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

Mayo 2020

Documento maquetado con TEXIS v.1.0+.

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

Mayo 2020

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

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 características y lo utilicén 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 utilziado 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 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 usarios. 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, la 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

2.1. Herramientas para el desarrollo

En este capítulo se expondrán las diferentes tecnologías que se van a utilizar para la realización de este proyecto. Además de eso, se pondrán encima de la mesa las diferentes aplicaciones desarrolladas por empresas del sector de los videojuegos que han sido tomadas como referencia para la creación de este trabajo.

2.1.1. Unity

Unity es un motor de videojuegos multiplaforma 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. Con unity se han realizado algunos de los juegos más famosos del mercado como **Ghost of a Tale**, **Cuphead** o **Hollow Knight**.

Las principales características buscadas son:

- **Multiplataforma.** El hecho de que Unity sea sistemas multiplataforma, te garantiza poder hacer juegos/aplicaciones que puedan ejecutarse en cualquier dispositivo a un coste bastante bajo en cuanto a esfuerzo.
- **Herramientas y servicios.** Unity es una herramienta que no engloba únicamente motores para el renderizado de imágenes, simulaciones físicas 2D y 3D, animación de personajes y audio sino que al tratarse de un motor que usa C# como lenguaje de scripting, dispone de todas las herramientas de .NET para el desarrollo. Uno de los servicios que

ofrece Unity es **Unity Analytics** que ayuda a los creadores a realizar analíticas para ver cómo juegan sus jugadores.

- **Documentación.** Los desarrolladores de Unity ponen a disposición de los usuarios una documentación amplia y detallada, tanto que incluso disponen de un historial de versiones de la documentación por si trabajas con un proyecto de Unity con una versión no actualizada. Además de esto, ofrecen documentación sobre la API y ofrecen proyectos realizados por los propios desarrolladores del motor y tutoriales de como sacar el mayor rendimiento posible a las herramientas que ofrece el motor en su plataforma **Unity Learn.**.
- **Comunidad.** Debido a que Unity es un motor de videojuegos demandado y gratuito, cuenta con un gran número de desarrolladores que saben utilizarlo, lo que facilita la solución de problemas más específicos. Unity tiene una comunidad muy amplia y muy activa tanto en foros como **Stack Overflow** y en su pripio portal de preguntas **Unity Answers.**.

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.

2.1.3. ZXing

Este es un proyecto del tipo Open-Source. Esta libreria de procesamiento de imágenes de códigos de barras y QR's está implementada en Java y tiene

diferentes versiones en los distintos lenguajes. En el caso de este proyecto, se ha usado la versión de .NET para usarla desde Unity. Al ser una librería que tiene soporte en muchos otros lenguajes que no son Java, puede ser utilizada en proyectos multiplataforma si estas plataformas usan diferentes leguajes como en este caso con Java y C#.

2.1.4. Android SDK

Cada vez que Android saca una nueva versión de su sistema operativo le acompañ su correspondiente SDK. Estas SDK de Android incluyen librerias que son necesarias para el desarrollo para dispositivos Android, además de documentación del API, un debugger y un emulador para probar los proyectos sin necesidad de tener un dispositivo físico. Este SDK suele usarse acomplado a un IDE que como se ha explicado anteriormente en el punto **2.1.2** ha sido **Android Studio**.

2.2. Interfaces de entrada

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:

2.2.1. 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 mando tenía la función de una segunda pantalla y la de un mando tradicional 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 mando en pantalla principal.

2.2.2. Controlador de videojuegos inalámbricos

Un controlador de videojuegps es un dispositivo de entrada cuya función es interactuar con los elementos de un juego para realizar diferentes acciones. Los controladores de videojuegos están extendidos tanto en ordenadores como en consolas y en el último año están saliendo al mercado nuevos mandos

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 controladores de videojuegos han ido cambiando su forma y el número de botones de los que disponen. La necesidad del uso de mandos 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.

2.2.3. 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.

2.2.4. PS4 Remote Play

En el último trimestre del año 2019, Sony lanzó la aplicación **PS4 Remote Play** para que los usuarios pudieran controlar su PlayStation 4 desde un dispositivo móvil. Esta aplicación tenía como requisito una conexión a internet de entre 5 y 12 Mbps por conexión LAN para una mejor experiencia. Esta aplicación permite no solamente el manejo de una PS4 usando un dispositivo móvil sino que además permite conectar un Dualshock 4 para controlar el juego y usar el móvil únicamente como pantalla.

