
Control Remoto de Videojuegos con Smartphones



MEMORIA DE TRABAJO DE FIN DE GRADO

Pablo Gómez Calvo
Sergio J. Higuera Velasco

Grado de Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

Abril 2020

Control Remoto de Videojuegos con Smartphones

Memoria de Trabajo de Fin de Grado
Grado de Desarrollo de Videojuegos
Abril 2020

Director: Carlos León Aznar
Director: Pedro Pablo Gómez Martín

Versión 0.3

Grado de Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

Abril 2020

Copyright © Pablo Gómez Calvo y Sergio J. Higuera Velasco

A todo aquel que confió en nosotros

Agradecimientos

Nadie es innecesario.

Yitán, Final Fantasy IX

El primer agradecimiento hay que dárselo a la Universidad Complutense por aceptar la creación de este grado, un grado que demuestra la importancia del mundo de los videojuegos en la sociedad actual. Con este grado se han conseguido romper muchas barreras, entre ellas está poder especializarse y adoptar los videojuegos como nuestra profesión.

Dar gracias a los profesores que nos han acompañado estos años y que han contribuido en el desarrollo del grado. Una mención aparte para las dos personas que han hecho posible la realización de este Trabajo de Fin de Grado, Carlos León Aznar y Pedro Pablo Gómez Martín.

Nuestro último agradecimiento va dirigido a nuestras familias por soportarnos en todos nuestros momentos durante el grado.

Resumen

¡No estáis preparados!

Illidan Tempestira, World of Warcraft

Índice

Agradecimientos	VII
Resumen	IX
1. Introducción	1
1.1. Introducción	1
2. Estado del arte	3
2.1. Herramientas para el desarrollo	3
2.1.1. Unity	3
2.1.2. Android Studio	4
2.2. Librerías usadas	5
2.2.1. ZXing	5
2.2.2. Unity	5
2.2.3. Android	5
2.3. Proyectos Similares	5
2.3.1. Wii U	6
2.3.2. GamePad	6
2.3.3. PlayLink	6
3. Objetivos y especificación	7
3.1. Objetivos	7
3.2. Plan de trabajo	7
3.3. Metodología	8
3.4. Herramientas utilizadas	8
4. Arquitectura	11
4.1. Arquitectura General	11
4.2. Protocolo de conexión	12
4.3. El uso de los QR para inicio de conexión	12
4.4. Pantalla secundaria y controlador	12

5. Desarrollo de la aplicación de Android	15
5.1. Manejo de actividades	15
5.2. Interfaz del modo mando	16
5.3. Manejo de hilos de ejecución	16
6. Desarrollo de la aplicación de Unity	19
6.1. Establecimiento de conexión	19
6.2. Recepción de Input	20
6.3. Envío de imágenes por red	20
7. Conclusiones	23
7.1. Conclusiones Pablo	23
7.2. Conclusiones Sergio	23

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Introducción

La industria del videojuego desde su inicio nos ha enseñado que la innovación a la hora de crear experiencias nuevas para los usuarios es algo que enriquece a muchos jugadores, por ende se está ayudando a que la experiencia de juego sea cada vez más cómoda y flexible para los jugadores.

Es por esta razón que empresas como Electronic Arts, Ubisoft, Kunos Simulazioni y Polyphony Digital, entre otras, dedican gran cantidad de sus recursos a hacer realidad muchas experiencias que los usuarios quieren tener como, por ejemplo, jugar a juegos deportivos realistas como en FIFA o conducir automoviles de competición por un circuito famoso como en Assetto Corsa.

