Escuela Superior de Ingeniería

DESARROLLO DE SISTEMAS HIPERMEDIA

Grado en ingeniería informática

AR Party: Entretenimiento para el confinamiento

Curso 2019-2020

Autores:
Germán Ramírez Lerate
Pablo de los Ríos Gestoso

Cádiz, 3 de junio de 2020

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

En tiempos de confinamiento, hemos aprendido que estar unidos es algo fundamental. Por ese motivo, hemos querido desarrollar **AR Party**: una sala de juegos de realidad aumentada para toda la familia, donde todos podrán interactuar con un mundo virtual sin necesidad de tecnología específica, a excepción de móviles con cámara y utensilios que todos podemos tener en casa, como libros o cuadros.

Esta aplicación dispondrá de una serie de minijuegos, para uno y para dos jugadores, donde se deberá hacer uso de agilidad física y mental, además de fomentar el espíritu competitivo y, por supuesto, pasar un buen rato en familia.

1.2. Objetivos y alcance del proyecto

El objetivo principal de este proyecto es desarrollar una aplicación Android para móvil, empleando Unity y, en concreto, el motor de realidad aumentada **Vuforia**. Se van a implementar tres minijuegos que constituirán el núcleo de la aplicación:

1. **The Virtual Maze**. Este juego para un solo jugador despliega ante el usuario un laberinto virtual entre cuyas paredes se encuentra encerrada una pelota. El objetivo del juego es ir girando e inclinando el laberinto para trasladar la pelota hacia la salida del laberinto antes de que finalice el tiempo.

2. **Destroy The Planes**. Este juego multijugador permitirá a dos jugadores enfrentarse a un ejército de bombarderos que volarán por el espacio de la casa. Los jugadores deberán destruir todos los aviones que puedan en un límite de tiempo. Aquel jugador que destruya más aviones, se alzará como ganador.

3. **Home for Speed**. Este juego multijugador permitirá a dos jugadores realizar una carrera de coches a través de un circuito virtual. Aquel jugador que llegue antes a la meta, se alzará como vencedor de la carrera.

Para la consecución de estos objetivos se deberá implementar, además, un sistema capaz de dar soporte a los niveles multijugador y un sistema que permita al usuario definir los

targets o etiquetas que va a usar como referencia para los objetos virtuales. Asimismo, se implementará una interfaz intuitiva y usable que permita hacer uso de todos los requisitos funcionales especificados.

1.3. Repositorio de GitHub

Todo nuestro proyecto se encuentra disponible en el siguiente repositorio de GitHub: `https://github.com/AR-Party/AR-Party-Unity-Project`.

Capítulo 2

Estado del Arte

En este capítulo, nos disponemos a describir las tecnologías específicas empleadas para el desarrollo del proyecto.

2.1. Photon Engine

Para el desarrollo de los videojuegos multijugador necesitábamos una infraestructura capaz de sincronizar los elementos de la escena entre los distintos jugadores. Para ello, Unity ofrece, descargándolo desde su Asset Store, la posibilidad de usar **Photon Engine** [2].

Photon Engine es un software de servidores locales que se ejecutan en lugares de todo el mundo, y que permiten el desarrollo de juegos multijugadores de baja latencia.

Permite reunir jugadores con una sesión de juego compartida y transferir mensajes de manera síncrona, en tiempo real, entre dichos jugadores. Todos los SDK del cliente pueden interactuar entre sí, sin importar si se trata de Android, iOS o consola.

2.2. Targets dinámicos

Como hemos mencionado anteriormente, se ha usado el motor de realidad aumentada **Vuforia**. Este motor permite una integración relativamente simple con Unity, y está pensado para interactuar con él mediante el uso de targets definidos en tiempo de compilación, es decir, facilita el desarrollo de aplicaciones de realidad aumentada con targets (de imagen, tridimensionales, VuMarks, etc) definidos a priori, siendo estos almacenados en ficheros o mediante los servicios cloud de Vuforia.