Capítulo 3

Objetivos y especificación

3.1. Objetivos

Tal y como se ha podido ver en los capítulos anteriores, las empresas de videojuegos han puesto todos sus esfuerzos para adaptarse e integrar los dispositivos móviles dentro del mundo de los videojuegos. Con las tecnologías mencionadas en el cápitulo dos como referencia directa, los objetivos que se plantearon para el proyecto son:

- Desarrollar una librería para Unity3D. Esta consistirá en un servidor que ofrecerá su IP y el puerto elegido para la comunicación mediante un QR que aparecerá en pantalla.
- Desarrollar una aplicación Android para usar el propio móvil como mando virtual. Esta aplicación utilizará la cámara para leer el código QR e iniciar una conexión con el servidor.
- Integrar las herramientas mencionadas en los puntos anteriores en un proyecto ya cerrado para demostrar su funcionalidad.

Con la integración del proyecto en un juego ya cerrado se pretende hacer una serie de pruebas con usuarios para ver el rendimiento de la herramienta en ordenadores y dispositivos móviles con diferentes características. Para eso se implementarán herramientas para la recogida de datos de los usuarios y las acciones realizadas durante las sesiones. Estos datos serán analizados en el capítulo 5 con mayor profundidad.
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[1] 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 2-3 semanas, dependiendo de la disponibilidad y horarios. El objetivo de estas reuniones será llevar un control de lo que se haya avanzado y plantear las próximas 2-3 semanas de trabajo. Se han marcado unos hitos específicos:

1. **Hito 1:** 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. **Hito 2:** Se deberá de haber conseguido tener una cámara en Unity3D enviando una imagen via streaming al dispositivo Android mientras se juega con el Input recibido del dispositivo móvil.

3. **Hito 3:** Aqui se debe tener una demo que sea capaz de demostrar las capacidades de la herramienta.

4. **Hito 4:** Pruebas con usuarios, recogida de información y análisis.

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.

[1] https://github.com/

3.4. Herramientas utilizadas

Es resultado final del proyecto involucra tanto un ordenador como un dispositivo móvil conectados a la misma red. Para el desarrollo usaremos las siguientes herramientas software de comunicación, seguimiento y planificación:

1. **LaTeX**[2]

   Para la realización de este documento se ha usado LaTeXen su distribución de MiKTeX [3] para Windows. 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[4]. 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.

3. **Discord**[5]

   Discord es una plataforma de comunicación por voz, texto y vídeo. Discord ofrece la posibilidad de crear canales dedicados, lo que se ha usado para guardar enlaces de interés para el desarrollo prematuro de este proyecto. Esta herramienta es multiplataforma y no requiere de instalación ya que puede usarse desde navegadores como Chrome o Firefox. Además de esto consta con un sistema de tranferencia de archivos y de retransmisión de tu pantalla a las personas que se encuentren en la llamada. Todas estas características han sido utilizadas para las sesiones de trabajo grupal.

[2] https://www.latex-project.org/
[3] https://miktex.org/
[4] https://git-scm.com/
[5] https://discord.com/

4. **Unity**[6]

   Unity ha sido el entorno de desarrollo elegido, la versión de Unity utilizada ha sido la 2018.4.18f1. La elección de Unity fue por la gran comunidad de usuarios y la extensa y detallada documentación con la que cuenta este motor. Además, al usar C# como lenguaje de programación, se ha contado con toda la API y documentación de Microsoft sobre .NET.[7]

5. **Visual Studio 2019**[8]

   La decisión de usar Visual Studio como IDE fue por la integración que tiene con Unity. Gracias a esta integración se ha podido crear y mantener archivos de proyectos de Visual Studio automáticamente. Algo a tener en cuenta es que no se usa el compilador de Visual Studio, sino que se usa directamente el compilador de C# para la compilación de los scripts creados en Unity.

6. **Android Studio**[9]

   Se ha decidido usar este IDE para el desarrollo de la aplicación para Android. Android Studio dispone de una construcción del proyecto basada en Gradle[10], lo que hace que su construcción se haga de manera dinámica y un editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario. Además de incluir las SDK de Android y todo lo que estas incorporan para que el desarrollo en Android sea posible.

[6] https://unity.com/es
[7] https://docs.microsoft.com/es-es/dotnet/
[8] https://visualstudio.microsoft.com/es/
[9] https://developer.android.com/studio
[10] https://gradle.org/

Capítulo 4

Implementación y Especificación

4.1. Introducción

Una vez decididas las herramientas que se iban a usar y con gran parte de la información recogida, llegó el momento de realizar un análisis del funcionamiento de Android y diseñar un protocolo de red que ayudase a conectar la aplicación de Android con el juego de Unity. En este capítulo se explicarán todas las decisiones que se tomaron a la hora de implementar ambas aplicaciones y se ampliará la información sobre como interactuan ambos sistemas.