Pero para invertir en crear este nuevo tipo de estilos de juego se necesita una gran financiación, cosa que estudios pequeños no tienen. Estos estudios independientes usan motores como Unity3D o Unreal Engine 4. Estos motores, cuyo pago es porcentual a las ganancias obtenidas por tu juego o en algunos casos mensual, ofrecen una gran cantidad de herramientas a disposición de sus usuarios para que los desarrolladores puedan ahorrar tiempo de implementación de una nueva característica y lo utilicen para que su juego siga creciendo. En el caso de Unity el pago es mensual dependiendo de la cantidad de dinero generado por el usuario o su empresa; los precios van desde gratuito si hace menos de 100 mil dólares anuales hasta 125 dólares mensuales si el usuario o su empresa genera más de 200 mil dólares anuales. Si hablamos del pago de Unreal Engine, si cualquiera de los productos realizados por el estudio se comercializa de forma oficial, Epic Games obtendría el 5 % de los beneficios de la obra cada trimestre cuando este producto supere sus primeros 3000 dólares. El desarrollo para dispositivos móviles en estudios independientes ha crecido de manera exponencial gracias a que motores como Unity lo hacen bastante accesible.

Unity es utilizado por el 34 % del top mil de juegos para dispositivos

móviles. Algunos de estos juegos son *Alto's Adventure*, *Monument Valley* o el famoso *Hearthstone* de Blizzard en su versión de Android e iOS.

Entonces, ¿Y si un juego se pudiese jugar en el teléfono móvil pero en verdad el juego se estuviese ejecutando en el ordenador?

Con esta pregunta se pretende poner encima de la mesa las tecnologías desarrolladas por diferentes empresas como **Nintendo** con Nintendo Switch que alterna modo portátil con modo sobremesa, lo que da una flexibilidad a la hora de jugar donde quieras que nunca se había experimentado, como **Sony** con sus aplicaciones **PS4 Remote Play** y **PS4 Second Screen** que te dan la posibilidad de controlar de manera remota e incluso jugar desde tu dispositivo iOS o Android.

Para la conexión de ya sea una consola o un ordenador a un dispositivo móvil o tablet es necesaria una conexión a internet estable. El tiempo de respuesta, lo que se conoce como *Input Lag*, puede llegar a convertirse en un quebradero de cabeza para un estudio pequeño ya que requiere de pruebas de rendimiento, pruebas con usuarios y pruebas con diferentes anchos de banda de red para buscar posibles cuellos de botella.

Este trabajo pretende aplicar los modelos propuestos anteriormente y crear una herramienta libre para el motor de videojuegos Unity con la finalidad de dar a estudios independientes y con escasa o nula financiación la oportunidad de habilitar nuevo gameplay para sus usuarios. La finalidad es dar todas estas herramientas de una forma rápida e intuitiva con documentación y una prueba de implementación de este Plugin con uno de los juegos que se ofrecen de manera gratuita en la tienda de Unity, Asset Store. Además de esto, la herramienta consta de una licencia libre, lo que da la posibilidad de ampliación y modificación dependiendo de las necesidades de cada usuario/estudio.

Capítulo 2

Estado del arte

RESUMEN: Este capítulo explica las herramientas que han sido utilizadas en el desarrollo de este trabajo. También se ofrecen unos cuantos ejemplos en los cuales nos hemos basado y nos han llevado a pensar que la idea de este proyecto es algo útil.

2.1. Herramientas para el desarrollo

En este proyecto se ha unificado el uso de varias tecnologías para formar una herramienta que una el desarrollo de videojuegos para ordenador y el uso de una aplicación de Android que nos facilite la conexión del PC con Android. Estas tecnologías/herramientas han sido:

2.1.1. Unity

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies. Se encuentra disponible para sistemas Windows, Mac Os X y Linux. Es una de las herramientas de desarrollo de videojuegos más populares actualmente en el mundo de los desarrolladores independientes. También cuenta con una gran cantidad de documentación generada tanto por los diversos usuarios como por sus creadores. Otra opción barajada era Unreal Engine 4; este motor es más potente que Unity y también es altamente usado por las desarrolladoras.