Dicho esto, Vuforia también permite el uso de targets definidos en tiempo de ejecución, pero esta sección de la API carece de una integración transparente en el editor de Unity, a diferencia del resto de funcionalidades que el motor ofrece.

La carencia de dicha integración nos forzó a documentarnos extensamente en el funcionamiento del motor con respecto a la creación de targets. Esta tarea resultó más compleja de lo esperada debido a la falta de documentación oficial, además de escasos foros discutiendo nuestra situación y, los pocos que encontrábamos, con información desactualizada.

Capítulo 3

Planificación

Tras presentar el contexto en el que se va a desarrollar este proyecto y las tecnologías empleadas con este objetivo, nos disponemos a exponer la metodología seguida, así como la planificación del proyecto mediante diagramas de tiempo y la distribución de tareas entre los componentes del grupo.

3.1. Metodología de desarrollo

Para la realización de este proyecto, hemos elegido una metodología de desarrollo iterativo. Según la guía *Software Engineering Body of Knowledge* (SWEBOK), la metodología iterativa es un "*modelo en el que el software se desarrolla incrementando las funcionalidades en ciclos iterativos*" [1]. De esta forma, al desarrollar ciertas funcionalidades en cada iteración, podremos realizar las correspondientes pruebas sobre estas y corregir sus defectos antes de añadir nuevas funcionalidades a la aplicación. Asimismo, esta metodología facilitará el mantenimiento y la escalabilidad de la aplicación para un trabajo futuro.

3.2. Planificación temporal

Las distintas etapas de desarrollo del proyecto requieren una planificación temporal apropiada, que permita cumplir con los distintos objetivos del proyecto en una serie de plazos factibles. Por tanto, se ha utilizado un diagrama de Gantt para gestionar, de manera sencilla y visual, la planificación temporal del proyecto.

En la Figura 3.1 se aprecia el diagrama de Gantt que recoge la planificación temporal del proyecto. En líneas generales, como se puede observar, se comenzó a investigar desde que se planteó la realización del proyecto hasta muy avanzado este, cuando la mayoría de los elementos novedosos que pretendíamos incorporar ya estaban desarrollados. Asimismo, se puede observar que lo primero que se ha desarrollado ha sido la infraestructura para los juegos multijugador y el juego del laberinto. A la mitad del proyecto, se han desarrollado el shooter, la interfaz y los targets añadidos dinámicamente, y al final del proyecto se ha procedido con el último juego. La elaboración de la documentación ha estado presente desde las primeras etapas del proyecto, una vez que se había investigado lo suficiente.

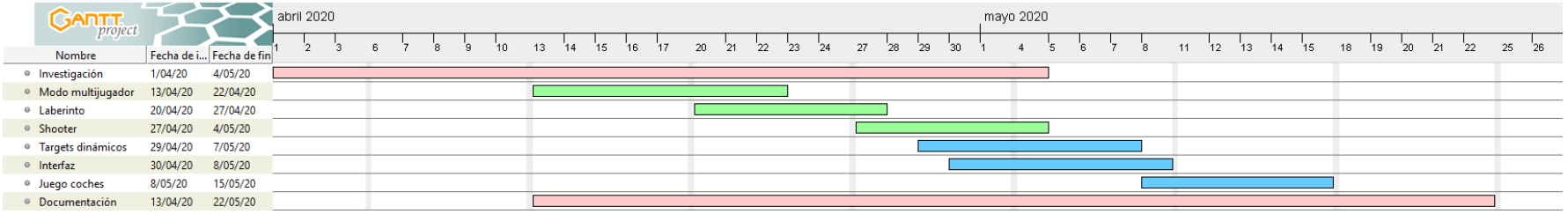

Figura 3.1: Diagrama de Gantt del proyecto

3.3. Distribución de tareas

Las tareas necesarias para la consecución de los objetivos del proyecto se han distribuido de forma equitativa entre los componentes del proyecto. En general, y basándonos en la planificación temporal presentada, cada miembro del proyecto ha tenido una serie de objetivos clave asignados, al tiempo que ha ayudado al otro miembro del grupo en aspectos concretos de sus objetivos, si es necesario. De esa forma, aunque cada miembro tiene sus tareas asignadas, sabe en todo momento qué hace el otro miembro y, sobre todo, cómo lo ha hecho.

