Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

MÁSTER UNIVERSITARIO EN DESARROLLO DE VIDEOJUEGOS

Trabajo Fin de Máster

Colaboración en los proyectos de TenerifeJuega.

Collaboration in TenerifeJuega projects

Álvaro González Rodríguez

D. **Jesús Miguel Torres Jorge**, con N.I.F. 42.826.207-Y profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“Colaboración en los proyectos de TenerifeJuega”

ha sido realizada bajo su dirección por D. **Álvaro González Rodríguez**, con N.I.F. 79.083.919-Y, en la entidad colaboradora TenerifeJuega situada en Los Realejos y han sido tutorizadas por D. **José de la Osa Aldana**, fundador y codirector en la empresa TenerifeJuega, en calidad de tutor externo.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 17 de mayo de 2023.

Agradecimientos

Agradecer a la entidad colaboradora, Tenerife Juega, y al tutor externo José por acogerme y ayudarme en todo momento y tratarme como uno más de la empresa.

Agradecer a todos los profesores del Máster Universitario en Desarrollo de Videojuegos por demostrar su pasión y dedicación por la enseñanza.

Por último, agradecer a mi familia y amigos, todos han sido un gran respaldo y sin ellos no hubiera llegado a donde estoy hoy. Muchas gracias de corazón.

Resumen

El objetivo de este proyecto ha sido colaborar con la entidad TenerifeJuega con varios de sus proyectos activos, aportando los conocimientos adquiridos del presente máster de manera que, como resultado, agilice mis habilidades de desarrollo en Unity y finalice mi experiencia laboral con unas nuevas bases.

El inicio del trabajo fue el día 16 de marzo en Los Realejos y perduró de manera telemática los días siguientes hasta el 12 de mayo.

Se colaboró en cuatro proyectos concretos. Cada uno de ellos tienen sus distintas tareas y responsabilidades, siendo el primero el desarrollo de una aplicación en realidad aumentada que soporte el seguimiento de imágenes como pivote para distintos objetos en una escena de Unity, el segundo un prototipo propio de un videojuego del género Tower Defense en realidad aumentada, el tercero pequeñas aportaciones en un prototipo que hace uso de la inteligencia artificial ChatGPT y el cuarto una nueva aplicación de realidad aumentada que hace uso de códigos QR para generar distintos resultados sobre un mismo objeto.

Concretamente, durante el transcurso del trabajo, se usaron las tecnologías de Unity y AR Foundation para desarrollar los proyectos de realidad aumentada, mientras que en el trabajo con códigos QR se hace uso de la librería ZXing.

Palabras clave: realidad aumentada, seguimiento de imágenes, Unity, AR Foundation, QR.

Abstract

The objective of this work has been to collaborate with the entity TenerifeJuega with several of its active projects, providing the knowledge acquired from this master's degree. So, as a result, I speed up my development skills in Unity and end my work experience with a new basis.

The work began on March 16 in Los Realejos and lasted telematically for the following days until May 12.

I collaborated on four specific projects. Each of them has different tasks and responsibilities, being the first one the development of an augmented reality application that supports image tracking as a pivot for different objects in a Unity scene, the second one a prototype of a Tower Defense video game in augmented reality, the third one small contributions in a prototype that makes use of ChatGPT and the fourth one a new augmented reality application that makes use of QR codes to generate different results on the same object.

Specifically, during the course of the work, Unity and AR Foundation technologies were used to develop the augmented reality projects, while the work with QR codes made use of the ZXing library.

Keywords: augmented reality, image tracking, Unity, AR Foundation, QR.

Índice

Índice de figuras

Capítulo 1
Introducción

Este capítulo consta de una pequeña descripción de la empresa donde se realizó el Trabajo Fin de Máster, junto con una exposición de los objetivos impuestos por la empresa externa.

1.1. Antecedentes

TenerifeJuega[1] es una empresa que principalmente se dedica a realizar cursos de formación en el ámbito del desarrollo de videojuegos, principalmente en el motor Unreal Engine, el cual cuenta con múltiples cursos de distintos niveles para todas las edades. TenerifeJuega nace de una iniciativa tomada en el año 2013 por las Escuelas Artísticas de los Realejos[2], con el fin de asentar un ecosistema de futuros desarrolladores de videojuegos en Tenerife. Actualmente, cuentan con sedes en Los Realejos, Santa Cruz de Tenerife, La Laguna y Tacoronte.

Por otro lado, TenerifeJuega también desarrolla sus proyectos propios en lo relacionado con el mundo de los videojuegos, además de realizar otros pequeños proyectos para distintos clientes de las islas, por lo que se lo podría considerar como una empresa de formación y de desarrollo de videojuegos. Cuentan con un pequeño equipo que no supera las 5 personas. El desenvolvimiento de estos proyectos se realiza principalmente en Los Realejos, concretamente en la Oficina de AFAVER, Calle San Agustín.

Figura 1. Logo de TenerifeJuega.

1.2. Periodo y Departamentos

El periodo de prácticas para la realización del Trabajo Fin de Máster transcurrió desde el 16 de marzo de 2023 hasta el 12 mayo de 2023, con una jornada de 8 horas diarias todos los días laborales de la semana, de 8:00 a 16:00, haciendo un total de 300 horas. Se nos dio a mi tutor externo y a mí una total libertad para organizarse como quisiéramos, pero el inicio tardío de mi trabajo nos obligó a realizar 8 horas diarias de prácticas, para poder acabar dentro de un margen de tiempo acorde para poder redactar la presente memoria.

Principalmente, se trabajó en el departamento de desarrollo de forma telemática, ocasionalmente junto con uno o dos compañeros de trabajo, donde se está trabajando en un proyecto propio y van surgiendo proyectos de clientes externos, siendo mi principal aportación el trabajo en lo segundo usando realidad aumentada y manejo de código QR.

1.3. Objetivo General

El objetivo del Trabajo Fin de Máster junto con TenerifeJuega es el de colaborar con sus distintos proyectos activos y por venir, donde, al usar realidad aumentada (RA), se trabajará con Unity y C#. Para ello, se deberán de adquirir los conocimientos necesarios sobre RA para realizar las tareas de manera óptima y dentro de unos márgenes de tiempo realistas.

1.4. Objetivos Específicos

Más concretamente, se adquirirán conocimientos sobre el uso del Image Tracking[3] de RA en Unity durante los primeros días de trabajo, además de agilizar y mejorar los conocimientos sobre C#, ya que es el lenguaje de programación usado por Unity. Por lo que se podría considerar uno de los objetivos el aprender a implementar RA en una aplicación móvil e incrementar los conocimientos ya adquiridos en el máster.

Se trabajará en varias aplicaciones de RA, yendo desde la implementación de la lógica de una interfaz de usuario hasta la incorporación del Image Tracking, junto con las especificaciones propias de cada proyecto.

Se aprenderá a generar y manejar los datos de códigos QR para una de las aplicaciones de RA haciendo uso de una librería de Unity llamada ZXing[4].

Se aprenderá a usar Plane Tracking[5] durante una de las semanas como conocimientos extra para el desempeño del Trabajo Fin de Máster.