Las principales características buscadas son:

- **Multiplataforma.** El hecho de que Unity y Unreal sean sistemas multiplataforma, te garantiza poder hacer juegos/aplicaciones que puedan ejecutarse en cualquier dispositivo a un coste bastante bajo en cuanto a esfuerzo.
- **Potencia.** Unreal Engine 4 es más potente que Unity gracias a su fotorrealismo y trabajo de gráficos hiperrealista como en la demo de Star Wars creada por los desarrolladores del motor, aun así Unity no es, ni mucho menos, un motor mediocre ya que consigue buenos acabados gracias al uso de diferentes shaders y filtros. Lo decisivo es que Unity es un motor ya dado en la universidad que facilitaba el uso del mismo y, que para el objeto de este proyecto, su potencia ya es más que suficiente.
- **Documentación.** El caso de Unity en cuanto a documentación es algo que los desarrolladores de Unity mantienen funcional de manera continua. Puedes acceder a la documentación de una manera muy intuitiva desde la misma página web de Unity y en los últimos meses están incluyendo proyectos de juegos completos de principio a fin.
- **Comunidad.** Debido a que Unity es un motor de videojuegos potente y gratuito, cuenta con un gran número de desarrolladores que saben utilizarlo, lo que facilita la solución de problemas más específicos.

2.1.2. Android Studio

Android Studio es el entorno de desarrollo oficial de la plataforma Android en contrapartida también cabe la posibilidad de descargar las Android SDK sin necesidad del entorno completo de Android Studio. Fue desarrollada por Google y sustituyó a Eclipse en el desarrollo de aplicaciones para dispositivos Android. Las SDK de Android son imprescindibles para la creación de una aplicación Android, las SDK ofrecen un Emulador para poder ver tu aplicación en ejecución. Android Studio tiene la función de encapsular estas SDK de Android y poner en marcha la aplicación. Este IDE puede utilizarse tanto en Windows, Mac OS X y Linux pero únicamente puede usarse para el desarrollo de aplicaciones para Android.

Las principales características de Android Studio son:

- **Específico.** Si el producto que quieres desarrollar va a ser exclusivo de un sistema Android, con Android Studio te vas a centrar en explotar las funcionalidades de ese sistema al máximo sin tener que lidiar con diferentes sistemas que no te interesan.
- **Editor de diseño.** Android Studio cuenta con un editor visual para poder acomodar el layout de tu aplicación Android de una manera mucho más sencilla.

2.2. Librerías usadas

2.2.1. ZXing

Este es un proyecto del tipo Open-Source. Esta librería de procesamiento de imágenes de códigos de barras y QR's está implementada en Java y tiene diferentes versiones en los distintos lenguajes. En nuestro caso hemos usando la versión de .NET para usarla desde Unity. Las principales características de ZXing son:

- **Versatilidad.** Al ser una librería que tiene soporte en muchos otros lenguajes que no son Java, puedes utilizarla en tus proyectos multiplataforma si estas plataformas usan diferentes leguajes como en nuestro caso (Android Studio y Unity).

2.2.2. Unity

Unity es el motor de videojuegos elegido para desarrollar la parte de PC de este proyecto.

2.2.3. Android

ESTO NO ES GILIPOLLAS ES LA PUTA API Para desarrollo en Android se ha utilizado Android Studio que es un entorno de desarrollo integrado(IDE) oficial para el desarrollo de aplicaciones en Android basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android. Entre ellas se ofrece un sistema de compilación basado en Gradle el cual es flexible, un emulador rápido con varias funciones, al ser un entorno unificado puedes desarrollar a la vez para todos tus dispositivos Android, Instant Run para aplicar cambios mientras tu app se ejecuta sin necesidad de compilar el APK de nuevo e integración de plantillas de código y Github entre otras features.

