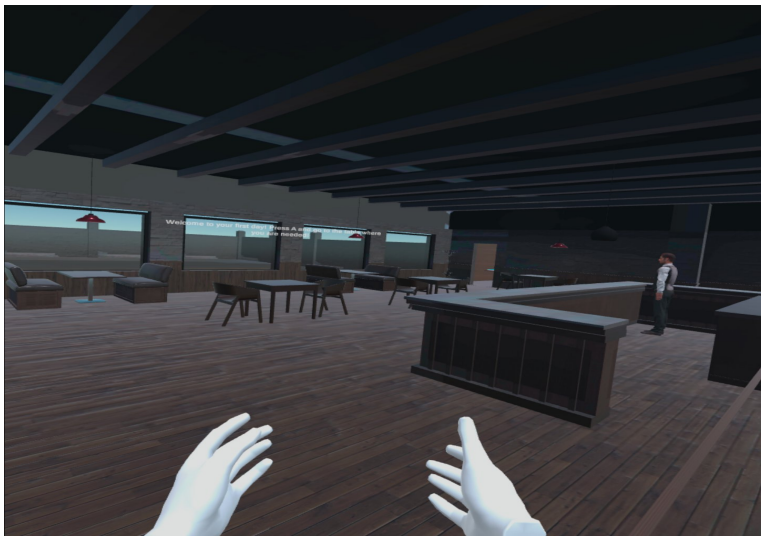

Figura 15: Zona Restaurante (Fuente: propia)

5.6.2. Distribución de la zona de pedidos

El primer objeto que se ve en esta zona es el televisor, que está situado al fondo de todo de la sala y lo veremos justo al entrar. Así el jugador se da cuenta desde el principio que el pedido dado por el cliente se puede ver en la pantalla, donde está escrito. Poniendo el televisor en esa posición se consigue que el jugador pueda ver la pantalla mientras realiza la creación del pedido.

Justo enfrente del televisor se encuentra una mesa con instrucciones. Esta mesa contiene los paneles básicos y el panel iterativo. Teniendo los paneles de los dos primeros niveles en esa mesa se consigue que el jugador se familiarice con las interacciones y el espacio de la zona de pedidos antes de añadirle más paneles.

A la derecha de la sala se encuentra la segunda mesa. Encima de esta mesa se verán todas las instrucciones condicionales.

Entre las dos mesas se encuentra la pizarra. En esa posición se consigue minimizar la cantidad de movimientos de cabeza y de manos. Así se facilita el proceso de creación del pedido y se reduce el mareo al eliminar el desplazamiento del jugador.

El último objeto de la sala es el botón. Este está colocado justo a la derecha de la pizarra, para hacer fácil tanto la interacción como su localización.

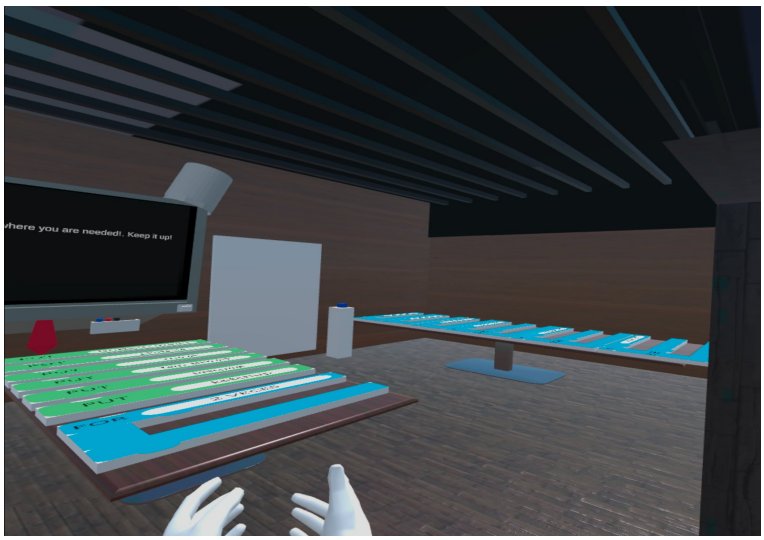

Figura 16: Zona de pedidos con todos los paneles desbloqueados (Fuente: propia)

6. Diseño de la aplicación

En este apartado se presenta el diseño que se ha realizado para el desarrollo del juego. Se verá el diagrama de clases completo, que contiene todas las nuevas clases creadas desde 0 y después se explicará las más importantes y su funcionalidad. A continuación, se expondrá los diagramas de flujo del juego. Después

se explicará el funcionamiento, estructura y contenido de la base de datos y finalmente se describirá los patrones de diseño utilizados, justificando su uso y sus beneficios.

6.1. Diagrama de clases

El diagrama de clases muestra todas aquellas clases que interaccionan a lo largo del juego.

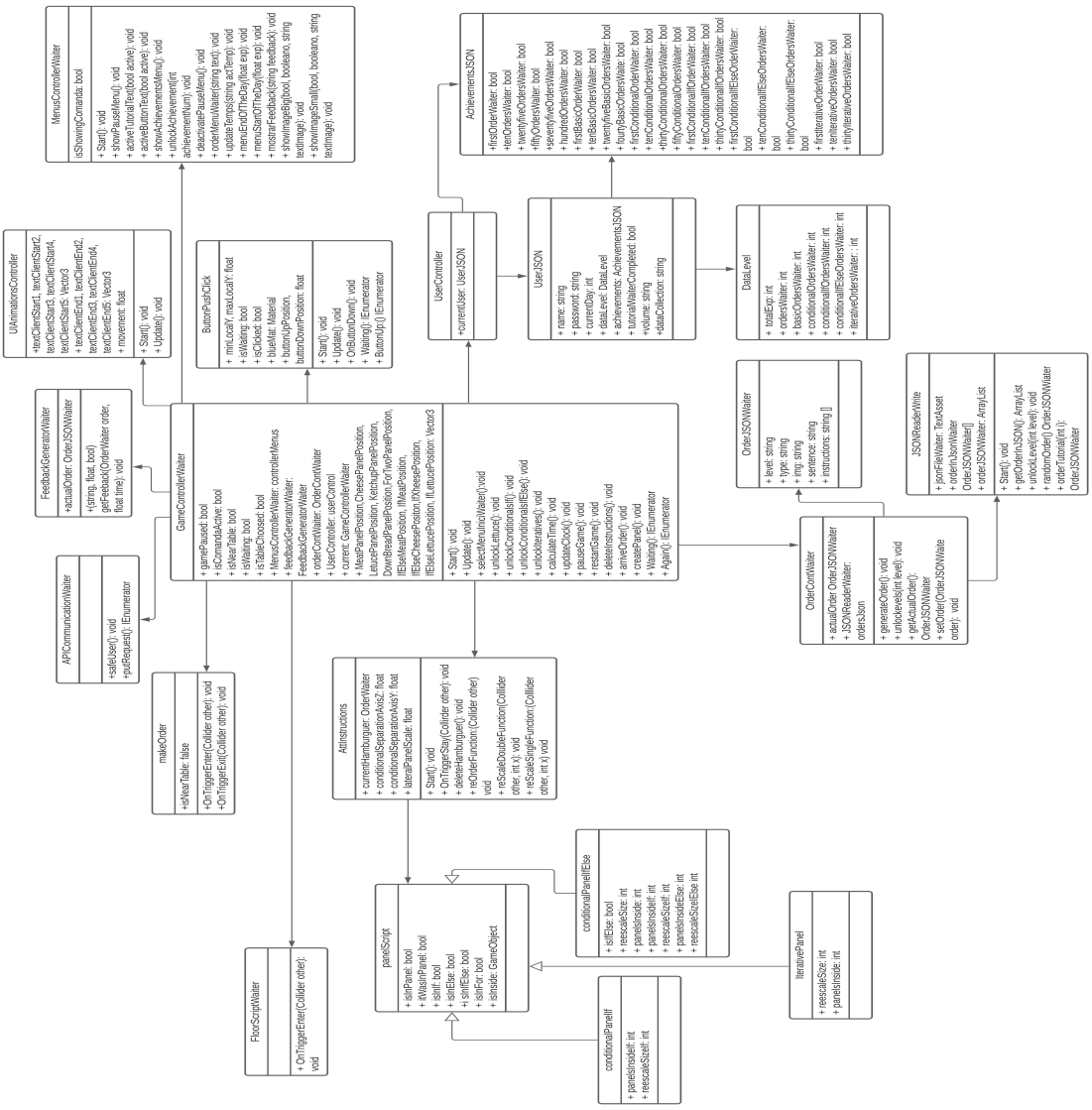

Figura 17: Diagrama de clases (Fuente: propia)

Controlador de la Partida

La clase *GameControllerWaiter* es la encargada de controlar y gestionar el funcionamiento de la partida y de lanzar los eventos, por lo tanto está conectada con muchas de las otras clases. Este controlador principal está conectado con otros controladores que nos servirán para gestionar subprocesos del bucle del juego como la interfaz del Menú, los logros conseguidos o las animaciones. Además tiene una instancia de ella misma, utilizando así el patrón Singleton (ver sección 8, apartado singleton).