Se trabajará durante un periodo de tiempo determinado en uno de los proyectos internos de la empresa, que consta de la implementación en un videojuego de la inteligencia artificial ChatGPT[6] con el fin de que el usuario le dé órdenes a la máquina y esta responda y reaccione de manera acorde con los elementos de una escena.

Capítulo 2
Alcance y desarrollo

Este capítulo consta de una descripción detallada de todas las tareas realizadas durante el periodo de trabajo en TenerifeJuega, junto con la metodología de trabajo utilizada, al igual que un cronograma con los distintos periodos que se dedicaron a cada una de las tareas.

2.1. Tareas

La asignación de las tareas se realizó el primer día de trabajo con la empresa, el 16 de marzo de 13:00 a 17:00, a través de una reunión presencial en Los Realejos, donde se me dio distintas posibilidades con las que colaborar. Estas fueron trabajar en un proyecto de RA asignado por un cliente, trabajar en un proyecto de la empresa sobre inteligencia artificial, trabajar en el escenario de un proyecto de la empresa sobre un videojuego sobre zombies o trabajar en un proyecto propio donde TenerifeJuega me apoyara. Finalmente, opté por empezar trabajando en el proyecto de RA, el cual aún no había sido empezado, por lo que sería el encargado de crear el prototipo con el diseño que la empresa y el cliente tenían en mente. Para la preparación del trabajo, primero tendría que estudiar el funcionamiento de RA en Unity. Además, en esa misma reunión también se llegó a la conclusión de que podía hacer el trabajo de forma telemática para ahorrarme el trayecto de ida y vuelta a Los Realejos, ya que de otra forma llevaría más de una hora y media de trayectos diarios.

2.1.1. Aprendizaje

Este trabajo de aprendizaje vino de la mano de un pequeño estudio de las distintas posibilidades que ofrece Unity para el trabajo en RA. Esta aplicación requiere de un sistema que detecte imágenes y proyecte un modelo 3D sobre esa imagen. Destacaron dos framework:

AR Foundation[7]: La principal característica de este framework es que tiene la capacidad de incluir distintas funcionalidades de ARKit[8], ARCore[9], Magic Leap[10] y HoloLens[11], para construir aplicaciones que estén preparadas para ser lanzadas en los distintos dispositivos sin la necesidad de realizar cambios drásticos. Además, incluye múltiples funcionalidades ya incluidas en el framework como el Image Tracking, Plane Tracking, Face Tracking, Body Tracking, entre otras. Es gratuito y viene incluido en la sección de Unity Registry dentro de la ventana del Package Manager.

Vuforia [12]: Es compatible con Android e IOS, incluso utiliza ARKit o ARCore si el hardware donde está siendo utilizada lo permite. Su punto fuerte es el Object e Image Tracking, es decir, el seguimiento basado en marcadores. El principal inconveniente es el precio del framework, el cual es de pago si se quiere utilizar con todas las funcionalidades que ofrece.

Figura 2. Logo de Vuforia.

Aunque Vuforia ofrece un sistema de seguimiento de imágenes más refinado que AR Foundation, la diferencia en los precios y la facilidad para encontrar documentación y tutoriales sobre AR Foundation decanta más la balanza por este framework.

En lo que respecta sobre mi aprendizaje, los primeros días de trabajo pueden ser resumidos en consumir videotutoriales sobre AR Foundation y lecturas sobre su documentación[12], sobre todo en lo que respecta al Image Tracking. También se desarrolló un pequeño prototipo para comprobar que mi dispositivo móvil cumplía con los requisitos para ejecutar este tipo de aplicaciones y así no tener ningún tipo de inconvenientes durante el desarrollo.

2.2.1. Desarrollo de una aplicación AR

El proceso de desarrollo de la aplicación se realizó de forma autónoma, donde tuve la total libertad de experimentar y tomar mis propias decisiones, luego sería un compañero de trabajo quien valoraría mi trabajo y seguiría con las implementaciones más específicas. Mi principal objetivo en este desarrollo era básicamente preparar la lógica de la aplicación para que añadir cambios en un futuro no cueste mucho tiempo ni trabajo.

La aplicación trata de un tour turístico en la Orotava, donde el usuario deberá caminar e ir a distintos puntos de interés para escanear códigos QR. En estos códigos deberá de aparecer un objeto 3D de algún personaje y este modelo será el encargado de comentar al usuario sobre la información del lugar.

Como redacté en el estudio, para hacer uso de las funcionalidades de AR Foundation, se deben descargar las librerías correspondientes a la realidad aumentada. Estas son la propia de AR Foundation, ARCore para el funcionamiento en Android y ARKit para el funcionamiento en IOS.

Luego es importante realizar ciertos cambios en la configuración del proyecto, donde se tiene que habilitar los distintos plugins de XR para los distintos sistemas operativos. Además, para los dispositivos Android es importante deshabilitar la API gráfica de Vulkan, ya que no es compatible con AR Foundation, e incrementar el nivel mínimo de la API hasta Android 7.0.

Continuando con la implementación de la lógica, todas las aplicaciones de RA con AR Foundation en Unity necesitan de dos componentes esenciales, estos son AR Session[13] y AR Session Origin o XR Origin[14]. El primero controla el ciclo de vida y las opciones de configuración de una sesión de realidad aumentada y solo puede existir una de ellas en la escena, mientras que el segundo contiene una cámara de RA y cualquier GameObject que haya sido detectado por cualquiera de las funcionalidades y quiera ser añadido en la escena. También traduce lo que se conoce como el espacio de la sesión al espacio de Unity, para ello la posición donde se inicia el dispositivo deberá de ser la (0, 0, 0) del espacio de la sesión y todos los objetos que se necesitan colocar en la escena tendrán como referencia esa posición inicial.

El componente AR Session Origin es quien contiene todas las funcionalidades que se le quiera añadir a la aplicación de RA. En el caso que nos ocupa, se ha añadido el componente AR Tracked Image Manager. Este componente necesita principalmente de una librería serializada de imágenes y un prefab del objeto a añadir en la escena. Estas librerías son un elemento que viene incluido en AR Foundation, donde se tiene que asignar una textura 2D de Unity marcada como Default, un nombre y opcionalmente un tamaño de imagen.

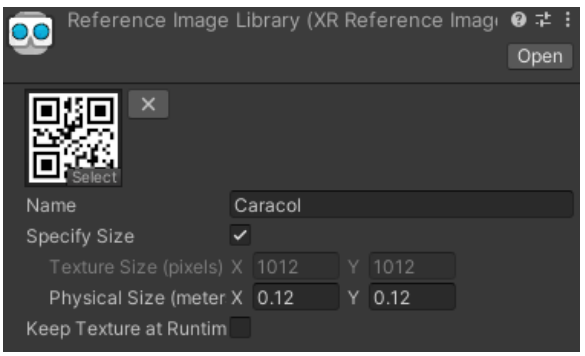

Figura 3. Librería de imágenes de AR Foundation.