2.3. Proyectos Similares

Hay empresas de videojuegos como Nintendo y Sony que han invertido mucho dinero en innovar y crear nuevas formas para que los usuarios disfruten de los diferentes juegos. Algunos de los ejemplos que ya existen en el mercado y han servido de inspiración para la realización de este proyecto son:

2.3.1. Wii U

Wii U es la consola doméstica que Nintendo creó en la octava generación de consolas. La innovación principal de esta consola era la de cambiar radicalmente el modo de jugar a una consola de sobremesa ya que desarrollaron el Wii U GamePad. Este GamePad tenía la función de una segunda pantalla y la de un mando a la vez. Esta pantalla portátil es táctil y recibe una señal de 480p.

Este nuevo mando permitía a los desarrolladores tener un HUD mucho más limpio dentro del juego ya que, en muchos casos, esta pantalla era utilizada para poner elementos como el minimapa y en algunos juegos como Mario Bros, había cambios de zonas que convertían al GamePad como pantalla principal.

2.3.2. GamePad

Un GamePad es un dispositivo de entrada cuya función es interactuar con los elementos de un juego para realizar diferentes acciones. Los GamePads están extendidos tanto en ordenadores como en consolas y en el último año están saliendo al mercado nuevos GamePads para dispositivos móviles, los cuales se conectan por bluetooth. Estos últimos son mandos físicos que no están al alcance de todos y actualmente no son compatibles con todos los juegos, se utilizan para juegos muy concretos como PUBG Mobile. Los GamePads han ido cambiando su forma y el número de botones de los que disponen. La necesidad del uso de GamePads en videojuegos de móvil hizo que Android 9 incluyese la posibilidad de hacer que un usuario conectase su Dualshock 3 a su Android para poder jugar.

2.3.3. PlayLink

PlayStation es una de las compañías que más tráfico de jugadores mueve llegando a la cifra de 90 millones de usuarios activos mensuales en enero de 2019, sus consolas son de las más vendidas en todo el mundo y para que esto sea posible siempre tienen que intentar estar a la cabeza de nuevos periféricos, nuevos juegos y, por supuesto, darles a sus usuarios las mejores experiencias posibles. En 2017, Sony PlayStation sacó al mercado una nueva serie de juegos llamados PlayLink. Estos juegos tienen una particularidad con respecto a un juego convencional de consola, estos juegos están hechos para jugarlos con gente y usando un dispositivo móvil como mando/controlador. Conectando la consola y el móvil a la misma red WIFI y conectándolos a través de una aplicación, se pueden conectar de 2 a 8 jugadores, depende de los que admita cada juego, para poder jugar en familia o con amigos.

Capítulo 3

Objetivos y especificación

RESUMEN: Este capítulo se centra en exponer los objetivos a cumplir por este proyecto, tales como la creación de la librería (Unity), la aplicación (Android) y la comunicación entre ambas ayudado por las metodologías y herramientas como Scrum y Github.

3.1. Objetivos

Con este proyecto se han establecido los siguientes objetivos generales:

- Crear una librería para Unity3D que se comunique con una aplicación Android usando una comunicación Cliente-Servidor, Unity3D(Servidor), el cual se encargará de recibir los inputs del Android(Cliente).
- Crear la aplicación Android para usar el propio móvil como mando virtual.
- Crear un prototipo que demuestre las capacidades de la herramienta desarrollada.

A continuación plantearemos la metodología y el plan de trabajo seguido.

3.2. Plan de trabajo

La primera fase estará dedicada a la investigación y el estudio de las herramientas ya existentes que exploran los aspectos en común con este TFG.

Se usará Github como sistema de control de versiones, donde estarán 2 repositorios: uno para la memoria y otro donde se encontrará el código de la demo. El uso de Github es debido a su amplio reconocimiento a nivel mundial, en enero de 2013 ya contaba con 3 millones de usuarios junto con 4.9 millones de repositorios. Unos datos más actuales indican que Github en Junio de 2018 alojaba casi 80 millones de proyectos. Dado a la gran cantidad de usuarios de dicha plataforma existe una gran ayuda para todo problema que surja.