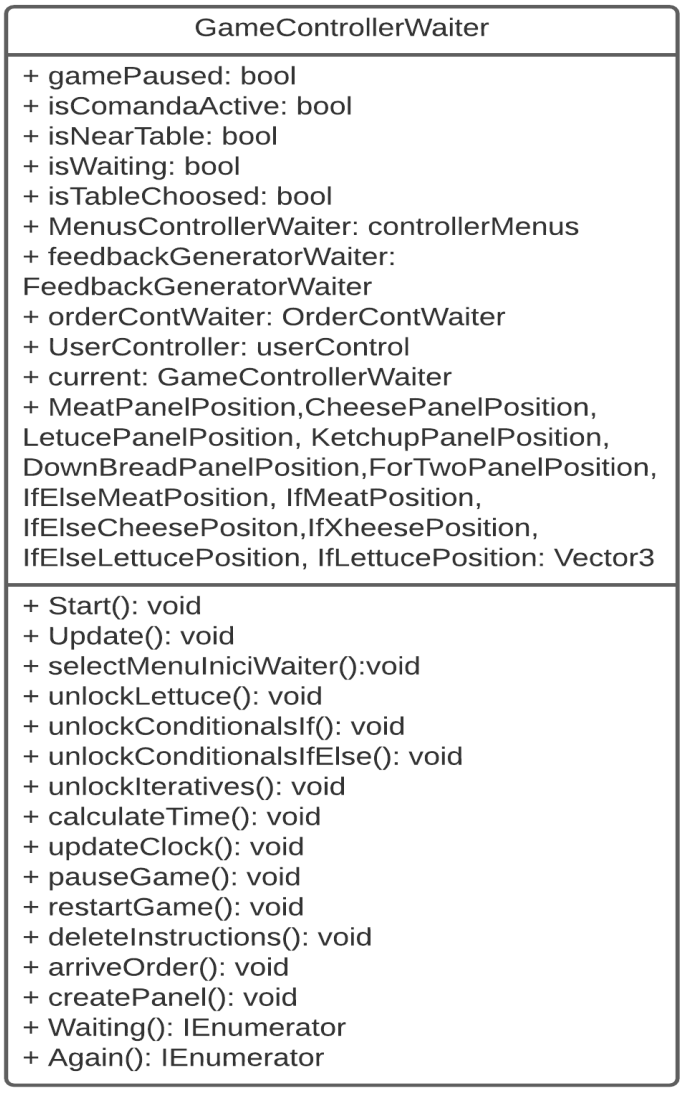

Figura 18: Clase del Controlador de la Partida (Fuente: propia)

Controlador de Menús

El controlador de menús, permite interaccionar con todos los elementos del menú y de la pantalla de la sala de pedidos. Se puede deducir de los métodos que contiene que es la única manera de interaccionar con los diferentes menús de la aplicación. Esto ayuda a tener un mayor control sobre la activación y desactivación de los elementos de interfaz de usuario.

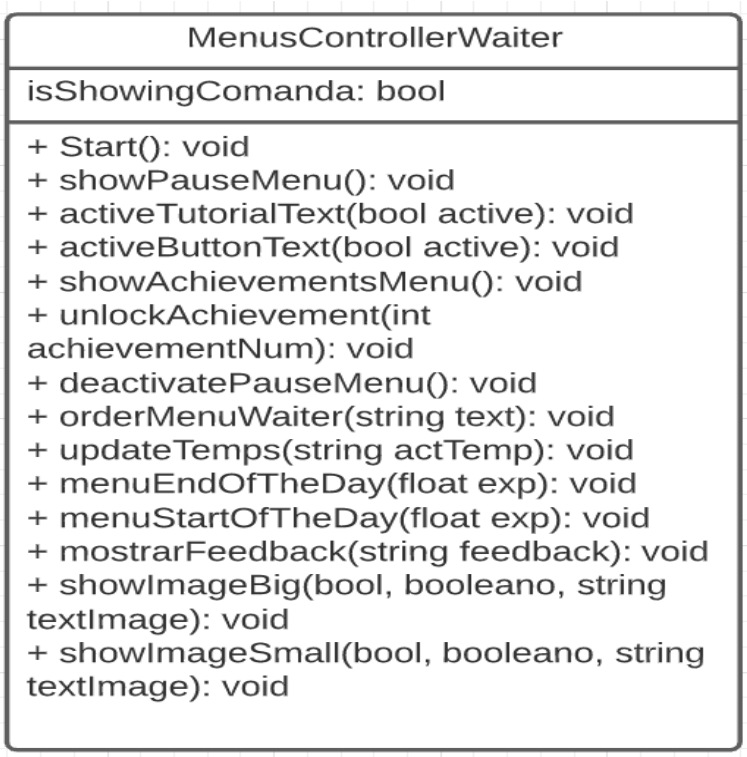

Figura 19: Clase del Controlador de los Menús y los elementos UI (Fuente: propia)

Usuario

La información del usuario llega desde la base de datos en formato JSON (JavaScript Object Notation) y esta información es la que se transforma en todas aquellas clases comentadas en este apartado. Estás clases servirán para volcar la información en las clases: *UserJSON*, *DataLevel* y *AchievementsJSON*. Igual que en los otras clases mencionadas anteriormente, con la clase *UserController* se mantiene un acceso ordenado a toda la información. Esta clase tiene de utilidad proporcionar los métodos de consulta y modificación de la información de las clases relacionadas con ella.

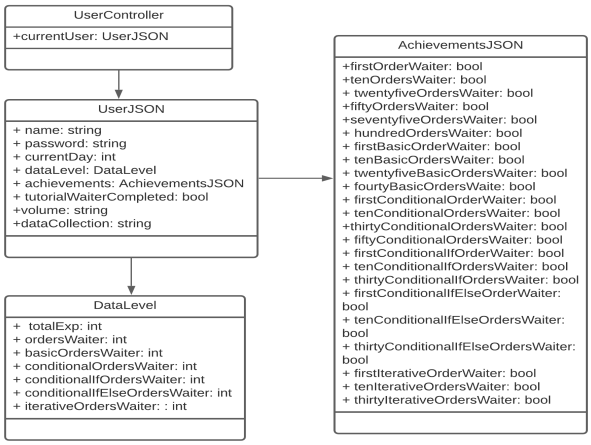

Figura 20: Clase que intervienen en la gestión de la información del usuario (Fuente: propia)

Pedidos

La información de los pedidos, al igual que la información del usuario, viene en formato JSON. Este archivo no proviene de la base de datos ya que se guardará localmente. Todos los pedidos se almacenan en la clase *JSONReaderWaiter* , donde se tiene una lista de todos los pedidos leídos, que están representados individualmente por la clase *OrderJSONWaiter*. Todo esto es gestionado por la clase *OrderContWaiter* que llama al controlador de la partida cuando es el momento de generar un nuevo pedido.

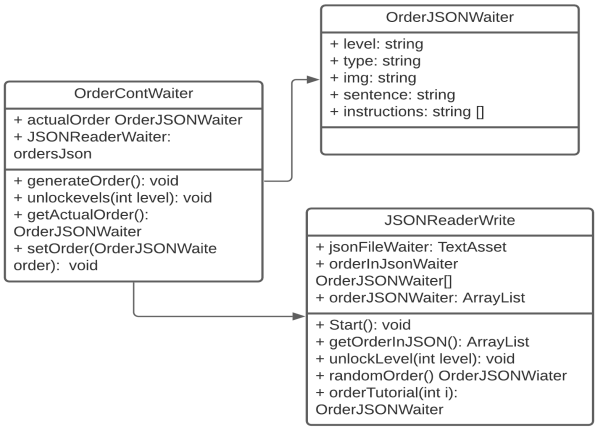

Figura 21: Clases que intervienen en la gestión la información de los pedidos (Fuente: propia)

Creación de los pedidos

El pedido se crea colocando los paneles en la pizarra. Para eso, la pizarra contiene la clase *AttInstructions*, que permite saber que paneles se colocan o quitan durante el juego, cuantos hay en la pizarra y en que orden están.

Además, se ha clasificado los paneles por clases, dependiendo de sus particularidades. Los paneles básicos son de la clase *panelScript* y a partir de ahí se encuentran dos más que heredan de esta: *conditionalPanelIf*, clase de los paneles condicionales básicos y *iterativePanel*, clase de los paneles iterativos. Finalmente *conditionalPanelIfElse* es la clase de los paneles condicionales compuestos y hereda de la clase *conditionalPanelIf*.

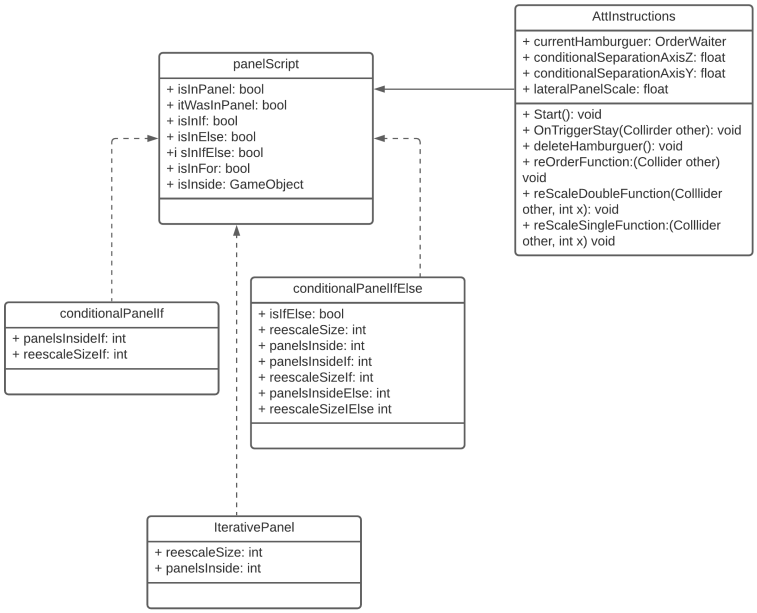

Figura 22: Clases que intervienen en la creación de los pedidos (Fuente: propia)

Feedback

La clase que genera el feedback que se mostrará al jugador cuando se entregue un pedido es *FeedbackGeneratorWaiter*. Su único atributo es el pedido que se muestra en ese momento en la pantalla, recibiéndolo de *OrderContWaiter* (ver sección 8.0, apartado de pedidos). Además, el algoritmo encargado de generarlo retorna una tupla (string, float, bool), donde la primera parte es el mensaje a mostrar, la segunda la experiencia obtenida y la tercera indica si el resultado ha sido correcto o no.