Con estos simples pasos ya tendríamos una aplicación funcional. Si la aplicación detecta la imagen especificada en la librería, aparecerá un GameObject en la escena. Pero esto no significa que el trabajo ya esté terminado, la aplicación necesita detectar múltiples imágenes diferentes y cada una de estas imágenes tiene que hacer que aparezca un GameObject diferente[15]. Para ello se tiene que crear un script nuevo que haga uso de un evento del Tracked Image Manager que se llama en cada frame y contiene la información de todas las imágenes trackeables que han cambiado. La idea es tener una librería con múltiples imágenes con distintos nombres y en el script referenciar múltiples prefabs con el mismo nombre de la imagen que va a estar asociada al objeto. De esta manera se puede comparar el nombre de la imagen con el objeto a aparecer.

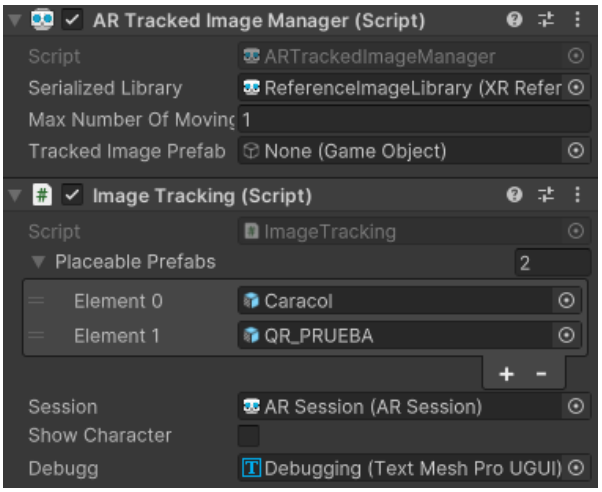

Figura 4. Componentes necesarios para el Image Tracking.

Una vez finalizada la lógica de RA se empezó a trabajar en la interfaz de usuario. El diseño de la interfaz se me proporcionó en la primera reunión presencial. La aplicación necesita de una pantalla de selector de idiomas, una intermedia para el escaneo de las imágenes y otra para la visualización del modelo 3D.

El selector de idiomas tiene una pequeña lógica para saber qué idioma está seleccionado. Consta de tres botones, si se selecciona uno de ellos va a llamar a una función que cambia el lenguaje actual por el seleccionado, por defecto, el español. Una vez seleccionado el idioma, se pasará a la ventana del escaneo de la imagen. Esta transición viene contemplada en un controlador para la interfaz de usuario, donde todas las posibles transiciones están contempladas junto con las operaciones necesarias que se tienen que cumplir para la realización de estas transiciones. Desde esta pantalla se puede salir de la aplicación.

Figura 5. Selector de idioma.

En el estado de escaneo de imágenes simplemente se necesita de un texto informativo que haga entender al usuario que tiene que escanear un código QR. Este código QR realmente no tiene ninguna información asociada y si la tuviera, no se realiza ningún tipo de operación con ella, ya que simplemente se usa como pivote para que la aplicación posicione el objeto 3D correspondiente. El texto informativo cambia su contenido para que este corresponda con el idioma seleccionado. Para ello se tiene un script que cambia el texto por el del lenguaje seleccionado cada vez que se hace un Awake al objeto. Para pasar a la siguiente ventana se tiene que detectar una imagen que el Image Tracking Manager acepte. Este estado, como dije anteriormente, está contemplado en el controlador de la interfaz de usuario y es quien AR Session Origin, desde el script Image Tracking, llama a esta transición. Desde esta pantalla se puede volver al selector de idioma pulsando el botón con forma de casita.

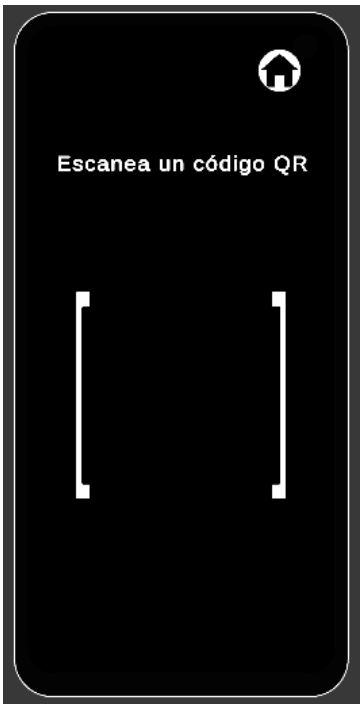

Figura 6. Escaneo de imagen.

La última pantalla de la aplicación corresponde a la vista del modelo 3D. Constituye de simplemente un texto con una breve descripción del código QR escaneado, varios botones que manejan el flujo de ejecución de la aplicación y otro botón para volver a la ventana de escaneo del código QR. El flujo de la aplicación es bastante sencilla, al escanear el código aparecerá la descripción del QR junto con un botón para iniciar el modelo, luego de pulsar el botón, el modelo 3D aparecerá por pantalla y el botón ahora tiene la opción de iniciar el audio del personaje. En este estado, el botón ahora tiene la opción de reiniciar el audio. Al igual que el resto de pantallas, el idioma del texto varía dependiendo del idioma seleccionado. La única transición posible en esta pantalla es la de volver al escaneo de código QR pulsando el botón de arriba a la derecha o esperando a que el audio del personaje se agote.

Las siguientes figuras están capturadas desde el editor de Unity, el fondo en una ejecución normal de la aplicación desde un dispositivo móvil sería el de la propia cámara del dispositivo.

Figura 7 y 8. Vista del modelo 3D.

Una vez terminada la implementación de la interfaz de usuario, se trabajó en mejorar algunas de las implementaciones, como refactorizar código o cambiar algunas características para qué se ajustará más a lo que el cliente pedía. Luego, se pasó el trabajo realizado a un compañero de trabajo. Él sería el encargado de añadir un control de animaciones del personaje. Este control de animaciones consta de un movimiento de la boca cuando se detecta un determinado tipo de onda en el audio del personaje, un parpadeo cada cierto tiempo y un movimiento arbitrario de los ojos.

Este trabajo inicial conllevó de un total de dos semanas de trabajo, del 20 al 31 de marzo.

No fue hasta el 26 de abril que no volví a trabajar en este proyecto. Mi compañero de trabajo que estaba asignado a este proyecto se iba de vacaciones, por lo que tenía que dar los últimos retoques a la aplicación. Estos retoques consistieron en añadir unas animaciones de transición entre ventanas en la interfaz de usuario y agregar un sistema para que la aplicación soportara la rotación en horizontal.

Las animaciones de transiciones no llevaron mucho trabajo. Se creó una clase nueva que controla las animaciones de *Fading In* y *Fading Out*. Estas utilizan una imagen con fondo negro que existe en la escena para hacer que esta aparezca o desaparezca dependiendo del tipo de animación. En la clase controladora de la interfaz de usuario se añadieron llamadas a estos métodos dependiendo de la transición entre las distintas pantallas.

El soporte de la aplicación para dispositivos en horizontal conlleva realizar cambios mayores en la interfaz de la aplicación. Fue necesario crear un panel nuevo para cada ventana con una disposición en horizontal y realizar una comprobación en cada frame para saber la orientación del dispositivo y activar o desactivar los paneles correspondientes.

Figura 9. Selector de idioma en horizontal.