Las reuniones con los directores serán cada 2 semanas. El objetivo de estas reuniones será llevar un control de lo que se haya avanzado y plantear las próximas 2 semanas de trabajo. Se han marcado unos hitos específicos:

1. **Navidad.** Sobre estas fechas deberemos tener la conexión entre Android y Unity3D establecida. Con esto se pretende que un personaje en Unity3D sea capaz de moverse gracias al Input que recibe desde Android.
2. **Semana Santa.** Se deberá de haber conseguido tener una cámara en Unity3D enviando una imagen via streaming al dispositivo Android mientras se juega con este.
3. **Pre-Exámenes Junio.** Aquí se debe tener una demo que sea capaz de dostrar las capacidades de la herramienta.

3.3. Metodología

La metodología utilizada para la realización de este proyecto está basada en el desarrollo ágil. Se han marcado pequeños objetivos de 2 semanas de duración llamados sprints hasta llegar a las metas grandes o hitos mencionados en el apartado anterior.

La metodología que se iba a usar estaba definida desde el primer día ya que es la que los miembros del proyecto siguen en todos los trabajos que realizan. Se decidió usar **Scrum**. Scrum es una metodología ágil que adopta una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del proyecto en cuestión. La particularidad que se tiene en este proyecto es el constante cambio de las tareas nuevas que se pueden llegar a incluir tras cada reunión.

3.4. Herramientas utilizadas

En la sección 3.1 se ha hablado de las herramientas que se acordaron utilizar para la realización de este proyecto:

1. **MiKTeX y LaTeX**

Las características más apreciables de MiKTeX son su habilidad de actualizarse por sí mismo descargando nuevas versiones de componentes y paquetes instalados previamente, y su fácil proceso de instalación.

2. **GitHub**

GitHub es una plataforma de desarrollo cooperativo en la que se pueden alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza para la creación y almacenamiento de código fuente de manera pública. La herramienta Git se ha usado para mantener un control de versiones sobre 2 repositorios, uno para la demo y otro para la memoria.

Capítulo 4

Arquitectura

RESUMEN:

4.1. Arquitectura General

Desde el punto del desarrollador, para poder utilizar esta herramienta va a ser necesario tener la versión XX.XX de Unity3D y Android Studio en su versión XX.XX y un teléfono móvil que podamos utilizar como emulador. Esta herramienta tiene 2 partes que están diferenciadas por la plataforma para las que van destinadas. En el lado de PC, se ha implementado una serie de clases que son independientes al juego. Estas clases están escritas en C# y utilizan las librerías nativas de red de este lenguaje para comunicación. Por encima de estas clases se encuentra la parte específica de Unity3D que facilita su uso.

Como parte extra a la herramienta, se ha propuesto un inicio de conexión a través de la lectura de un QR. Con esto lo que se pretende es enlazar el dispositivo Android con el juego que se está ejecutando en Unity. La lectura de este QR se hace directamente desde la aplicación creada para este proyecto. Al leer el QR y emparejarse ambos dispositivos, la aplicación Android cambia de cámara a mando.

[INSERTE IMAGEN DE ARQUITECTURA Y DIAGRAMA DE CLASES Y COMPONENTES + EXPLICACION]

4.2. Protocolo de conexión

La conexión se inicia con la lectura de un QR que se mostrará en el juego ejecutado en Unity. Una vez el dispositivo Android haya leído ese QR, Android manda la resolución del dispositivo con 2 bytes. El servidor que se está ejecutando en Unity, guarda esa información y escala el mando que se haya seleccionado a la resolución que tiene el dispositivo Android. Una vez hecho esto, se calculan las coordenadas de los botones del mando y se pasan a un XML.

Este XML junto a la imagen es enviado al dispositivo Android. El teléfono guarda esta información y deja de mostrar la cámara para mostrar la imagen del mando que se le ha enviado. En la aplicación Android han quedado guardadas las coordenadas de los botones, por lo que ya podría empezar la sesión de juego.