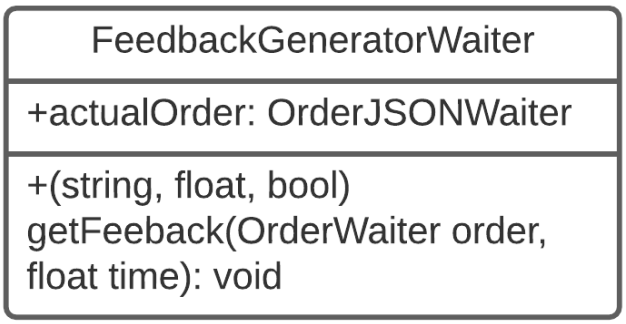

Figura 23: Clase que genera el feedback al usuario (Fuente: propia)

6.2. Base de Datos

La base de datos guarda toda la información del usuario. Eso permite que el dispositivo se pueda compartir, ya que más de una persona podrá utilizar la aplicación en el mismo dispositivo iniciando sesión en su propia cuenta. La información que se guarda en la base de datos son: nombre de usuario y contraseña, logros obtenidos, experiencia obtenida, si ha terminado el tutorial de ese modo e información de las interacciones del usuario durante el juego. Como se ha comentado en el apartado 9, hemos utilizado MongoDB . Esto significa que la base de datos es no relacional y basada en documentos.

El esquema principal es el que corresponde al jugador. Primero, se encuentran los campos necesarios para la autenticación, otros como el día en el que se ha quedado, si ha completado o no el tutorial de cada modo de juego y finalmente una variable de tipo string que guarda la información que quiera el desarrollado sobre la interacción del jugador durante la partida.

Esquema del usuario (UsuariSchema):

- name : string / Obligatorio
- password: string / Obligatorio
- currentDay : Number / Default: 1
- dataLevel: dataLevelSchema
- achievements: logrosSchema
- tutorialChefCompleted : Boolean / Default: false
- tutorialWaiterCompleted : Boolean / Default: false
- volume: string / Default: ”1.0”
- dataCollection: string / Default:

Por otro lado, tenemos el esquema de datos de la partida, que guarda la experiencia acumulada del usuario y el número de cada tipo de pedidos que ha completado con éxito. Esquema de datos de la partida (dataLevelSchema):

- totalExp : Number
- waiterExp: Number
- chefExp: Number
- levelWaiter: Number
- levelChef: Number
- basicOrdersChef : Number
- conditionalIfOrdersChef : Number
- conditionalIfElseOrdersChef : Number
- iterativeOrdersChef : Number
- basicOrdersWaiter : Number
- conditionalIfOrdersWaiter : Number
- conditionalIfElseOrdersWaiter : Number
- iterativeOrdersWaiter : Number

Por último, el esquema de logros, que guarda si el usuario ha completado el logro correspondiente. Esquema de logros (logrosSchema):

- firstOrderChef: Boolean / Default:false
- tenOrdersChef: Boolean / Default:false
- twentyfiveOrdersChef: Boolean / Default:false
- fiftyOrdersChef: Boolean / Default:false
- hundredOrdersChef: Boolean / Default:false
- firstBasicOrderChef: Boolean / Default:false
- tenBasicOrderChef: Boolean / Default:false
- twentyfiveBasicOrdersChef: Boolean / Default:false
- fiftyBasicOrdersChef: Boolean / Default:false
- firstIterativeOrderChef: Boolean / Default:false
- tenIterativeOrdersChef: Boolean / Default:false

- thirtyIterativeOrdersChef: Boolean / Default:false
- firstConditionalOrderChef: Boolean / Default:false
- tenConditionalOrdersChef: Boolean / Default:false
- twentyfiveConditionalOrdersChef: Boolean / Default:false
- fiftyConditionalOrdersChef: Boolean / Default:false
- firstConditionalIfOrderChef: Boolean / Default:false
- tenConditionalIfOrdersChef: Boolean / Default:false
- thirtyConditionalOrdersChef: Boolean / Default:false
- firstConditionalIfElseOrderChef: Boolean / Default:false
- tenConditionalIfElseOrdersChef: Boolean / Default:false
- thirtyConditionalIfElseOrdersChef: Boolean / Default:false
- firstOrderWaiter: Boolean / Default:false
- tenOrdersWaiter: Boolean / Default:false
- twentyfiveOrdersWaiter: Boolean / Default:false
- fiftyOrdersWaiter: Boolean / Default:false
- hundredOrderWaiter: Boolean / Default:false
- firstBasicOrderWaiter: Boolean / Default:false
- tenBasicOrderWaiter: Boolean / Default:false
- twentyfiveBasicOrdersWaiter: Boolean / Default:false
- fiftyBasicOrdersWaiter: Boolean / Default:false
- firstIterativeOrderWaiter: Boolean / Default:false
- tenIterativeOrdersWaiter: Boolean / Default:false
- thirtyIterativeOrdersWaiter: Boolean / Default:false
- firstConditionalOrderWaiter: Boolean / Default:false
- tenConditionalOrdersWaiter: Boolean / Default:false
- twentyfiveConditionalOrdersWaiter: Boolean / Default:false
- fiftyConditionalOrdersWaiter: Boolean / Default:false
- firstConditionalIfOrderWaiter: Boolean / Default:false
- tenConditionalIfOrdersWaiter: Boolean / Default:false

- thirtyConditionalOrdersWaiter: Boolean / Default:false
- firstConditionalIfElseOrderWaiter: Boolean / Default:false
- tenConditionalIfElseOrdersWaiter: Boolean / Default:false
- thirtyConditionalIfElseOrdersWaiter: Boolean / Default:false

7. Patrones utilizados

Los patrones de diseño son técnicas que sirven para resolver problemas en el desarrollo de software. Hay muchos patrones y durante el proyecto se han utilizado tres:

Adapter

Este patrón adapta una interfaz para que así pueda ser utilizada por otra clase que de otro modo no podría. Durante el proyecto es utilizada tres veces.

La primera con la clase JSONReaderWriter, donde utilizamos la librería JsonUtility para poder leer y escribir la información de los pedidos.

La segunda vez es cuando leemos la información del usuario de la base de datos, que volvemos a leer en formato JSON.

La tercera cuando guardamos el progreso ya que lo hemos de guardar en la base de datos en formato JSON.

Singleton

Singleton es un patrón de diseño que permite restringir la creación de objetos de una determinada clase, asegurando de esta forma que siempre se trabaje con una misma instancia. Este patrón ha sido utilizado en la clase GameControllerWaiter.

Como su nombre indica, es la clase que controlará todas las variables y orquestará los diferentes eventos o comunicaciones que se realicen en el juego. Es quién contendrá todas las variables que determinan el estado de la partida, instanciando o creando eventos cuando sea necesario. Para esta clase, interesa tener una única instancia, que además será accesible para una gran cantidad de objetos. Al ser el estado del juego un recurso disponible para toda la aplicación, aplicar el patrón Singleton facilitaba mucho el control, asegurando que siempre se trabaja con los mismos datos.

Modelo-Vista-Controlador

El patrón modelo-vista-controlador(MVC) es un patrón de arquitectura de software, que permite organizar y separar los datos de una forma más efectiva. De esta forma, se consigue separar cada una de

las responsabilidades, permitiendo un mejor desarrollo y una escalabilidad más fácil. Se ha utilizado este patrón para el backend[3]

Para este proyecto, se ha querido aplicar el MVC, pero adaptándolo a Unity.

En este proyecto las clases se organizan como:

- Modelo: Es donde se encuentran las clases que conectan con la base de datos, así como los esquemas que definirán como serán dichos datos.

- Controlador: En este caso, el controlador contiene gran parte de la lógica, lo que nos permite comunicar de forma segura la vista con el modelo.

- Vista(Rutas): Se encuentra la clase que definirá las diferentes rutas de nuestra aplicación. Además, definiremos que método del controlador tratará la petición que se realice a la dirección web.

De esta forma, el código queda mucho más legible y sencillo de utilizar. Por ejemplo, si queremos añadir una ruta nueva, no tenemos porque escribir código en las clases encargadas de conectar la base de datos, haciendo mucho más seguro el desarrollo.

8. Implementación del juego

8.1. Tecnologías Utilizadas

Oculus Quest

Las Oculus Quest[10] son unas gafas de realidad virtual lanzadas en Mayo de 2019 y desarrolladas por Oculus, compañía estadounidense que desarrolla tecnología de realidad virtual, perteneciente a Facebook desde 2014. Es un dispositivo independiente que puede ejecutar juegos y aplicaciones de forma inalámbrica bajo un sistema operativo basado en Android. Utiliza sensores internos y una serie de cámaras situadas en la parte frontal del visor, posibilitando el seguimiento posicional con seis grados de libertad permitiendo así que el jugador se mueva en los tres ejes tridimensionales: arriba/abajo, derecha/izquierda, delante/atrás. Las cámaras también forman parte del sistema de seguridad, que muestran la vista real cuando el usuario sale fuera del área designada. Gracias a una actualización posterior, las Oculus Quest también se pueden conectar a un ordenador mediante un dispositivo USB, dando así la posibilidad de jugar utilizándolas como las Oculus Rift, otras gafas desarrolladas por Oculus, que permiten ejecutar aplicaciones y juegos que demandan mayor potencia y rendimiento.

[3]El bakend es la parte del desarrollo que se encarga de que toda la lógica del juego funcione. El backend del juego consiste en un servidor y una base de datos

Figura 24: Controlador izquierdo, Visor, Controlador Derecho (Fuente: https://www.oculus.com/quest/))

Los controladores[9] de las Oculus Quest ajustan perfectamente a la mano. Cuentan con los dos botones básicos y un joystick en cada mando además de dos gatillos, uno situado justo en el cierre de la mano y otro donde se sitúa el dedo índice. Además, la circunferencia superior del mando permite detectar la altura del dedo situado en esa posición.