Una vez comprobado que todo funcionaba correctamente, se procedió a realizar pequeños cambios en la interfaz como la modificación de algunos botones o ajustes en la distribución de algunos elementos.

Contando el tiempo que conllevo la corrección de algunos bugs y errores, el trabajo fue realizado entre el 26 de abril y 5 de mayo, sin contar los días festivos del 1 y 3 de mayo.

2.3.1. Desarrollo propio

Después de que finalizara mi trabajo inicial en la aplicación de RA llegó la Semana Santa, donde tuve unos días libres para realizar un proyecto de idea propia, con el objetivo de demostrar mi habilidad con AR Foundation.

La primera idea fue realizar un juego plataformas donde el usuario pudiera controlar a un personaje que se moviera por un entorno real. Para ello, se haría uso del Mesh Manager, el cual permite recrear una escena real en Unity a través del escaneado de la geometría del entorno del usuario. Por desgracia, después de unas horas de trabajo y ver que el prototipo no funcionaba, me di cuenta de que esta funcionalidad no está disponible en Android, por lo que esta idea quedó descartada.

Como segunda opción se pensó en realizar un juego del género Tower Defense utilizando RA, donde el usuario tiene que construir unas torres para defenderse de la oleada de ataques de los enemigos.

Para ello, se utilizará la funcionalidad Tracked Image para el posicionamiento de las torres. Esto da total libertad al jugador para construir su propio escenario utilizando elementos que puede encontrar en su casa. Luego, se haría uso de Plane Tracking para poder hacer aparecer enemigos en un determinado punto de la escena, por ejemplo, al pulsar en alguna zona del plano generado usando Raycast[16].

Con esta idea en mente se inició el desarrollo del prototipo. Los pasos para la configuración del Image Tracking son exactamente igual que en el proyecto anterior, crear el AR Session y AR Session Origin, asignar el componente del trackeo de imágenes al AR Session Origin y crear un script nuevo para el soporte de múltiples prefabs.

Luego sería necesario crear la lógica de las torres, las cuales básicamente tienen una determinada cantidad de vida y esta disminuye si es tocada por un enemigo. Estas se pueden defender disparando al objetivo más cercano que se encuentre dentro del rango de acción, el cual se actualiza en cada frame.

Para el manejo de planos, en AR Foundation existe el componente Plane Manager, el cual también va asignado al AR Session Origin. Este necesita de un prefab del plano a instanciar cuando se detecte una zona plana en horizontal. Con este sencillo paso ya tendríamos un detector de planos básico totalmente funcional. Luego quedaría habilitar una manera en la que los enemigos aparezcan en la escena. Dentro del script GameManager se controla el Input del usuario. Se comprueba si la pantalla está siendo pulsada, si es así, a través de un Raycast se comprueba si se está tocando unos de los planos generados por la RA.

Al ser tocado uno de los planos se creará una instancia de generador de enemigos.

Para finalizar, solo sería necesario crear una lógica para los enemigos y un spawner[17]. Los enemigos, al igual que las torres, tienen una barra de vida que disminuye cuando son alcanzados por una bala y son destruidos de la escena cuando su vida es igual o menor a cero. Su único comportamiento es el de ir a la torre más cercana para destruirla, lo cual se comprueba al crear el objeto. El spawner es una simple esfera, la cual genera enemigos cada cierto tiempo.

Los modelos de las torres y el enemigo son assets gratuitos de Unity[18][19][20].

Como líneas futuras para este pequeño proyecto:

- Crear un sistema de mejoras de las torres, donde al derrotar enemigos se obtuviera una moneda para mejorar torres existentes o desbloquear algunas nuevas.
- Crear un sistema de oleadas y progreso de los enemigos, además de añadir nuevos tipos de enemigos. Por ejemplo, después de derrotar a un cierto número de monstruos, dejar tiempos muertos para que el usuario pueda mejorar sus defensas con calma y, una vez terminado este tiempo, que aparezcan nuevamente más enemigos con estadísticas mejoradas.
- Crear un sistema que permitiera al usuario generar caminos para que los enemigos circularan por él o hacer que este camino se generara automáticamente dependiendo de la posición de las torres.
- Añadir nuevos efectos en las balas que varíen estadísticas de los enemigos como su ataque o velocidad de movimiento.

Se trabajó desde el día 3 de abril hasta el 5 de abril, a la siguiente semana se me asignó trabajar con otros dos compañeros en un proyecto nuevo.

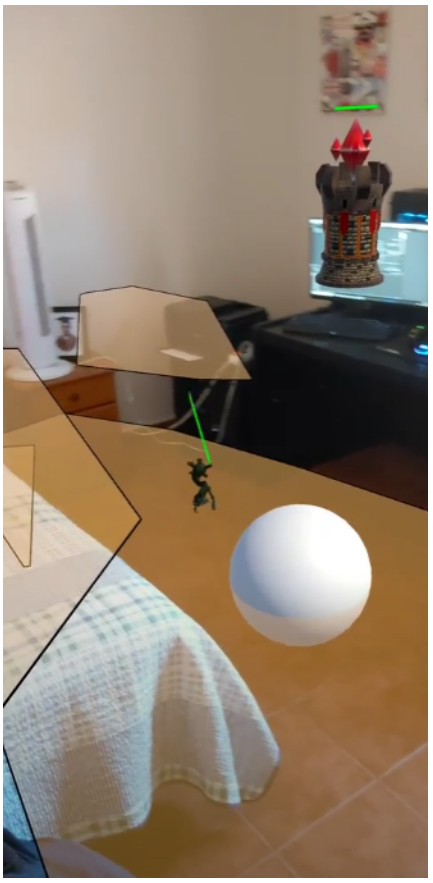

Figura 10. Frame del proyecto propio ejecutándose.

2.4.1. Proyecto ChatGPT de la empresa

Al inicio de la segunda semana de abril se me propuso ayudar en el proyecto de inteligencia artificial. Se trata de un proyecto del cual ya estaba familiarizado, ya que era un tema que se comentaba en todas las reuniones que hacíamos diariamente, por lo que comprender el trabajo que se estaba realizando y la propia incorporación en el mismo no iba a tardar más que un día.

En resumidas cuentas, el objetivo del proyecto es incorporar la inteligencia artificial ChatGPT dentro de un videojuego para que esta, a través de unas directrices que le pase un usuario, interaccione con el entorno que le rodea, además de usarlo para generar diálogos para los distintos NPC. El resultado del trabajo tiene que ser un prototipo funcional en Unity, ya que el producto final va a ser producido en Unreal Engine.

En lo que respecta a la implementación de la inteligencia artificial, yo no trabajé en ello. En cambio, la tarea que se me asignó fue diseñar e implementar una manera en la que el personaje que representa a ChatGPT pudiera examinar y usar varios objetos que se encuentran en la escena cuando la inteligencia artificial así quisiera.