Durante una sesión de juego estándar, el dispositivo Android recibirá pulsaciones del jugador en la pantalla, calculará el botón al que corresponde y enviará al ordenador un array de 10 bytes en los que irá la información necesaria para saber la acción que ha hecho el jugador y reaccionar consecuentemente.

[ADJUNTAR IMAGEN DESCRIPTIVA DEL CONTENIDO DEL ARRAY + POSTERIOR EXPLICACION DEL SIGNIFICADO DE CADA COSA Y LOS VALORES QUE PUEDE TOMAR]

4.3. El uso de los QR para inicio de conexión

Los QR's son los códigos de barras 2D, la razón por la que se ha elegido esta tecnología para sincronizar el dispositivo Android con el juego en ejecución en Unity es debido a que es una tecnología que ambos sistemas saben interpretar y porque actualmente es un estándar, a parte nuestros usuarios ya tienen el propio dispositivo móvil en sus manos listo para sincronizar y pasar a ser su mando. En la mayor parte de ocasiones, este contenido es una URL específica, en el caso de este proyecto, lo que contiene el QR es la dirección IP del equipo que está ejecutando el juego de Unity. A su vez contiene el puerto donde el dispositivo Android debe conectarse y mandar sus mensajes. Una vez leído el QR, la aplicación Android guarda estos datos para que el juego pueda comenzar y el QR pueda desaparecer.

4.4. Pantalla secundaria y controlador

Como se dijo en el capítulo 2, para este proyecto se ha usado de referencia las consolas Wii U de Nintendo y PlayLink de Sony. Se quiere que los

desarrolladores de los juegos hecho en Unity pudiesen mostrar elementos del juego como un mapa, un inventario o unas notas que seguir fuera del HUD de juego para que este no se viese saturado.

Es por esta idea, por lo que a la aplicación de Android se le ha dado la capacidad de recibir imágenes y mostrarlas por la pantalla del Android. Con esto se da la posibilidad de enviar una imagen que ya haya sido creada con anterioridad o incluso se abre la posibilidad de enviar lo que está renderizando una cámara dentro del juego en tiempo real y hacer que nuestro dispositivo Android lo muestre.

Esto lo que implica es que la comunicación entre la aplicación Android y el juego de Unity va a ser constante y se corre el riesgo de saturar la red. Para solventar este posible fallo, se decidió comprimir la imagen que se enviaba a formato PNG. Este formato fue el elegido porque tanto Unity como Android dan soporte de compresión y descompresión de manera eficaz.

Capítulo 5

Desarrollo de la aplicación de Android

RESUMEN:

5.1. Manejo de actividades

Toda aplicación de Android está basada en el manejo de actividades. El ciclo de vida de Android establece el framework que cualquier aplicación va a seguir y en los casos en los que hay que prestar especial atención a las interacciones con usuario. Debido a que es Android el que gestiona la actividad, siempre se tiene que tener en cuenta que el usuario puede minimizar, salir o incluso cerrar la aplicación de manera abrupta debido a un apagón en el dispositivo.

En el caso particular de este proyecto, se quiere usar la cámara para poder leer el código QR y así obtener la información necesaria para iniciar una comunicación con la aplicación ejecutada en PC. Tras leer el código, la interfaz necesita cambiar para mostrar el mando que se quiere usar para jugar.

Para resolver este problema se ha planteado la creación de una segunda actividad cuando se lea un QR válido. El funcionamiento de esto hace que la actividad con la que arranca la aplicación sea la de una cámara usando el API proporcionado por Android. Esta cámara está ejecutándose hasta que lee un código QR válido y crea una nueva actividad que es en la que se muestra el mando para empezar a jugar.

La primera actividad con la que se inició la aplicación termina y se queda como actividad principal la segunda, que se ha creado al leer el QR. Con esto se consigue un reseteo de la interfaz para que esta pueda ser modificada.