4.2. Arquitectura global

Tal y como se ha planteado en los anteriores capítulos, para la realización de este proyecto se ha requerido de la conexión entre una aplicación de escritorio realizada con Unity3D y una aplicación de un dispositivo móvil Android. Ambos dispositivos recibirán y enviaran mensajes pero el protocolo a seguir ha sido UDP. UDP es un protocolo a nivel de transporte que permite el envío de datagramas a través de la red sin necesidad de haber realizado una conexión previa. Esto se consigue gracias a que la cabecera del propio datagrama UDP incorpora la información suficiente para realizar su direccionamiento.
Con la utilización de UDP se corre el riesgo de pérdida de información, cosa que para este proyecto es menos probable ya que en ningún momento este

datagrama sale a internet. Se ha puesto como requisito que tanto el ordenador como el dispositivo móvil se encuentren conectados a la misma red local, lo que hace muy poco probable la pérdida de paquetes UDP. Esta posible pérdida será analizada en el capítulo siguiente donde se analiza las pruebas realizadas con usuarios.

A la hora de iniciar la conexión, es la aplicación móvil la que va a conectarse al puerto que abra la aplicación de escritorio previamente. Para poder hacer esto, se propuso la utilización de un QR. Este QR contiene la información de la IP del ordenador donde se está ejecutando el juego y el puerto designado para la recepción de los mensajes provenientes de la aplicación móvil. Una vez la aplicación móvil guarda esos datos extraidos tras leer el código QR, se incia la conexión y el intercambio de paquetes entre ambas aplicaciones.
La decisión de usar UDP cobró su importancia en este apartado ya que no se espera un intercambio ordenado de información entre ambas aplicaciones ,ya que el usuario puede hacer una pulsación en la pantalla del móvil en cualquier momento. La información que se envía desde la aplicación Android es la siguiente:

1. Dimensiones de la pantalla del dispositivo móvil con 2 enteros, ancho y alto.
2. Pulsación en la pantalla con 3 enteros (X, Y, tipo de evento).

Las dimensiones se mandan únicamente una vez al iniciar la conexiones mientras que las coordenadas de las pulsaciones se envían cada vez que estas ocurren. El tercer entero que se manda en estos paquetes viene definido por cada tipo de pulsación que se puede registrar en Android. Los tipos contemplados son:

- **ACTION_DOWN:** Envía el valor 0. Este evento se dispara al realizar una pulsación en la pantalla.
- **ACTION_UP:** Envía el valor 2. Este evento se dispara al levantar el dedo de la pantalla.

En el lado del servidor de la aplicación de Unity se espera la llegada de las dimensiones de la pantalla que envía la aplicación móvil para enviar desde su lado la siguiente información en paquetes diferentes:

1. Tiempo de vibración que debe realizarse cada vez que se pulse una posición correcta de la pantalla del móvil. Se considera una pulsación correcta un punto que esté dentro de un botón.
2. Imagen del mando con o sin fondo cogido directamente de una cámara de Unity.

3. En caso de que el juego admita la vibración del móvil, se envía la acción de realizar la vibración o no.

El primero de estos mensajes con el tiempo de vibración solamente se envía una única vez al inicio de la conexión.
El flujo de mensajes durante la ejecución de estas aplicaciones tiene un intercambio de datagramas constante de órdenes para que el dispositivo móvil vibre, imágenes compridas y coordenadas de pulsaciones de la pantalla del dispositivo móvil. La figura XXXX describe este protocolo de manera más gráfica.