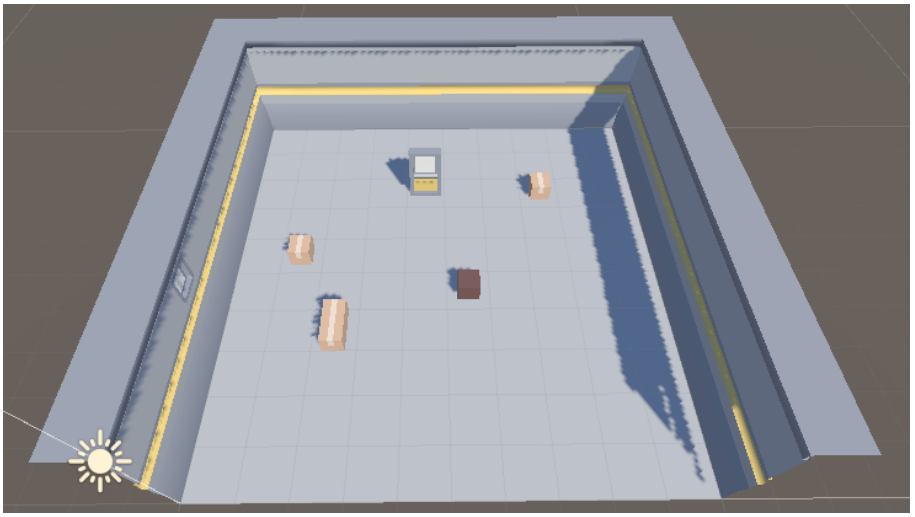

Figura 11. Entorno del prototipo ChatGPT.

La aplicación se basa en un sistema de verbos para interpretar las acciones de ChatGPT. Es decir, cuando un usuario le manda una acción a la inteligencia artificial esta responde de manera que la aplicación pueda procesar la información. Por lo que, para mi trabajo es necesario crear dos nuevos verbos, inspeccionar y usar.

Inspeccionar se usa para ver si un objeto de la escena tiene otro objeto en su interior. Con esto en mente, los objetos interactuables ahora tienen un nuevo parámetro, que corresponde al del objeto en su interior. Luego, ha sido necesario crear un nuevo script con el comportamiento del nuevo verbo. Primero hay que realizar una serie de comprobaciones para que la acción se realice, estas son comprobar si el personaje está suficientemente cerca, si el objeto con el que se está interactuando existe y si existe un objeto en su interior. Si se cumplen todas estas condiciones, se agrega el objeto al inventario del personaje y se actualiza la interfaz con la nueva información. La mayoría de estas funcionalidades ya estaban implementadas por mis compañeros de trabajo. Una vez comprobado que el verbo se comportaba de la forma esperada, se empezó a trabajar en darle un uso a estos objetos.

Para implementar el verbo usar, primero hay que determinar una manera para limitar el uso de los objetos de la escena, ya que no todos pueden ser objetos usables. Por ejemplo, las cajas de la figura 11 se pueden inspeccionar pero no usar. Para ello, se creó una nueva interfaz llamada IUsable y se le asignó a todos los objetos que tuvieran esta utilidad.

Además, es necesario crear unas acciones determinadas para cada objeto, ya que el verbo usar, a diferencia del resto, responde de diferentes maneras dependiendo del ítem a usar. Se creó una nueva clase abstracta llamada UseAction, la cual tiene que ser implementada por todas las acciones de los ítems, por ejemplo, si se tiene una llave se tiene que crear una acción para desbloquear otro objeto o si se tiene un cofre se tiene que tener una acción para abrirlo. Luego, en cada objeto interactivo, cada vez que se llama al método de la clase IUsable "Use" se llama también al método "DoAction" de la clase abstracta UseAction. De esta manera se pueden añadir y cambiar las acciones de un objeto simplemente inicializando la clase abstracta como un tipo diferente.

```
public class ObjetoLlave : ObjetoInteractuable, IUsable
{
    private UseAction useAction = null;
    public override void Use(PlayerController pController, string object2)
    {
        // En un futuro se le podria dar mas de un uso a un objeto
        Debug.Log("ESTOY USANDO LA LLAVE");
        useAction = new UnlockObject();
        useAction.DoAction(pController, this, object2);
    }
}
```

Figura 12. Implementación del objeto llave.

De momento, la aplicación tiene tres acciones de objetos implementadas, las cuales son UnlockObject, OpenObject y Throw.

Por otro lado, también ha sido necesario crear unas interfaces nuevas para el comportamiento de algunos objetos, concretamente IUnlock, para determinar si el objeto está bloqueado y se puede desbloquear con una llave, y IOpenable, para determinar si un objeto se puede abrir. Por ahora solamente la puerta implementa estas dos interfaces.

Al igual que con el verbo inspeccionar, usar necesita de un nuevo script con su comportamiento. Se tiene que comprobar si existe el objeto a usar en su inventario y si este tiene agregada la interfaz usable. Si es así, tendrá que realizar la acción del objeto. Mi aportación transcurre desde el 10 hasta el 14 de abril.

2.5.1. Prototipo de aplicación AR junto lector de QR

Mi trabajo con la inteligencia artificial se vio interrumpido el día 17 de abril cuando se nos comunicó que un potencial nuevo cliente contactó con la empresa para realizar una nueva aplicación de RA. Este nuevo proyecto tiene como objetivo mostrar un modelo 3D de un estadio y marcar la ubicación de una butaca. Este proceso se inicia cuando se escanea la información de un código QR, es decir, el QR contiene la información de la butaca a marcar en el modelo 3D del estadio.

Antes de iniciar con el trabajo, hay que encontrar una librería que maneje la información de códigos QR. Después de unas búsquedas por la red, llegué a la conclusión de que la librería con más ejemplos de uso y documentación era una versión modificada de ZXing[21]. La versión original está desarrollada en Java y distribuida para su uso principalmente en aplicaciones Java. En cambio, esta versión modificada está desarrollada en C#, por lo que su ensamblaje está disponible para .NET[22], Android, Unity, entre otros. Esta librería permite generar y decodificar códigos de barra de distintos tipos, tales como QR, que es lo que se busca para esta aplicación. Además, dispone de una documentación bastante amplia, incluyendo una versión para Unity.

En este proyecto se necesita un generador de códigos QR al cual se le pueda asignar información y también se necesita un decodificador para poder procesar esa información, para ello se hace uso de dos clases que nos ofrece la librería:

- BarcodeWriter: Clase que permite codificar información o algún tipo de contenido a un código de barras. Para ello primero se tiene que especificar el formato del código de barras a utilizar, en el caso que nos ocupa código QR, y el ancho y alto del código resultante. Como se trata de un QR, es recomendable que el ancho y alto tengan el mismo valor. Una vez especificado los distintos valores, se debe de hacer uso del método Encode, el cual devuelve el resultado en forma de una matriz de bits.
- BarcodeReader: Clase que permite decodificar la información de cualquier código de barras. Para ello necesita de hacer uso del método Decode, al cual se le tiene que pasar una matriz de bytes que representaría el código a decodificar, además de especificar el ancho y alto de la imagen junto con el formato de la misma. Para especificar el formato, ZXing usa un enum llamado BitmapFormat, en el cual viene contemplado todos los formatos que se pueden procesar, es decir, si la imagen está en escala de grises se tiene que pasar la palabra reservada Gray8 o Gray16, dependiendo de la cantidad de bytes por pixel, o si utiliza los canales rojo, verde y azul se debe de usar RGB24 o RGB32 dependiendo de la cantidad de bytes por pixel.