Todas estas características hacen de las Oculus Quest unas gafas muy adecuadas para ser utilizadas en un ambiente educativo. A diferencia de las mayorías gafas de realidad virtual no es necesario disponer de un ordenador para poder ejecutar las aplicaciones.

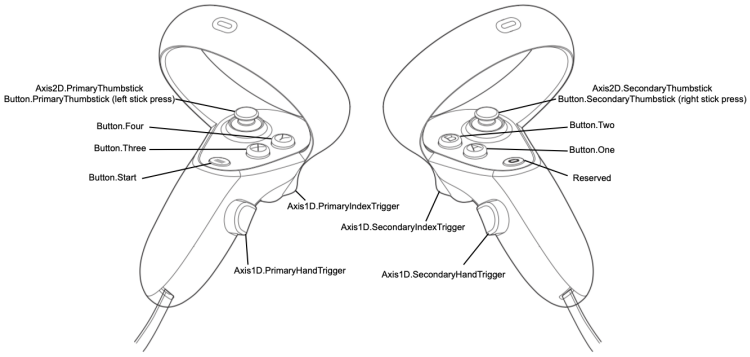

Figura 25: Oculus Controller Map (Fuente: https://developer.oculus.com/documentation/)

Unity

Unity[12] es un motor de videojuegos multiplataforma creado por Unity Technologies y está disponible en las plataformas de desarrollo de Mac OS, Microsoft Windows y Linux.

Unity es uno de los motores más conocidos y utilizados en todo el mundo, lo que permite resolver la mayoría de las dudas que se tienen a la hora de programar. Añadiendo eso a la compatibilidad con las Oculus Quest y a que Cooking Code había sido programado con Unity, la elección del motor de videojuego fue sencilla.

La versión utilizada ha sido la 2019.4[13], ya que es la última versión LTS (Long Term Support), las cuales están pensadas para proyectos que van a estar en desarrollo durante un largo periodo tiempo y que permiten tener una base sólida y sin grandes cambios en cada actualización.

NodeJS

NodeJS[8] es un entorno de programación en tiempo de ejecución multiplataforma, de código abierto e ideado para la creación de plataformas altamente escalables. Esta pensado para ser utilizado la mayoría de las veces del lado del servidor y el lenguaje utilizado es JavaScript.

Se ha utilizado este entorno para ampliar y modificar el backend del proyecto anterior con un lenguaje que ya conocía.

Express

Express[2] es un framework de aplicación web para NodeJS y está diseñado para crear aplicaciones web y API (Application Program Interfaces).

Es conocido como el framework estándar de NodeJS y por lo tanto en el momento en que se decidió que se seguría con el proyecto utilizando NodeJS, la utilización de Express era obvia.

MongoDB

MongoDB[7] es un sistema de base de datos NoSQL, orientado a documentos y ha objetos. MongoDB guarda las estructuras de datos con un esquema dinámico lo que permite que la utilización sea más fácil y rápida.

Esta base de datos ha sido la escogida para el desarrollo del proyecto por su sencillez, y porque los documentos estén en un formato parecido al JSON, formato de texto que se manipulará en Unity.

AtlasDB

AtlasDB[1] es una base de datos en la nube desarrollada por los creadores de MongoDB. En Cooking Code la base de datos utilizada era mLab pero su cierre obligó a tener que migrar los datos de mLab a AtlasDB. mLab ofrecía una serie de herramientas[6] para el proceso de migración que nos guiaría para efectuar todos los pasos.

Photoshop

Adobe Photoshop[11] es un editor de gráficos rasterizados creado por Adobe Systems. Se utiliza principalmente para modificar o retocar fotografías. Es líder mundial en el mercado de la edición de imágenes, dominando el sector de una forma clara.

Se ha utilizado Photoshop para poder modificar los pedidos ya existentes en el proyecto anterior e incorporar las fotografías de las acciones básicas a los paneles (ver sección 7.2.1) que se han utilizado para

escribir el pseudocódigo. Además, la cantidad de tutoriales y recursos que ayudan a su aprendizaje ha sido el motivo principal por el cual se han escogido esta herramienta de edición.

Github

GitHub[3] es una plataforma de colaboración y control de versiones de código abierto para desarrolladores de software. Se inició en 2008 y se fundó en Git, un sistema de gestión de código de fuente abierta creado por Linus Torvalds para hacer que el software se compile más rápido.

Git se usa para almacenar el código fuente de un proyecto y rastrear el historial completo de todos los cambios en ese código. Permite a los desarrolladores colaborar en un proyecto de forma más eficaz al proporcionar herramientas para gestionar cambios posiblemente conflictivos de varios desarrolladores. GitHub facilita la codificación social al proporcionar una interfaz web al repositorio de código Git y herramientas de administración para la colaboración. También permite a los desarrolladores cambiar, adaptar y mejorar el software de sus repositorios públicos de forma gratuita.

Todo el proyecto se puede encontrar en Github donde hay dos repositorios, uno del backend donde se subían las nuevas versiones cuando modificábamos la base de datos y también tenemos el repositorio donde hay el proyecto de Unity.

8.2. Requisitos para el desarrollo

Para desarrollar el proyecto se ha utilizado Unity como motor de videojuego, para el cual los requisitos mínimos son:

- SO: Windows 7 / 8 / 10
- Procesador: Core 2 Duo ó superior
- Memoria: 1 GB de RAM
- Gráficos: DirectX11 Compatible GPU con 512 MB Video RAM
- Almacenamiento: 100 MB de espacio disponible
- Tarjeta de sonido: DirectX compatible Tarjeta de sonido

Además, añadir que durante el proyecto se ha podido trabajar no sólo con los requisitos mínimos, sino con los requisitos recomendados:

- SO: Windows 7 / 8 / 10
- Procesador: Core 4 Duo ó superior
- Memoria: 2 GB de RAM
- Gráficos: DirectX11 Compatible GPU con 1 GB Video RAM
- Almacenamiento: 100 MB de espacio disponible

- Tarjeta de sonido: DirectX compatible Tarjeta de sonido

Ya que las características del ordenador con el que se ha desarrollado el proyecto han sido:

- SO: Windows 10
- Procesador: Intel(R) Core(TM) i7-7000
- Memoria: 16 GB de RAM
- Gráficos: GeForce GTX-1070
- Almacenamiento: 500GB MB de espacio disponible (antes de empezar el proyecto)
- Tarjeta de sonido: NVIDIA High Definition Audio

Por otro lado, para desarrollar con Oculus se han necesitado las Oculus Quest. Es importante utilizar estas gafas de realidad virtual ya que sino se pueden encontrar incompatibilidades en algunas funcionalidades.

Finalmente, también se ha utilizado un cable para poder conectar las gafas al ordenador y un móvil Android para poder enlazar la cuenta de Oculus a las gafas.

8.3. Hilo principal

Este apartado explica las diferentes funcionalidades que se han incorporando al juego o al propio jugador para así poder crear la experiencia de Restaurant Code.

8.3.1. Carga de nivel del usuario

La información del nivel se encuentra en la base de datos. Gracias a ciertas librerías de Unity, se puede extraer la información de un JSON y almacenarla en las correspondientes clases. Para ello, se ha de tener en cuenta que los nombres han de ser exactamente iguales y que se ha de respetar la jerarquía en todo momento. A la hora de cargar el usuario, se mira que nivel tiene y por lo tanto, que paneles ha desbloqueado previamente, para así mostrarle todos los paneles correspondientes a su nivel.

En ese momento, también se gestiona el archivo de pedidos que hay guardado localmente en formato JSON. Se vuelven a utilizar las librerías para extraer información de un archivo JSON. A la hora de generar un pedido, se escoge uno de dentro de la lista de pedidos con un nivel igual o inferior al del usuario de forma completamente aleatoria. Cuando un nuevo tipo de pedido es desbloqueado, se leen los niveles correspondientes en el JSON y se añaden a la lista de posibilidades. Cómo se ha leído el nivel del usuario previamente, se saben que pedidos se pueden generar acorde a ese nivel.

La información de este documento de pedidos queda volcada en determinadas clases del proyecto ver sección 8.1, apartado de Pedidos y su contenido se estructura de la siguiente forma:

- Level: Representa el nivel al que corresponde el pedido.
- Type: Que clase de pedido es: básico, condicional simple, compuesto o iterativo.

- Sentence: Corresponde al pedido en lenguaje natural.
- Instructions: Lista de instrucciones que corresponden al pedido.

8.3.2. Tutorial

La primera vez que el jugador entre en el juego, aparece en una sala parecida a la zona de pedidos, donde le enseñaran las dos cosas básicas para poder jugar a ese modo de juego.

Movimiento del jugador

La primera forma que tiene el jugador para moverse por el nivel es utilizando los joystick: el joystick izquierdo para moverse y el derecho para la cámara. Esto lo pudimos hacer gracias al OVRPlayerController, un prefab[4] del paquete Oculus Integration que permite al jugador moverse en el entorno virtual. Incluye componentes y objetos secundarios que son necesarios para el control 3D y también el prefab OVRCameraRig para que sirva como cámara de realidad virtual y está conectado a un controlador de personajes.

Como ya hemos comentado a lo largo del proyecto, uno de los grandes problemas, sobretodo cuando uno ha utilizado poco las gafas de realidad virtual, son los mareos que ocasiona la distorsión de que tu ojo crea que se está moviendo pero tu cuerpo no. Por eso se decidió incorporar otro tipo de movimiento, muy utilizado para los juegos de Realidad Virtual: el teletransporte.