En general, la distribución de tareas es la siguiente:

- **Pablo de los Ríos Gestoso**:
  - Desarrollo e implementación del diseño y organización de la interfaz de usuario.
  - Desarrollo e implementación del uso de targets dinámicos.
  - Desarrollo e implementación del minijuego *Home for Speed*.
- **Germán Ramírez Lerate**:
  - Desarrollo e implementación de la infraestructura multijugador.
  - Desarrollo e implementación del minijuego *The Virtual Maze*.
  - Desarrollo e implementación del minijuego *Destroy The Planes*.

Capítulo 4

Implementación del juego

En este capítulo, detallamos aspectos de la implementación de nuestra aplicación, como los elementos propios del Asset Store que se han utilizado.

4.1. Elementos del Asset Store usados

Para el desarrollo del proyecto se han empleado los siguientes elementos del Asset Store:

- **Photon Unity Networking 2** [**3**]. También conocido como PUN 2, es la implementación gratuita de Photon para usar callbacks, RPCs y, en general, los mecanismos de sincronización de objetos del juego.
- **ARCADE - Free Racing Car** [**4**]. Paquete que ofrece una serie de modelos de coches predefinidos, distinguiendo cada una de sus partes (ruedas, chasis,...). De estos modelos hemos usado dos, a los cuales le hemos añadido la capacidad de ser controlados por el usuario. Este paquete se ha empleado para el juego `Home for Speed`
- **Modular Track** [**5**]. Paquete que ofrece piezas para construir un circuito de carreras, para el juego `Home for Speed`. Estas piezas tuvieron que ser modificadas para optimizar el rendimiento, además de añadir barreras para que los coches permaneciesen dentro el circuito.
- **Awesome Cartoon Airplanes** [**6**]. Paquete que ofrece una variedad de aviones, para el juego *Destroy The Planes*.
- **Native Camera** [**7**]. Paquete que ofrece la posibilidad de acceder a la cámara del dispositivo móvil. Esto se ha usado para realizar la captura del target en tiempo de ejecución.

4.2. Elementos Externos usados

Para el desarrollo del proyecto se han empleado los siguientes elementos descargados de la web:

- **Mobile Joystick** [8]. Paquete que ofrece controles táctiles como botones y joysticks. Usado en el juego *Home for Speed*.

Capítulo 5

Conclusiones

En este capítulo final, nos disponemos a reseñar los problemas que hemos encontrado durante el desarrollo de este proyecto, además de indicar las posibles mejoras que se podrían realizar sobre la aplicación y el trabajo futuro que se podría llevar a cabo.

5.1. Problemas encontrados

Entre los problemas con los que nos hemos encontrado, destacamos los siguientes:

- Como comentamos al comienzo del documento en cuanto a la realización del target dinámico, hemos encontrado escasa documentación disponible a través de la web, y la poca que había se encontraba desactualizada. Esto nos ha conducido a experimentar por nuestra cuenta y a toparnos con múltiples errores, lo cual ha conllevado pérdidas de tiempo y recursos. Finalmente nos ha llevado a crear una clase propia que controla la creación de los targets con un dataset dinámico, la cual esta diseñada para poder *"soltar"* en el editor y funcionar de manera similar al elemento *ImageTarget* de Vuforia.
- En relación al target dinámico, la implementación del mismo es volátil, es decir, sólo se mantiene durante una sesión de juego. No hemos hallado la manera de lograr persistencia del target al cerrar la aplicación.
- Un gran problema relativo a los targets fue también la perdida de tracking, lo cual pudimos solucionar parcialmente con el tracking extendido de Vuforia. Pero esto no era una solución para elementos que usasen físicas (coches, laberinto, etc) ya que al perder el tracking, estos caían al vacío. Finalmente llegamos a una solución modificando la clase por defecto que controla el comportamiento de los targets, la cual desactivaba los rigidbodies al perder tracking.
- En cuanto al juego *Home for Speed*, hemos hallado cierta dificultad para adaptar la física con el objetivo de mover el coche correctamente en la realidad aumentada.
- También, en relación al mismo juego, hemos encontrado problemas para lograr una sincronización óptima de los coches en ambos dispositivos.
- Nos hemos llevado tiempo para establecer la lógica para contar las vueltas de un coche en una única dirección del circuito.