En lo que respecta a la creación de códigos QR, aún queda un paso fundamental por realizar, y es transformar la información de un mapa de bits a una imagen, ya que el resultado que devuelve BarcodeWriter no es suficiente como para que posteriormente el decodificador pueda escanear y comprender la información. Para ello se utiliza otra clase de ZXing llamada SvgRenderer que, como su propio nombre indica, representa un código de barras como una imagen SVG[23] haciendo uso de un método llamado Render, el cual necesita del resultado del codificador más el tipo del código de barras, devolviendo como resultado los datos necesarios para representar la imagen.

Como último paso, se hace uso de una librería experimental de Unity llamada VectorGraphics para dibujar la imagen Svg en la escena con la información que devuelve el método Render del SvgRenderer.

```
//PASOS PARA RENDERIZAR EL SVG EN UNITY
using StringReader textReader = new StringReader(result.Content);
var sceneInfo = SVGParser.ImportSVG(textReader);

//HAY QUE HACER UN TESELADO
var geometries = VectorUtils.TessellateScene(sceneInfo.Scene, new VectorUtils.TessellationOptions
{
    StepDistance = 10,
    SamplingStepSize = 100,
    MaxCordDeviation = 0.5f,
    MaxTanAngleDeviation = 0.1f
});

//                                List<Geometry>   pixelsPerUnit   alignment              pivot      resolution   flipYAxis
Sprite sprite = VectorUtils.BuildSprite(geometries,    10.0f, VectorUtils.Alignment.Center, Vector2.zero, 128, true);
GameObject sprites = GameObject.FindGameObjectWithTag("Sprite");
sprites.GetComponent<SpriteRenderer>().sprite = sprite;
```

Figura 13. Teselado de imagen Svg.

La información del código QR generado tiene una estructura fija para que la propia aplicación, en el proceso de decodificación, entienda que es un QR válido. Esta estructura es la de ASIENTOS:PUERTA:TRIBUNA:FILA:ASIENTO, siendo el único valor fijo ASIENTOS, los otros cuatro son valores numéricos únicos para cada código QR.

El escaneo de códigos se realiza desde una corrutina que se inicia en el momento que la aplicación arranca. Para ello, se necesita de la información de la cámara del dispositivo, por lo que en cada frame se comprueba su contenido haciendo uso de una clase de Unity llamada WebCamTexture[24]. Esta información se sigue obteniendo hasta que el decodificador obtiene algún resultado de la imagen del dispositivo. Si el contenido del código leído sigue el formato que comenté anteriormente, la aplicación prosigue sin problemas.

El manejo de la información de las butacas es sencillo, cada butaca tiene su propia puerta, tribuna, fila y asiento asignada desde el editor de Unity, mientras que la del usuario se obtiene en tiempo de ejecución al leer el código QR. El principal problema de este método es que si se tiene un estadio con mil butacas se tendría que añadir la información una a una, lo cual es inviable, pero para este prototipo se ha buscado optimizar el manejo de la información de los códigos QR y saber si una implementación junto con AR Foundation es posible.

La implementación de la realidad aumentada en esta aplicación es un tanto distinta. En los otros dos proyectos, se necesita de múltiples prefabs, todos ellos asociados a una imagen específica. Mientras en este proyecto solo hay un prefab disponible, el del estadio, el cual debe de ser trackeado sobre cualquier código QR válido para la aplicación. Crear todos los posibles códigos de antemano y añadirlos a la librería no es una opción, por lo que es necesario añadir la imagen en tiempo de ejecución.

AR Foundation dispone de un tipo de librería que puede ser modificada en tiempo de ejecución, pero no todos los dispositivos permiten esta funcionalidad. En este proyecto, se añade a XR Origin un Tracked Image Manager en tiempo de ejecución y no se añade ninguna imagen a la librería hasta que el lector de código de barras escanee un código QR. Este proceso también se realiza en una corrutina, ya que es un proceso que necesita esperar por una textura 2D a añadir a la librería.

El último tema a abordar sería el método de búsqueda de la butaca del usuario entre todas las butacas disponibles. Por ahora se comparan los valores del usuario con cada butaca y cuando se encuentra con una información idéntica, la marca como encontrada y rompe el bucle. Este ejemplo tiene el mismo problema que comente anteriormente, muchos asientos hacen que el proceso de búsqueda sea largo y costoso.

En resumen, el flujo de ejecución de la aplicación se resume en dos estados, siendo el primero el de lectura del código QR y el segundo siendo la muestra de la búsqueda de la butaca.

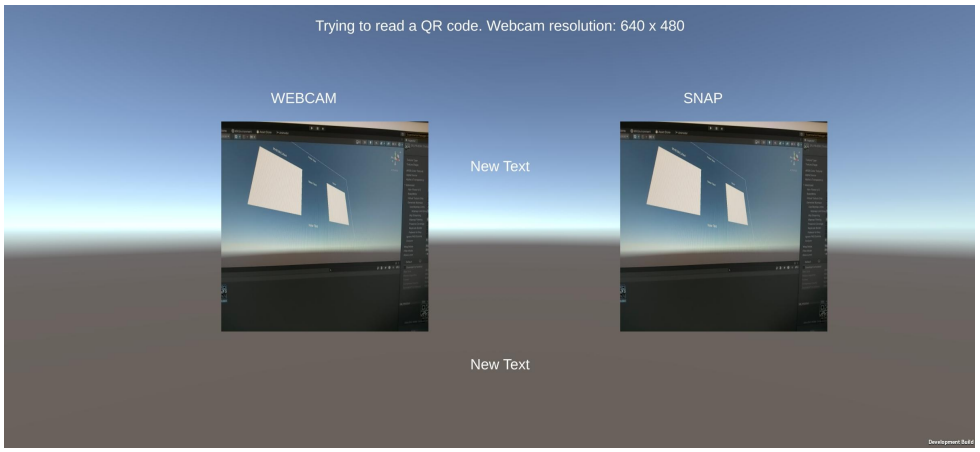

Figura 14. Pantalla de escaneo de QR.

La anterior figura corresponde a un frame de la ejecución de la aplicación en un dispositivo móvil, donde se está intentando escanear un código QR. La imagen de la izquierda corresponde a los datos que se están obteniendo de la cámara del dispositivo y la de la derecha a los datos que se están pasando al escáner de códigos QR.