Para es tuvimos que añadir otro prefab al al OVRPlayerController, al cual llamaríamos LocomotionController. Los scripts de este nuevo prefab son:

- Locomotion Controller: Hijo de OVRPlayerController que controla y centraliza la funcionalidad de los distintos tipos de teletransportes.
- Locomotion Teleport: Componente principal para controlar y centralizar la funcionalidad de los distintos tipos de telepuertos. Aquí, el parámetro más importante para la teletransportación es Teleport Destination Prefab, que contiene un objeto con un script apropiado de TeleportDestination (como nuestro TeleportDestination prefab) que se activa al apuntar. TeleportDestination de este tema.
- Teleport Input Handler Touch: el jugador podrá apuntar y activar el comportamiento de teletransportación usando los controladores Oculus Touch.
- Teleport Target Handler Physical: Este controlador de objetivos devuelve cualquier ubicación detectada por los tests de colisión de objetivos. Básicamente, cualquier espacio en el que quepa el jugador es un destino de teletransporte válido.
- Teleport Aim Visual Efect: El encargado de mostrar visualmente el puntero láser en los manipuladores objetivo.
- Teleport Aim Handler Parabolic: Este controlador de objetivo simula la curva parabólica que seguiría un objeto lanzado.

[4]Tipo de asset que permite al usuario almacenar un objeto GameObject completamente con componentes y propiedades

- Teleport Orientation Handler Thumbstick: Controlador de orientación que usa un joystick específico para seleccionar la orientación de aterrizaje después de un teletransporte.
- Teleport Transition Instant: Esta transición mueve al jugador instantáneamente sin otros efectos.

Todos estos scripts se incorporan desde el paquete Oculus Integration. Con este nuevo tipo de movimiento permitimos a los jugadores no iniciados en realidad virtual otro tipo de movimiento, menos realista, pero que reduce la sensación de mareo.

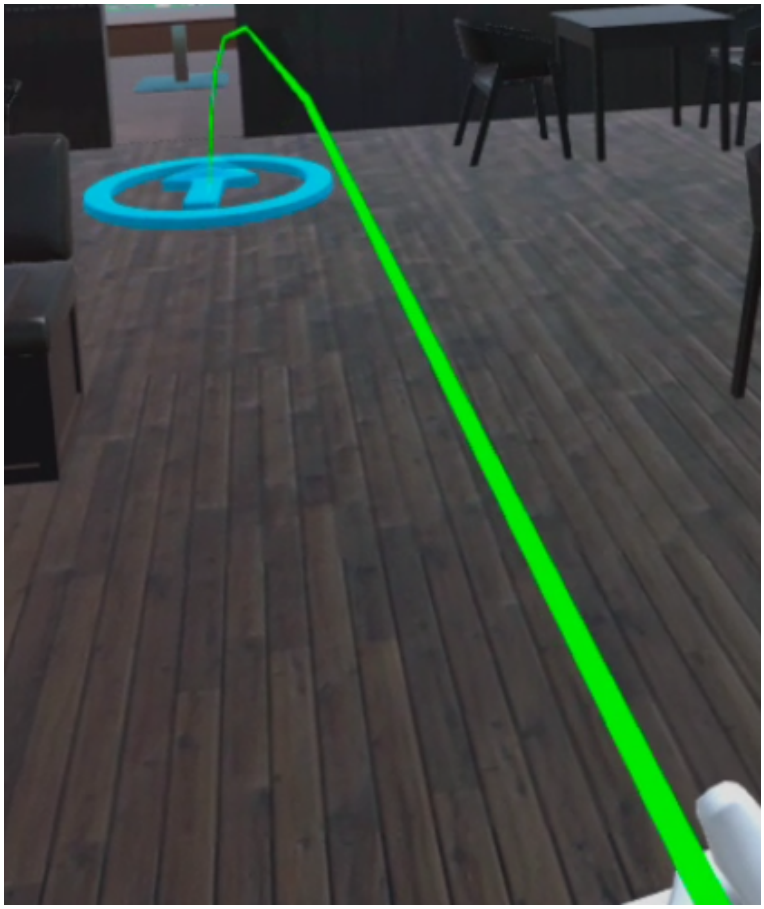

Figura 26: Jugador seleccionando sitio donde teletransportarse. Fuente: propia

Interacción con los paneles

Para poder interaccionar con los paneles se han utilizado dos clases: OVRGrabber y OVRGrabbable. Estas clases vienen con el paquete de Oculus Quest y gracias a ellas se puede coger, mover y soltar los paneles.

Durante el tutorial, se enseña tanto a coger los paneles como a soltarlos en la pizarra. Para familiarizarse con ellos, cada acción se tiene que realizar dos veces y durante ese proceso también se tiene que quitar un panel de la pizarra, para saber que también existe esa posibilidad.

Durante este proceso intervienen algoritmos explicados más adelante (ver sección 10.2.5)

Al final de la interacción es al jugador se le da la opción de seguir en el tutorial o de empezar a jugar.

8.3.3. Preguntar el pedido

Para que el jugador sepa que pedido tiene que que crear se realizan los siguientes procesos (ver Algoritmo 1):

- El jugador pulsa A. El sistema comprueba que no se haya pulsado previamente, para así no cambiar la mesa donde ya se le ha avisado. Además, como se ha explicado en la sección Helpers 6.4.2 cuando el jugador entra por primera vez en escena, aparece un texto de para que así sepa que botón ha de pulsar para iniciar el siguiente proceso.
- Se muestra un texto (”HERE!) encima de la mesa donde se tiene que dirigir.
- Cuando el jugador está lo suficientemente cerca de la mesa, vuelve a pulsar A y se genera el pedido que debe crear en pseudocódigo.
- El texto (”HERE!”) cambia al pedido que el jugador debe hacer.

Figura 27: Aviso de la mesa donde te dirán el pedido que quieren (Fuente: propia)

8.3.4. Creación del pedido

Para poder crear el pedido, el jugador debe dirigirse a la sala de pedidos donde se encuentran los paneles y la pizarra. La primera vez que se debe dirigir a ella, aparece una flecha para facilitar la localización de la sala.

Para poder crear el pedido, el jugador debe interaccionar con los paneles y colocarlos en orden en la pizarra. Ya se ha explicado como funcionaba la interacción con los paneles ((ver sección 9.3.2, apartado Interacción con los paneles)), por lo tanto falta saber como controlamos la colocación de estos en la pizarra.

El algoritmo principal que se utiliza se encuentra en la clase *AttIngredients*. Este algoritmo se activa cuando un panel es soltado cerca de la pizarra. En ese momento, coloca el panel automáticamente en su posición, añade el panel al pedido que estamos creando y si el panel que se está utilizando no ha sido cogido de la pizarra, crea uno igual en la mesa.

Para poder colocar el panel en su posición, los paneles han sido creados con una estructura 3D concreta:

- Panel Básico: El panel básico se compone sólo por un panel 3D.
- Panel condicional simple: A diferencia del panel básico, el panel condicional simple se compone de tres paneles 3D: el panel superior que es el que contiene la instrucción IF, el panel inferior o CLOSING para cerrar el condicional y el panel lateral que une ambas partes y que deja un espacio vacío para poder colocar las instrucciones dentro.
- Panel condicional compuesto: Es muy parecido al condicional simple pero añadiendo un panel intermedio para el ELSE y otro lateral para unir el IF y el ELSE.
- Panel iterativo: Exactamente igual que el panel condicional simple, cambiando la instrucción IF por una instrucción FOR.

En el caso de los paneles condicionales e iterativos, no sólo se añade el panel principal (IF o FOR) al pedido actual, también se añade el CLOSING del panel. Para el panel condicional compuesto, también se añade el ELSE. Esto es necesario para que a la hora de comprobar si el pedido está bien realizado, sepamos si las instrucciones están dentro o fuera de las condiciones.

Durante todo el proceso de colocación, el algoritmo controla tres cosas:

- Si el panel es el primero en colocarse. En ese caso, el algoritmo lo coloca arriba de todo de la pizarra.
- Si el panel es colocado debajo de todo. Entonces el algoritmo lo coloca justo debajo del último panel de la pizarra. Si el panel que se coloca debajo es uno que ya estaba en la pizarra, hay que utilizar el Algoritmo de reecolocación para no dejar un espacio en blanco en la pizarra.
- Si el panel es colocado entre otros dos paneles. En ese momento entrará en juego otro algoritmo: El Algoritmo de colocación entre paneles, explicado en el siguiente apartado.

Algorithm 1: PanelPlacement
input:panelToPlace
output:panelPlaced

```
actualOrder //order where the placed panels are stored
positionX //blackboard X position
positionY //blackboard Y position
positionZ //blackboard Z position
scaleX //panelToPlace X scale
scaleY //panelToPlace Y scale
scaleZ //panelToPlace escala Z
const lateralSeparation = 0.22f; //lateral separation to put instructions inside IF/ELSE/FOR
const verticalSeparation = 0.18f; //vertical separation to put instructions inside IF/ELSE/FOR
if numberOfPanels(actualOrder)==0 then
    placePanel(positionX + scaleY, positionY + scaleZ,positionZ)
    orientatePanel(180,180,90);
    addInstructionToCurrentOrder(panelToPlace);
    freezePanel();
else if positonY(panelToPlaced) <lastInstructionPositionY(panelToPlace) then
    placePanel(positionX + scaleY, lastInstructionPositionY(currentOrder) -
     conditionalSeparationAxisY, positionZ)
    rotatePanelToPlace(panelToPlace, 180, 180, 90);
    addInstructionToCurrentOrder(panelToPlace);
    FreezeAllConstraints(panelToPLaced);
    isInPanel(panelToPlace);
    if isFor(panelToPlace) or isIf(panelToPlace) then
        addClosingToCurrentOrder(panelToPlace);
    if isIfElse(panelToPlaced) then
        addClosingToCurrentOrder(panelToPlace);
        addElseToCurrentOrder(panelToPlace);
    if itWasInPanel(panelToPlace) then
        int i=1;
         while i<totalInstructions(actualOrder) do
            reOrderFunction(panelToPlace);
             i++;
        end
    else
        createNewPanel(panelToPlace);
    end
else
    betweenPanelsFunction();
    int i=1;
     while i<totalInstructions(actualOrder) do
        reOrderFunction(i,panelToPlace);
         i++;
    end
end
```