- La implementación de la interfaz ha requerido múltiples iteraciones a medida que se iba comprendiendo mejor el funcionamiento de los elementos UI de Unity.
- En cuanto a la infraestructura multijugador, se ha perdido mucho tiempo resolviendo errores con los servidores de Photon.
- Para el juego *Destroy the Planes*, resultó problemática la sincronización de los aviones en la misma vista para ambos jugadores (algo similar al juego *Home for Speed* pero con una cantidad mucho mayor de elementos). Inicialmente fue un problema a solventar pero tras un estudio de jugabilidad, vimos que era más interesante tener un set de aviones por cada jugador, ya que si ambos comparten el mismo set, estos necesitaban ser generados demasiado rápido para no dejar a los jugadores esperando a que se generasen más o *çampeando"* en el punto de creación de aviones.
- El juego *The Virtual Maze* aún genera problemas de estabilidad del target, el cual pierde en ocasiones la detección incluso con el tracking extendido y el tablero vibra descontroladamente, algo que no ocurre con el cubo de prueba. Creemos que puede deberse al gran número de paredes del laberinto, ya que se ha creado en unity mediante un cubo por cada pared y esto puede estar afectando a la estabilidad.

5.2. Trabajo futuro

En cuanto a las mejoras y al trabajo futuro de la aplicación, podemos destacar:

- Añadir mayor variedad de minijuegos para que la experiencia del usuario resulte más satisfactoria y pueda dedicar mayor número de horas a nuestra aplicación.
- Lograr una mejor sincronización de los objetos en los juegos multijugador.
- Lograr un mayor refinamiento estético de los juegos, añadiendo efectos visuales (explosiones de los aviones en *Destroy The Planes* o marcas de las llantas al derrapar en *Home for Speed*) y efectos sonoros (ruidos de motor en *Home for Speed*).
- Intentar optimizar el laberinto creándolo desde cero en un software 3D como Blender.

Bibliografía

[1] P. Bourque, R.E. Fairley, eds., *Guide to the Software Engineering Body of Knowledge, Version 3.0*, IEEE Computer Society, 2014. www.swebok.org.

[2] Photonengine.com. *Multiplayer Game Development Made Easy: Photon Engine*. https://www.photonengine.com/en-us/Photon. [Accedido el 19-05-2020].

[3] assetstore.unity.com. *PUN 2 - FREE*. https://assetstore.unity.com/packages/tools/network/pun-2-free-119922. [Accedido el 31-05-2020].

[4] assetstore.unity.com. *ARCADE: FREE Racing Car*. https://assetstore.unity.com/packages/3d/vehicles/land/arcade-free-racing-car-161085. [Accedido el 31-05-2020].

[5] assetstore.unity.com. *Modular Track*. https://assetstore.unity.com/packages/3d/environments/modular-track-85356. [Accedido el 31-05-2020].

[6] assetstore.unity.com. *Awesome Cartoon Airplanes*. https://assetstore.unity.com/packages/3d/vehicles/air/awesome-cartoon-airplanes-56715#content. [Accedido el 31-05-2020].

[7] assetstore.unity.com. *Native Camera for Android & iOS*. https://assetstore.unity.com/packages/tools/integration/native-camera-for-android-ios-117802. [Accedido el 31-05-2020].

[8] youtube.com. *Easy Mobile Input in Unity - MMAG 5*. https://www.youtube.com/watch?v=zHvp2PN5NIQ. [Accedido el 31-05-2020].