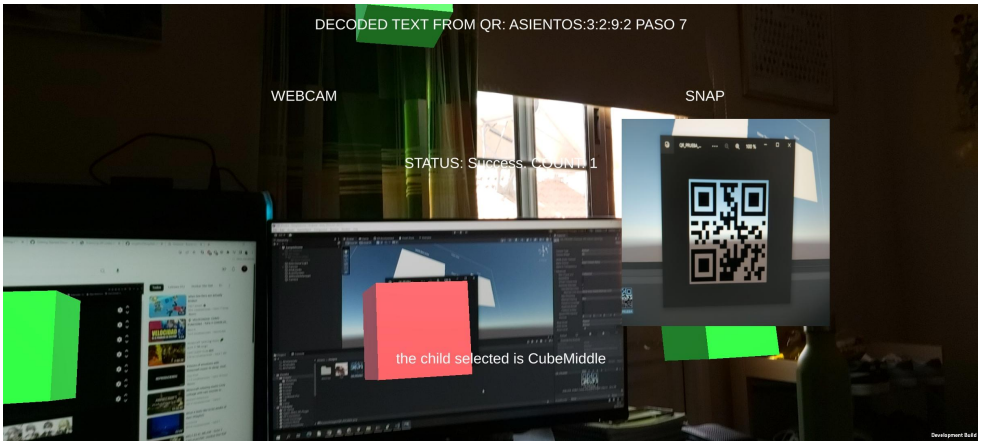

Figura 15. Pantalla con el resultado del escaneo.

Esta última figura corresponde al escaneo de un código QR válido para la aplicación. En este caso, se han leído los datos correspondientes a que se marque la butaca central del estadio. Se puede observar que la imagen de la izquierda ha desaparecido, ya que se ha dejado de usar los datos de WebCamTexture y ahora se hace uso de los datos de la cámara de RA. La imagen de la derecha corresponde a la imagen que se ha añadido a la librería de Tracked Image Manager para hacer como pivote del estadio.

Fue un trabajo de prueba y error, ya que son muchas las maneras de las que se puede abordar este tema. Por ejemplo, al principio recogía la información de la cámara del dispositivo a través de una función llamada Graphics.Blit, pero en práctica ralentizaba mucho el resultado final, por lo que opté por seguir con la WebCamTexture. Por tanto, el trabajo empezó el día 17 de abril y terminó el 25 de abril.

2.2. Metodología utilizada

En los proyectos que colaboré trabajando de manera solitaria, tenía libertad absoluta de llevar el ritmo que quisiera mientras entregara los resultados dentro de unos plazos de tiempo razonables. Además, podía trabajar de la manera que me sintiera más cómodo, por lo que nunca me sentí presionado ni agobiado. Por otro lado, el proyecto que trabajé en grupo fue con dos compañeros que me ayudaron a comprender rápidamente cómo se organizaban y repartían las tareas, por lo que mi incorporación se llevó a cabo de manera exitosa y rápidamente pude seguir con el trabajo.

Al final de cada jornada laboral, se realizaba una reunión con el tutor de la empresa y el resto de compañeros de trabajo. Normalmente, el tutor me daba paso a mi primero para hablar, donde comentaba el trabajo realizado durante el día, si había tenido algunos inconvenientes y como los pude solucionar y la planificación para el día siguiente. Luego daba turno para el resto de compañeros, donde principalmente se comentaba sobre la aplicación de inteligencia artificial. Estas reuniones tenían una duración de alrededor de 30 minutos.

Respecto a la comunicación fuera de las reuniones de trabajo, se disponía de un grupo de WhatsApp que principalmente se usaba para organizar trabajos y enviar documentación y archivos para la realización de ciertos proyectos, además de poder comentar alguna duda o problema puntual. También existe un grupo de Discord que usan principalmente para el proyecto de inteligencia artificial, donde hay habilitados varios canales de texto para comentar distintos temas, además de un canal de voz.

2.3. Cronograma

- Inicio del trabajo: 16 marzo
- Estudio de optativas de realidad aumentada para Unity: 17 - 20 marzo.
- Implementación de la lógica RA: 20 - 24 marzo.
- Implementación de la lógica UI: 25 - 31 marzo.
- Desarrollo de prototipo juego propio RA: 3 - 5 abril.
- Aprendizaje aplicación ChatGPT: 10 abril.
- Desarrollo del verbo inspeccionar: 11 abril.
- Desarrollo del verbo usar y acciones de objetos: 12 - 14 abril.
- Estudio del uso de ZXing: 17 - 19 abril.
- Desarrollo de prototipo con ZXing: 19 - 21 abril
- Incorporación de RA en el prototipo ZXing: 21 - 25 abril.
- Agregar animaciones y mejorar de UI en la aplicación RA: 26 - 28 abril.
- Incorporar soporte horizontal aplicación RA: 2 - 5 mayo.
- Trabajos pequeños y preparar aplicación RA para build: 8 - 12 mayo.

Capítulo 3
Aportaciones

Este capítulo consta de una valoración final de mi trabajo realizado en la empresa, junto con una descripción de varios problemas detectados y propuestas a solucionar.

3.1. Problemas detectados y propuestas de solución

Los días que conllevaron más problemas asociados fueron los días que se estudiaba el uso de nuevas herramientas para las aplicaciones, como el estudio y prueba de un prototipo de RA y la incorporación y uso de la librería ZXing en un proyecto. Normalmente, la solución se buscaba de forma autónoma, indagando información en internet o realizando más pruebas siguiendo un estilo de prueba y error hasta encontrar el resultado esperado. En lo que respecta al trabajo con la inteligencia artificial, como ya tenía experiencia en Unity gracias a las clases del máster y mi parte no iba más que añadir funcionalidades básicas al avatar y no trabajar con la implementación de ChatGPT en Unity, no hubo grandes contratiempos en lo que respecta a la implementación de interfaz de usuario o desarrollo de código con elementos de Unity.

Simplemente por comentar dos problemas, los cuales considero que son los que me llevaron más tiempo solucionar, serían el que me surgió con el desarrollo del prototipo propio y con el añadido de imágenes en tiempo de ejecución a una librería de Tracked Images.

Durante el desarrollo propio, se tenía pensado realizar un juego de aventuras, donde el jugador pudiera mover un avatar por un entorno generado por AR Mesh, lo cual me llevó media jornada implementar la lógica para luego ver que nada funcionaba. La otra media jornada fue para darme cuenta de que era una característica que no soportaba ni mi Android ni el entorno de pruebas de Unity, por lo que al final la única solución fue empezar de cero y perder un día entero de trabajo.

Respecto al añadido de imágenes en tiempo ejecución, fue realmente una tontería, ya que estaba realizando el trabajo en el entorno de pruebas de Unity para evitar hacer build tras build en un dispositivo móvil y así optimizar el tiempo de trabajo. Resulta que el añadido de imágenes en tiempo de ejecución no se puede hacer en el entorno de pruebas, y descubrir el porqué la aplicación no me funcionaba, cuando realmente sí lo estaba haciendo, solo que tenía que probarlo en un móvil Android, me llevó un día entero de trabajo.

3.2. Valoración personal

Ha sido una experiencia novedosa y gratificante, ya que nunca antes había trabajado en una empresa relacionada con el mundo de los videojuegos. Además, agradezco que se me hayan asignado tareas con bastante importancia y se me haya dado libertad de trabajar como me sintiera más a gusto. También, el trato del resto de compañeros de trabajo conmigo siempre ha sido amigable y en todo momento se me habló desde el respeto, por lo que el ambiente de trabajo era ideal. De manera general, no podría haber ido mejor.