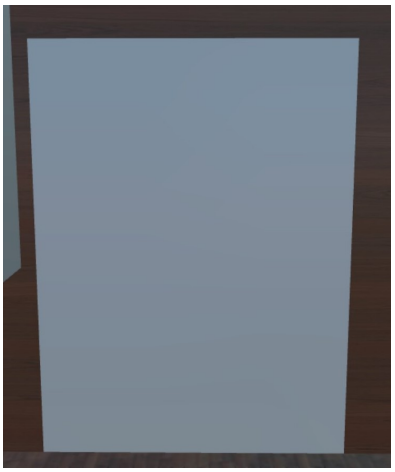

Figura 28: Pizarra vacía, antes de que se coloquen los paneles (Fuente: propia)

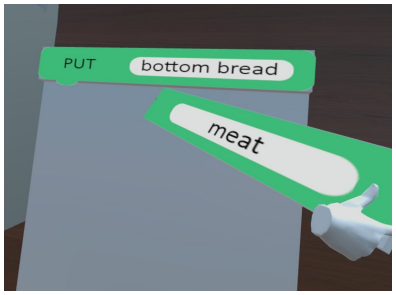

Figura 29: El jugador se dispone a colocar un panel debajo de todo (Fuente: propia)

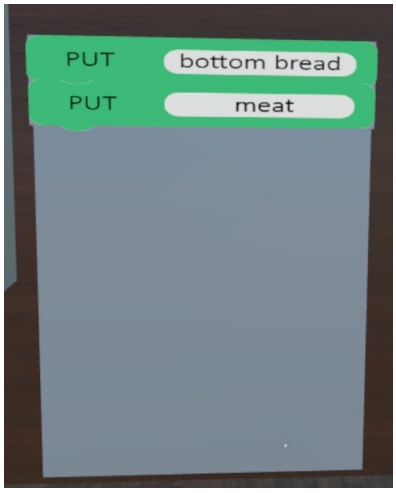

Figura 30: Paneles colocados (Fuente: propia)

Algoritmo de colocación entre paneles

Este algoritmo recorrerá la posición todos los paneles ya colocados hasta encontrar entre que dos paneles ha sido colocado el nuevo panel.

Después de encontrar la posición, el algoritmo hace las siguientes comprobaciones, por este orden, del panel:

- Si se ha colocado debajo de un CLOSING.
- Si se ha colocado debajo de un ELSE.
- Si se ha colocado debajo de un panel que se encuentra dentro de un ELSE.
- Si se ha colocado debajo de un IF de un panel condicional compuesto.
- Si se ha colocado debajo de un panel que se encuentra dentro del IF de un panel condicional compuesto.
- Si se ha colocado debajo de un FOR.
- Si se ha colocado debajo de un panel que se encuentra dentro del FOR.
- Si se ha colocado debajo de un IF.
- Si se ha colocado debajo de un panel que se encuentra dentro del IF.
- Cualquier otra opción, por lo tanto, cuando se coloca entre dos paneles básicos que no estén dentro de ningún panel condicional ni iterativo.

Durante todo este proceso, se ha de tener en cuenta que si el algoritmo detecta que entra a una de las condiciones, ya no puede entrar a ninguna de las siguientes.

En este algoritmo se vuelve a controlar que tipo de panel se está colocando, ya que, como hemos visto en el algoritmo anterior en el caso de cualquier panel no básico se debe añadir el CLOSING y en el caso de un panel condicional compuesto, también el ELSE.

Algoritmo de Recolocación

Como bien dice su nombre, este algoritmo controla la recolocación de los paneles y se llama en dos ocasiones:

- En el caso de colocar un panel entre dos colocados previamente.
- En el caso de recolocar un panel que ya estaba colocado.

Para eso, recorre todos los paneles a partir del segundo y hace cinco comprobaciones por cada panel:

- Primero comprueba que los paneles que hay colocados no sean paneles hijos (CLOSING o ELSE), ya que estos se moverán directamente con el padre.
- Después controla si el panel es un panel condicional compuesto. Entonces primero llama al algoritmo *reScaleDoubleFunction* para comprobar si se ha de hacer el reescalado y en caso afirmativo, hacerlo. Después recoloca el panel en su nueva posición.
- Si el panel es un condicional simple o un iterativo, llama al algoritmo reScaleSingleFunction, para comprobar si se ha de reescalar. Al igual que en el anterior, después de hacer el reescalado, coloca el panel en su nueva posición.
- En el caso de que el panel esté dentro de uno de los paneles condicionales o iterativo, recoloca el panel teniendo en cuenta la separación lateral.
- Si el panel no entra en ninguno de los casos anteriores, simplemente lo coloca en su nueva posición.

Algorithm 2: reOrderInstruction
input:panelToOrder, i
output:panelOrdered

```
actualOrder //order where the placed panels are stored
positionX //blackboard X position
positionZ //blackboard Z position
panelOrdering = locatePanel(actualOrder, i); //panel is being ordered
panelBeforePositionY //Y position of panel before panel it is being ordered
scaleY //panelToPlace Y scale
const lateralSeparation = 0.22f; //lateral separation to put instructions inside IF/ELSE/FOR
const verticalSeparation = 0.18f; //vertical separation to put instructions inside IF/ELSE/FOR

if panelName(i,actualOrder) != "ClosingPanel" and panelName(i, actualOrder) !=
 "ElsePanel" then
    if isIfElse(actualOrder, i) then
        reScaleDoubleFunction(panelOrdering); placePanel(positionX+scaleY,
         panelBeforePositionY - verticalSeparation, positionZ)
    else if isFor(panelOrdering) or isIf(panelOrdering) then
        reScaleSingleFunction(panelOrdering); placePanel(positionX + scaleY,
         panelBeforePositionY - verticalSeparation, positionZ);
    else if isInIf(panelOrdering) or isInIf(panelOrdering) or isInElse(panelOrdering) then
        placePanel(positionX + scaleY, panelBeforePositionY - verticalSeparation,positionZ +
         lateralSeparation);
    else
        placePanel(positionX + scaleY, panelBeforePositionY - verticalSeparation,positionZ);
    end
```

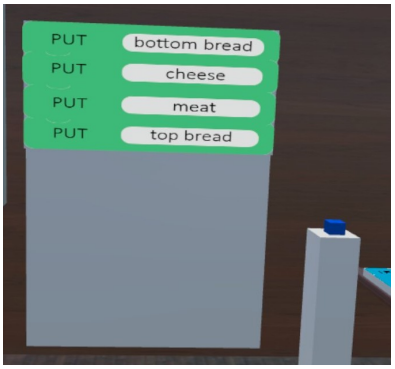

Figura 31: Paneles colocados, pero falta por añadir un panel entre la carne y el pan superior(Fuente: propia)

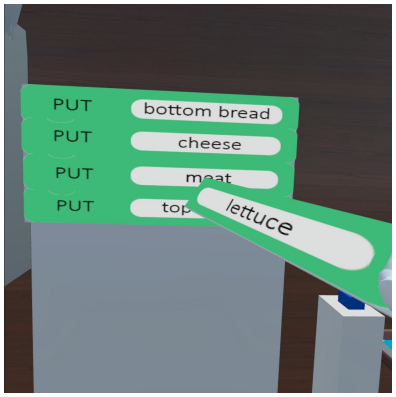

Figura 32: El jugador se dispone a colocar el panel que falta (Fuente: propia)

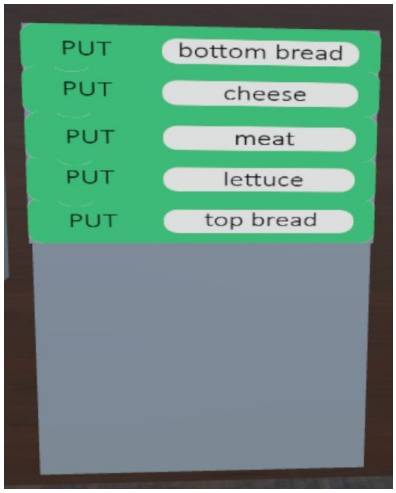

Figura 33: Paneles recolocados (Fuente: propia)

Algoritmo de reescalado

Este algoritmo sirve para controlar el tamaño del panel lateral de los paneles condicionales e iterativos. Este tamaño depende de la cantidad de paneles que tengamos dentro de las condiciones. Hay dos algoritmos de reescalado: el simple y el doble.

El algoritmo de reescalado simple es el más sencillo de los dos ya que sólo hay un panel lateral posible a reescalar. Para el reescalado se contemplan dos opciones:

- Si el jugador ha añadido un nuevo panel y ya tenemos otro dentro. Entonces reescalamos el panel lateral, sumándole el tamaño del nuevo panel introducido.
- Si el jugador ha extraído un panel ya existente. Si aún quedan paneles dentro de las condiciones, le restaremos el tamaño del panel extraído al panel lateral.

El algoritmo de reescalado doble es igual que el simple pero añadiendo una nueva comprobación: si el reescalado ha de ser del panel lateral que une al panel IF con el panel ELSE o ha de ser del panel lateral que une el ELSE con el CLOSING.

En ambos algoritmos hemos de tener en cuenta que el reescalado se hace tanto por la parte superior del panel lateral como por la inferior. Por lo tanto, deberemos también recolocar el panel lateral para que siga en contacto con los dos paneles a los que une.