5.2. Interfaz del modo mando

La interfaz de la nueva actividad cambia por completo respecto a la que habia con el modo cámara. Esta interfaz muestra el mando seleccionado en la aplicación de PC. Este mando emula la posición de los botones del mando seleccionado. La posición de estos botones tiene que variar con respecto a la resolución de la pantalla de cada dispositivo.

Para hacer esto posible, la aplicación de PC tiene guardado un XML con cada uno de los mandos. Este documento describe la posición de los botones en una resolución estándar que varía con respecto a una constante que viene dada por la resolución de cada dispositivo. Una vez la aplicación de PC calcula los valores de los puntos donde deben situarse los botones, este se los manda de vuelta al móvil para que este los muestre en la interfaz. Esto hace que en el XML no se tengan almacenadas todas las posibles combinaciones de la posición de los botones ya que esto sería un trabajo tedioso y por un error humano, hay combinaciones que pueden ser olvidadas a la hora de rellenar el documento.

Por otra parte, la interfaz de la aplicación Android está preparada para mostrar una imagen de fondo que puede ser estática o cambiante. Este modo deberá ser activado por el desarrollador de cada juego y será notificado a la aplicación Android en forma de flag.

En la demo realizada para este proyecto se ha puesto como fondo de la aplicación los frames capturados por la cámara de Unity para que pueda usarse el teléfono como pantalla y mando a la vez en el caso de querer usar el ordenador para otra tarea.

5.3. Manejo de hilos de ejecución

Para que todas las tareas mencionadas hasta ahora sean posibles, es útil el uso de hilos y una gestión minuciosa para no causar ningún problema de concurrencia.

La aplicación que se ha propuesto para este proyecto crean 2 hilos adicionales, estos hilos van a encargarse de recoger el input del usuario y mandarlo a la aplicación de PC, tambien van a encargarse de recibir la imagen de fondo, descomprimirla y plasmarla como fondo de la aplicación.

El primer hilo de ejecución y el que siempre va a estar activo es el que va a recoger el input del usuario. Este hilo, gracias al API de Android, tiene la capacidad de guardar la posición donde ha tenido lugar el input del usuario. A su vez, tiene guardados los hotspots de cada botón debido a los datos recibidos con anterioridad por la aplicación de PC. Para saber si el usuario ha pulsado en uno de los botones del mando, el hilo comprueba si este punto coincide con alguno de los hotspots y en caso afirmativo, activa el flag correspondiente a ese botón y lo manda a la aplicación de PC para

que este ejecute la lógica de la acción que debe ocurrir al pulsar dicho botón (moverse, saltar, etc). Esto también ocurre al levantar la pulsación de un botón.

El segundo hilo se usa exclusivamente para el tratado de la imagen de fondo. En el caso de que el desarrollador del juego no quiera usar una imagen de fondo, este hilo no va a necesitarse por lo que no se creará. Dado el caso de que se use, este hilo únicamente recibe información que en nuestro caso es una imagen en formato PNG comprimido. Este hilo es el encargado de descomprimir la imagen, adaptarlo a la resolución del teléfono y modificar la imagen que se muestra en el fondo de la aplicación. Tanto si la imagen es estática como si se quiere enviar un streaming de imágenes, el tiempo de latencia es de 15ms de media por lo que es prácticamente inapreciable para el ojo humano, cuya latencia aceptada es inferior a 13ms.

Capítulo 6

Desarrollo de la aplicación de Unity

RESUMEN:

6.1. Establecimiento de conexión

El motor de videojuegos de Unity está construido usando el lenguaje de programación C#. Con el uso de este lenguaje de programación, se abre el abanico de frameworks que pueden usarse para complementar cualquier aplicación programada en este lenguaje. Uno de estos frameworks es .NET. El framework .NET fue desarrollado en 2002 por la empresa Microsoft y en este proyecto se usa para el apartado de comunicación por red de Unity y una aplicación ejecutada en Android. .NET hace énfasis en la independencia de las plataformas en la que se ejecuten las aplicaciones que se quieren comunicar, este es el motivo de su uso en este proyecto. La versión a utilizar es la 4.7.2 y únicamente se usa como API de envío y recepción de mensajes en un socket.