Conclusiones y Líneas futuras

Como resultado del trabajo realizado se han desarrollado varias aplicaciones en realidad aumentada con distintos objetivos, haciendo uso de varias funcionalidades como son el seguimiento de imágenes o el detector de planos. También se han incrementado los conocimientos de C# al ayudar en el prototipo de inteligencia artificial. Como resumen final, estas son las tres principales herramientas que definen mi periodo de prácticas:

- Manejo de realidad aumentada con AR Foundation.
- Desarrollo de código C# en Unity.
- Generación y decodificación de códigos QR con ZXing.

Llegué a comprobar que la realidad aumentada es una tecnología que, gracias a la capacidad de superponer datos digitales sobre el mundo real, ayuda a divulgar información de manera óptima y hace que el usuario la retenga mucho mejor que si la estuviera leyendo desde un texto plano, además de mejorar la experiencia del usuario en general.

Respecto a los objetivos planteados al inicio de la presente memoria, todos ellos han sido cumplidos, incluso los que se fueron añadiendo al ir avanzando con el desarrollo de los distintos proyectos. Cada objetivo planteaba unos nuevos retos, pero gracias al esfuerzo y el apoyo de los compañeros de trabajo a través de las reuniones diarias, ninguno de estos retos fue imposible de superar.

Con relación a las líneas futuras, cada uno de los distintos proyectos podría considerar nuevas características y funcionalidades:

- Aplicación RA del caracol: Mejorar el diseño de la interfaz de usuario serían los siguientes pasos a seguir ya que añadir nuevas funcionalidades es responsabilidad del cliente. En estos momentos, la interfaz utiliza un diseño para los botones básico y la fuente de las letras es la que viene por defecto en Unity, por lo que se le debe de dar un estilo más distintivo. Además, se podría considerar cambiar la distribución de ciertos elementos para que estos sean más amigables para el usuario.
- Prototipo de Tower Defense: Están comentados en el capítulo correspondiente. En resumidas cuentas, se necesita pulir la interacción del usuario y añadir nuevas funcionalidades para dar una sensación de progreso al videojuego, por ejemplo añadiendo un sistema de mejoras de las torres o un sistema de oleadas que varíen la cantidad de enemigos y sus estadísticas.

- Prototipo ChatGPT: Añadir nuevos verbos para que la inteligencia artificial pueda interactuar con más elementos del escenario. Por ejemplo, después de que dejara de ayudar en el prototipo, mis compañeros se pusieron a trabajar en añadir interacción con varios NPC de la escena a través del verbo “hablar” y añadieron un nuevo verbo llamado “extinguir” para poder interactuar con unas llamas y, si el avatar tiene un extintor en el inventario, apagarlas.
- Prototipo RA con códigos QR: Diseñar un nuevo método para almacenar la información de las butacas del estadio. Por ahora, el modelo solo tiene 5 butacas, por lo que añadir y modificar su información una a una no es un problema. Pero en un caso real un estadio puede tener hasta 20.000 butacas, haciendo que la modificación individual sea totalmente inviable. Se podría hacer uso de alguna función matemática o asignación en bucle si el estadio sigue un patrón a la hora de asignar un número a las distintas butacas.

Summary and Conclusions

As a result of the work done, several augmented reality applications have been developed with different objectives, making use of several functionalities such as image tracking or plane detector. I have also increased my knowledge of C# by helping with the artificial intelligence prototype. As a final summary, these are the three main tools that define my internship:

- Management of augmented reality with AR Foundation.
- C# code development in Unity.
- QR code generation and decoding with ZXing.

I came to see that augmented reality is a technology that, thanks to the ability to overlay digital data on top of the real world, helps to spread information in an optimal way and makes the user retain it much better than if they were reading it from plain text, as well as improving the overall user experience.

With respect to the objectives set at the beginning of this report, all of them have been met, including those that were added as the different projects progressed. Each objective posed new challenges, but thanks to the effort and support of my coworkers through daily meetings, none of these challenges was impossible to overcome.

Regarding future lines, each of the different projects could consider new features and functionalities:

- Snail AR application: Improving the design of the user interface would be the next step to follow since adding new functionalities is the client's responsibility. At the moment, the interface uses a basic button design and the font of the letters is the default font in Unity, so it should be given a more distinctive style. Also, we could consider changing the layout of certain elements to make them more user-friendly.
- Tower Defense prototype: These are discussed in the corresponding chapter. In short, it is necessary to polish the user interaction and add new features to give a sense of progress to the game, for example by adding a system of tower upgrades or a wave system that varies the number of enemies and their statistics.

- ChatGPT Prototype: Adding new verbs so that the artificial intelligence can interact with more elements of the scene. For example, after I stopped helping in the prototype, my coworkers got to work on adding interaction with several NPCs in the scene through the verb "talk" and added a new verb called "extinguish" to be able to interact with some flames and, if the avatar has an extinguisher in the inventory, extinguish them.
- AR prototype with QR codes: Design a new method to store the information on the stadium seats. For now, the model has only 5 seats, so adding and modifying their information one by one is not a problem. But in a real case, a stadium can have up to 20,000 seats, making individual modification totally unfeasible. Some mathematical functions or loop assignments could be used if the stadium follows a pattern when assigning a number to the different seats.

Bibliografía

[1] TenerifeJuega. https://www.tenerifejuega.com/

[2] Escuela Artísticas de Los Realejos. https://escuelasartisticas.com/

[3] Image Tracking AR Foundation. https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/tracked-image-manager.html

[4] ZXing Java. https://github.com/zxing/zxing

[5] Plane Tracking AR Foundation. https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/plane-manager.html

[6] Chat GPT. https://openai.com/blog/chatgpt

[7] AR Foundation. https://unity.com/es/unity/features/arfoundation

[8] ARKit. https://developer.apple.com/augmented-reality/

[9] ARCore. https://developers.google.com/ar?hl=es-419

[10] Magic Leap. https://www.magicleap.com/en-us/

[11] HoloLens. https://www.microsoft.com/es-es/hololens

[12] AR Foundation documentación. https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html

[13] AR Session. https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.2/api/UnityEngine.XR.ARFoundation.ARSession.html

[14] AR Session Origin. https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/api/UnityEngine.XR.ARFoundation.ARSessionOrigin.html

[15] AR Tracked Image Manager Component. https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/features/image-tracking.html

[16] AR Raycast Manager.
https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.0/manual/raycast-manager.html

[17] Spawning (video games).
https://en.wikipedia.org/wiki/Spawning_(video_games)

[18] Canon Tower.
https://assetstore.unity.com/packages/3d/environments/fantasy/canon-tower-50215

[19] Awesome Stylized Mage Tower.
https://assetstore.unity.com/packages/3d/environments/fantasy/awesome-stylized-mage-tower-53793

[20] Alien Character.
https://assetstore.unity.com/packages/3d/characters/creatures/alien-character-20838

[21] ZXing Unity. https://github.com/micjahn/ZXing.Net

[22] .NET. https://dotnet.microsoft.com/es-es/learn/dotnet/what-is-dotnet

[23] Formato de imagen SVG.
https://es.wikipedia.org/wiki/Gr%C3%A1ficos_vectoriales_escalables

[24] WebCamTexture Unity.
https://docs.unity3d.com/ScriptReference/WebCamTexture.html