Algorithm 3: reScaleSingleFunction
input:panelToRescale, i
output:panelRescaled

```
actualOrder //order where the placed panels are stored
positionX //blackboard X position
positionZ //blackboard Z position
panelOrdering = locatePanel(actualOrder, i); //panel is being ordered
panelBeforePositionY //Y position of panel before panel it is being ordered
scaleY //panelToPlace Y scale
const lateralSeparation = 0.22f; //lateral separation to put instructions inside IF/ELSE/FOR
const verticalSeparation = 0.18f; //vertical separation to put instructions inside IF/ELSE/FOR
panelsInside//panels inside Panel before rescaling rescaleSize// actual size of the reescale const
 lateralPanelScaleZ = 1.4 //Z scale size for one planel
if panelsInside < rescaleSize and panelsInside > 0 then
    panelsLeftInside = rescaleSize - panelsInside;
     lateralScaleX = getLateralScaleX(panelToRescale);
     lateralScaleY = getLateralScaleY(panelToRescale);
     newLateralScaleZ = lateralPanelScaleZ * panelsInside;
     setLocalScale(panelToRescale, (lateralScaleX, lateralScaleY, newLateralScaleZ));

     lateralPositionX = getLateralPositionX(panelToRescale);
     newLateralPositionY = getLateralPositionY(panelToRescale) + (0.1 * panelsLeftInside);
     lateralPositionZ = getLateralPositionZ(panelToRescale); setLateralPosition(panelToRescale,
     (lateralPositionX, newLateralPositionY, lateralPositionZ));

     closingPositionX = gettClosingPositionX(panelToRescale);
     newClosingPositionY = getClosingPositionY(panelToRescale) + (0.1 * panelsLeftInside);
     closingPositionZ = getClosingPositionZ(panelToRescale);
     setClosingPosition(panelToRescale, (closingPositionX, newClosingPositionY,
     closingPositionZ));
     z setRescaleSize(rescaleSize + 1);
else if panelsInside > reScaleSize and panelsInside > 1 then
    lateralScaleX = getLateralScaleX(panelToRescale);
     lateralScaleY = getLateralScaleY(panelToRescale);
     newLateralScaleZ = lateralPanelScaleZ * panelsInside;
     setLocalScale(panelToRescale, (lateralScaleX, lateralScaleY, newLateralScaleZ));

     lateralPositionX = getLateralPositionX(panelToRescale);
     newLateralPositionY = getLateralPositionY(panelToRescale) − 0.1;
     lateralPositionZ = getLateralPositionZ(panelToRescale);
     setLateralPosition(panelToRescale, (lateralPositionX, newLateralPositionY,
     lateralPositionZ));

     closingPositionX = gettClosingPositionX(panelToRescale);
     newClosingPositionY = getClosingPositionY(panelToRescale) - 0.2;
     closingPositionZ = getClosingPositionZ(panelToRescale);
     setClosingPosition(panelToRescale, (closingPositionX, newClosingPositionY,
     closingPositionZ));
                                                                    56
    setRescaleSize(rescaleSize + 1);
```

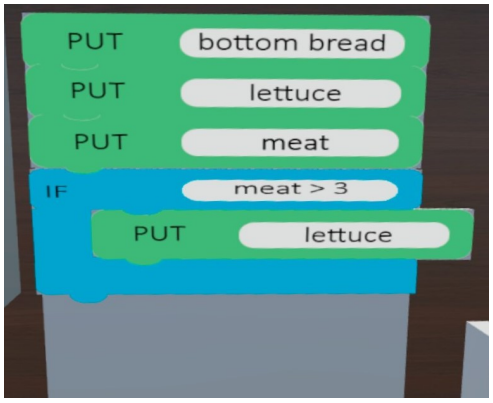

Figura 34: Sin reescalado. Sólo un panel ha sido colocado dentro del IF (Fuente: propia)]

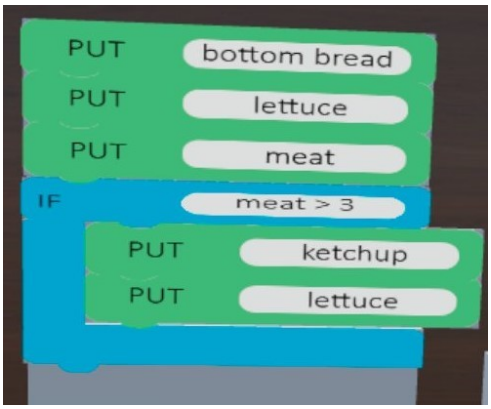

Figura 35: El segundo panel es colocado dentro del IF y por lo tanto se ejecuta el reescalado (Fuente:propia)

8.3.5. Entrega del pedido

Para entregar el pedido hemos de pulsar el botón que tendremos a la derecha de la pizarra. La primera vez que tenemos que interaccionar con el botón, veremos un mensaje encima de él que nos informará de ello. La clase encargada del movimiento del botón será la clase *buttonPushClick*.

En el momento en el que el botón sea pulsado, se genera el feedback y se muestra por pantalla. En el caso que el feedback sea positivo y el pedido esté bien realizado, los paneles de la pizarra serán destruidos. En caso contrario, los paneles se mantendrán en la pizarra para así poder arreglar el pedido.

Si el pedido ha estado bien generado, se activarán todas las comprobaciones para saber si el jugador a desbloqueado algún logro.

Para realizar estos procesos se utilizan las clases *FeedbackGeneratorWaiter*, *MenusControllerWaiter* y *GameControllerWaiter*.

9. Conclusiones

9.1. Resultados finales

Crear un videojuego no es una tarea para nada sencilla y si le añadimos la dificultad de crearlo en Realidad Virtual, la complejidad pasa a ser muy elevada, ya que los recursos disponibles son más bien escasos comparado con la mayoría de las otras plataformas. Por lo tanto, estoy muy orgulloso de haber conseguido los siguientes objetivos:

- Entender el código anterior y todas sus funcionalidades para así poder modificar lo que se ha creído necesario para mejorar el modo de juego ya existente.
- Familiarizarse con la plataforma de Oculus Quest, de la cual no tenía ningún conocimiento previo.

También he aprendido a programar con el lenguaje C#, el cual nunca había utilizado, y he profundizado más en mis conocimientos de Unity, los cuales eran muy básicos al empezar el proyecto.

- Poder diseñar los objetos de juego acorde a lo que se buscaba , con los modelos de figuras básicas de Unity y otros paquetes importados de su tienda.
- Crear un nuevo modo de juego, con mecánicas muy interesantes y siguiendo la idea de juego ya existente. Estas mecánicas han complicado el proceso de creación del juego ya que era difícil buscar ideas de como hacerlas, cosa que también lo ha hecho más interesante.
- Desarrollar casi todas las funcionalidades que teníamos en mente al principio del proyecto.

Para ver muestras del juego, hay que ir a: `https://www.youtube.com/channel/UCqhzRrTLJF-e4b3o-rk0V8w` donde se encuentran gameplays que muestran varias funcionalidades distintas del juego.

9.2. Trabajo futuro

Aunque este proyecto ha sido un gran paso para el juego, aún queda mucho por recorrer para considerar el juego terminado. Los siguientes apartados es todo aquello que nos gustaría incorporar en un futuro:

- Añadir un nuevo nivel para las instrucciones iterativas.
- Añadir un nuevo modo de juego que permita conocer los aspectos más básicos de la programación como la declaración de variables.
- Hacer un feedback más concreto.
- Crear modelos y animaciones que permitan que el escenario creado ser mucho más realista y reconocible.
- Hacer un estudio con usuarios reales para poder recopilar datos mientras juegan y poder mejorar el juego teniéndolos en cuenta.

10. Bibliografía

Referencias

[1] Atlasdb. `https://www.mongodb.com/cloud/atlas`. Accessed: 2021-06-02.

[2] Express. `https://expressjs.com/es/`. Accessed: 2021-06-02.

[3] Github. `https://github.com/`. Accessed: 2021-06-02.

[4] Gómez, I. Cooking code. projecte fi de grau de enginyeria informàtica. `http://diposit.ub.edu/dspace/handle/2445/172160`. Accessed: 2021-05-10.

[5] Huertos, A. A. Análisis de half life alyx, el juego que justifica la existencia de la realidad virtual. Accessed: 2021-06-02.

[6] Migration mlab to atlas documentation. https://github.com/. Accessed: 2021-06-02.

[7] Mongodb. http://https://www.mongodb.com/. Accessed: 2021-06-02.

[8] Nodejs. https://nodejs.org/es/. Accessed: 2021-06-02.

[9] Oculus controller. https://developer.oculus.com/documentation/unity/unity-ovrinput. Accessed: 2021-06-02.

[10] Oculus quest. https://www.oculus.com/quest/. Accessed: 2021-06-02.

[11] Photoshop. https://www.adobe.com/es/products/photoshop.html. Accessed: 2021-06-02.

[12] Unity. https://unity.com/. Accessed: 2021-06-02.

[13] Unitylts. https://unity3d.com/es/unity/qa/lts-releases. Accessed: 2021-06-02.

11. Apéndice 1 - Proceso de instalación y configuración

- Instalación Unity Hub: Lo primero que hay que hacer para poder utilizar Unity es instalar Unity Hub, que permite gestionar todos los proyectos en Unity.
- Desbloquear la opción de desarrollador en Oculus
  - Descarga la App de Oculus en el ordenador.
  - Enciende las Oculus Quest.
  - Selecciona Settings en la App.
  - Selecciona tus Oculus Quest de la lista de dispositivos.
  - Selecciona Other Settings.
  - Activa el Developer Mode.
- Instalación de las herramientas de Android
  - Descarga Android Studio.
  - Abre Android Studio y clica en File $->$ Settings $->$ Android SDK.
  - Utilizando el Package Manager descarga e instala:
    - Android 4.4 (SDK19).
    - Android 7.1 (SDK25).
    - SDK Tools (en la barra de navegación SDK Tools).
    - SDK - Platform (en la barra de navegación SDK Tools).
- Descarga la versión 2019.4 en Unity Hub y asegúrate que el soporte Android también va a ser instalado.