Como se ha explicado en capítulos anteriores, en este proyecto se comunican 2 aplicaciones: una ejecutada en un ordenador y otra en un dispositivo Android. Para iniciar la conexión entre estos es necesario el uso de un servidor que reciba datos y un cliente que los envíe.

En la primera parte de este proyecto el servidor es inicializado y ejecutado en la aplicación de PC, en un hilo aparte que se encarga de recibir el Input del jugador pero este punto se explicará con más profundidad en el apartado siguiente. Para poder emparejar ambos dispositivos, se llegó a la conclusión de que un QR era la opción menos intrusiva y a la vez más cómoda para el usuario ya que con un simple gesto, todo estaría emparejado y conectado sin que el usuario tuviese que hacer ninguna acción más. Este código QR, a

diferencia de en su uso cotidiano, no lleva una URL en su interior sino que lleva la IP del ordenador donde se está ejecutando la aplicación de Unity y el puerto al que deben enviarse los datos. La única premisa para que ambos dispositivos se emparejen, es que ambos estén conectados a la misma red. Para conseguir la IP del PC cada vez que ejecutamos la aplicación y que esta sea diferente en cada red a la que nos conectamos, usamos la API de .NET una vez más para tener acceso al DNS y que este nos de la dirección IP del ordenador.

6.2. Recepción de Input

Tal y como se dijo en el capítulo anterior, la aplicación de Android envía por red el botón que se ha pulsado. Esto cuando llega a la aplicación de PC es leído e interpretado, lo único que se necesita es saber el significado de la ristra de bytes que se envían pero eso ya fue explicado en el capítulo 4 en el apartado donde se explica el protocolo de conexión. Una vez hecho esto, hay que leer los bytes referidos al movimiento desde una clase exterior al servidor y que ya ejecute API de Unity porque el servidor como tal es independiente de Unity y sólo usa .NET.

Esta última opción es la que se plantea y se usa en el ejemplo que se ha hecho para demostrar el funcionamiento de este proyecto. Para que estos datos sean leídos correctamente sin ser borrados o alterados, desde el servidor se rellena un array que se encuentra en la clase del movimiento del jugador.

Hacer este guardado permite leer el array una única vez y una vez que se ha rellenado se ejecuta el Update de la clase encargada del movimiento del jugador, lo que hace que este se mueva y no entre en concurrencia con el resto de datos entrantes por el servidor tras el Input del jugador desde la aplicación de Android.

6.3. Envío de imagenes por red

Como feature adicional a lo planeado en un inicio, se propuso la idea de poder usar el dispositivo Android como un mapa interactuable o incluso como pantalla secundaria. La idea era dar la mayor libertad creativa al desarrollador del videojuego.

Para hacer esto posible se ha implementado un API que coge la textura de una cámara de Unity y la envía por red. La imagen que posteriormente va a ser enviada, es cogida de una cámara de Unity y guardada en otra textura 2D. Esta nueva textura es leída y transformada a PNG, lo que la comprime. Tras ser enviada, la textura es recibida en un formato comprimido que se descomprime en la aplicación Android para ser mostrada como fondo de la pantalla. Todo este proceso se lleva a cabo en un hilo que se ejecuta concurrentemente en la aplicación de PC. Este hilo tiene la opción de no ejecutarse

ya que el desarrollador puede elegir si necesita o no que su videojuego envíe una imagen de fondo o solamente reciba Input del jugador. Con esto puede enviarse una imagen pero en la demostración hecha para este proyecto se envía constantemente el contenido de la cámara principal del videojuego para que en el fondo de la aplicación se vea el juego de una manera fluida.

Capítulo 7

Conclusiones

RESUMEN:

7.1. Conclusiones Pablo

7.2. Conclusiones Sergio