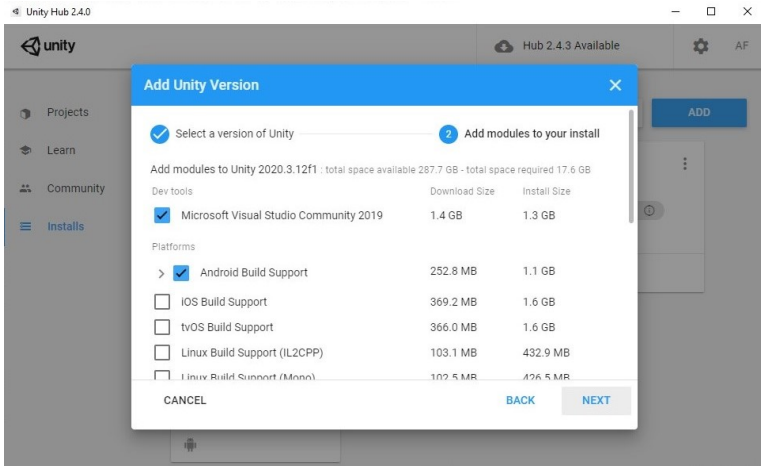

Figura 36: Instalando soporte de Android (Fuente: Propia)

- Descarga en la Asset Store el paquete Oculus Integration. Si se solicita, actualice y reinicia Unity.

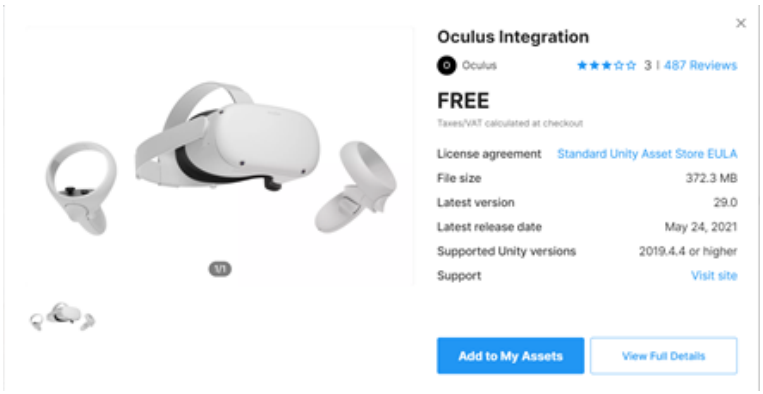

Figura 37: Paquete Oculus Integration de la Asset Store (Fuente:Propia

- En Unity, desde la escena que estás modificando establece la configuración del proyecto
  - Clica File y abre Build Settings.
  - Elimina las escenas que haya en Scenes in Build, pueden haber escenas de una configuración previa.
  - Añade las escena actual clicando en Add Open Scenes.
  - Cambia la plataforma a Android y la Texture Compression a ASTC.

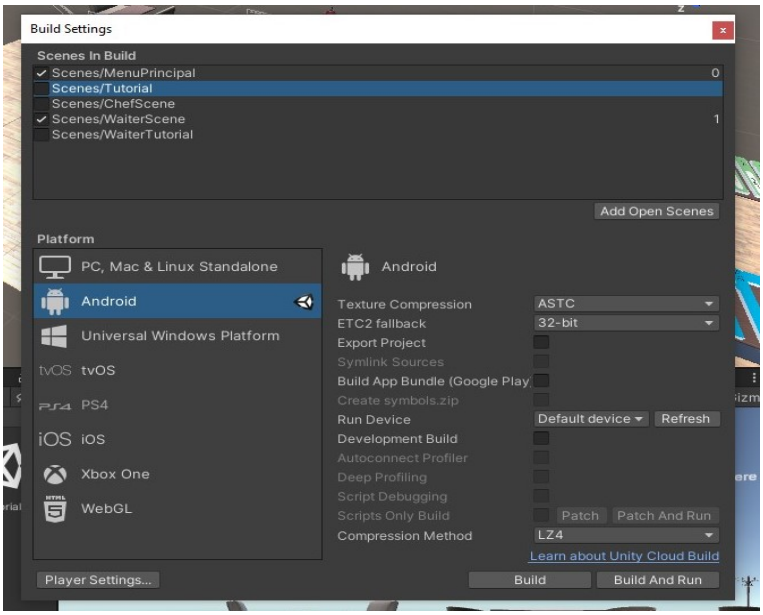

Figura 38: Build Settings (Fuente: Propia)

- Clica en Player Settings y cambia el Company Name y el Product Name.

- También en Player Settings ves a:

  - Other Settings − > Identification y cambia el mínimum API Level a 4.4 KitKat

  - XR Settings y confirma que la opción Virtual Reality Supported está activada.

  - La sección Virtual Reality SDKs, clica en el + y añade Oculus.

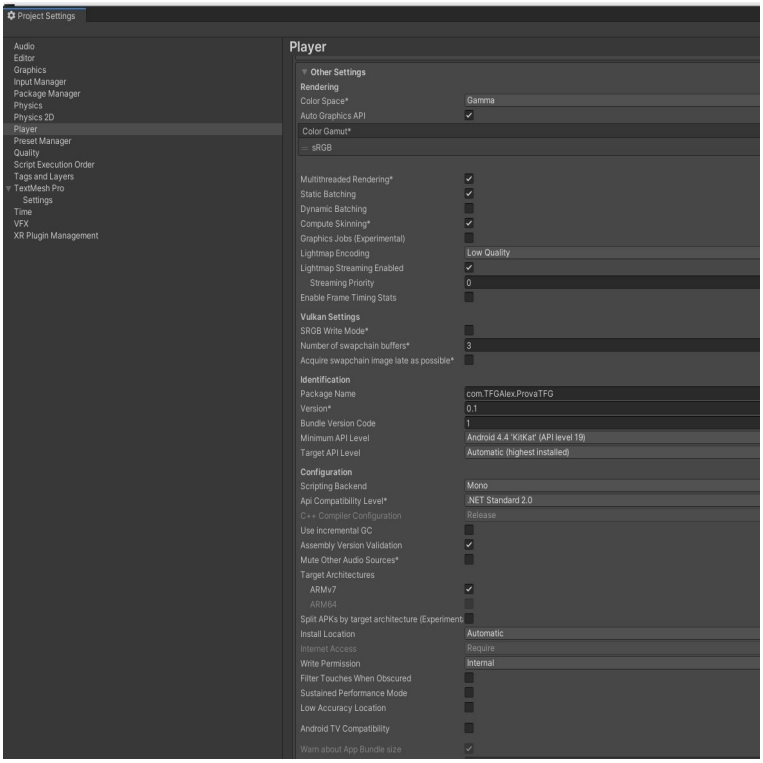

Figura 39: Other Settings (Fuente:Propia)

Figura 40: Player Projects Settings and XR Settings (Fuente:propia)

- Clica Build en Build Settings
- Si recibes un mensaje de error, ves a Player Settings− >Other Settings− >Graphics API y elimina Vulcan.

▪ Ejecuta el proyecto en las Oculus

- Conecta las Oculus al PC vía USB.
- Ponte las Oculus y activa la depuración USB para este PC.
- En Unity, ves a File− >Build Settings-¿Build and Run.
- Escoge un nombre para tu APK. Crea una nueva carpeta Builds y guarda tu APK ahí.
- Una vez que Unity ha instalado el juego en tus gafas, desconéctalas y póntelas.
- Desde tus gafas, podrás acceder al juego desde el Menú Principal− >Aplicaciones− >Orígenes Desconocidos.

12. Apéndice 2 - Guía para el jugador

En este documento se explica como jugar y que ha de hacer el jugador para poder tener un desarrollo del juego óptimo.

Restaurant Code de es un juego de VR que te pone en la piel de un camarero que ha de traducir a pseudocódigo los pedidos recibidos por los comensales. A medida que vas subiendo de nivel, desbloqueas instrucciones e ingredientes que te permitirán crear más variedad de pedidos. Además, cuando completas un número exacto de pedidos, desbloqueas diferentes logros.

Para poder jugar a *Restaurant Code*, necesitas las Oculus Quest y sus controles que te permiten moverte por el restaurante y crear los pedidos.

Los controles del juego són:

- Botón A: Tienes dos funciones
  1. Función "Continuar". Se utiliza para:
     - Iniciar el ciclo de pedidos
     - Cambiar aviso del comensal a su pedido.
     - Avanzar pantallas tanto en el tutorial como cuando desbloqueas un ingrediente.
  2. Función "Seleccionar". Utilizado en los menús:
     - Antes de empezar el juego, para escoger la opción que quieres ("LogIn" o "SignIn").
     - Después de escoger la opción, seleccionar las letras que quieres para tu nombre y contraseña.
     - Una vez te has registrado, escoger el modo de juego.
     - Dentro del juego, cuando nos encontramos en el menú, seleccionar la opción que deseas.
- Botón B: Salir del Menú de Juego.
- "Trigger" Superior Derecho: Mantenerlo para iniciar el Teletransporte y soltarlo para realizarlo.
- "Trigger" Inferior Derecho/Izquierdo: Hay que mantenerlos pulsados para ejecutar la acción de agarre de los paneles y soltarlo para dejarlos ir. También se utiliza cuando quieres pulsar el botón de confirmación de pedido.
- "Joystick" Izquierdo: Mover el jugador por el restaurante.
- "Joystick" Derecho: Si no se tiene el teletransporte activado, sirve para mover la cámara del jugador. Si se tiene activado (es decir, el Trigger Superior Derecho está pulsado), sirve para decidir dónde estará mirando el jugador cuando el teletransporte sea realizado.