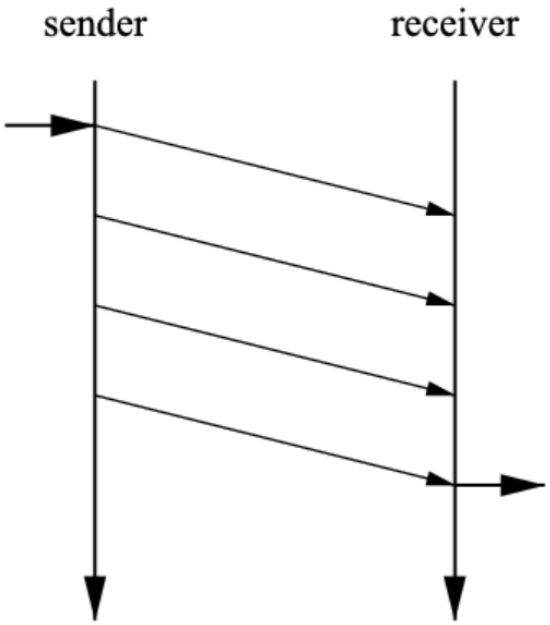

Figura 4.1: Protocolo de comunicación entre las aplicaciones

4.3. Aplicación móvil

La principal función de este proyecto es el uso de un dispositivo Android para controlar de manera remota un videojuego realizado en Unity. Para esto es necesario conocer la arquitectura de Android y cómo funciona el ciclo de vida de sus aplicaciones. La siguiente figura muestra el ciclo de vida de cualquier aplicación que se ejecute en un dispositivo Android.

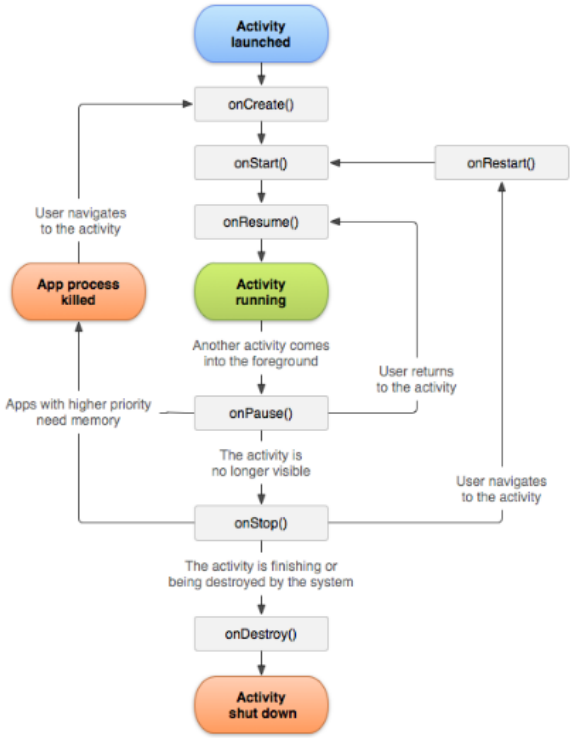

Figura 4.2: Ciclo de Vida de una Actividad de Android

Como puede observarse, cada Activity de una aplicación ejecutada en Android tiene varias etapas por las que puede pasar:

1. onCreate()
2. onStart()
3. onResume()
4. onPause()
5. onStop()
6. onDestroy()

Cuando el usuario comienza a abandonar la actividad, el sistema llama a métodos para desmantelarla. En algunos casos, este desmantelamiento es solo parcial ya que la actividad todavía reside en la memoria (por ejemplo, cuando el usuario cambia a otra app) y aún puede volver al primer plano. En caso de que el usuario vuelva a poner dicha actividad en primer plano, esta se reanuda donde el usuario la dejó.
Además de ciclo de vida, Android tiene un sistema de permisos. Desde la versión 6.0 de Android (nivel de API 23) pueden incluirse en el fichero de manifiesto de la app los permisos que se deben solicitar al usuario para poder acceder a algunos recursos específicos. En este proyecto se necesita usar la cámara para poder realizar la lectura del código QR, es por esto que se ha tenido que añadir este permiso en el fichero de manifiesto de la app.
Este código QR es generado gracias a la librería de ZXing.NET desde la aplicación de Unity, la cual se cuenta en la siguiente sección. Android por su parte incorpora una API para la lectura de códigos. El código QR contiene la siguiente información:

- IP del servidor donde se está ejecutando el juego.
- Puerto de escucha del servidor.

Si el código QR que se lee es correcto, se lanza una segunda Activitidad. La creación de una segunda actividad permite usar un archivo de manifiesto diferente y con él, generar una interfaz de usuario adecuada a esta segunda actividad. Esta segunda actividad tiene como propósito establecer la comunicación con el servidor ya iniciado en la dirección IP extraida del código QR y a su vez el intercambio de información (pulsaciones e imágenes) durante la sesión de juego.

Figura 4.3: Vista del mando enviado desde Unity a Android

En caso de que el código QR se lea correctamente, se inciará la conexión con

el servidor que se está ejecutando en Unity. En el siguiente apartado se explica la ejecución en la lado de Unity y el inicio de la conexión con este. Desde Unity se envía de manera constanste un streaming de imágenes sacadas de una cámara auxiliar colocada en la escena del juego. Esto permite en el envío de tanto un mando, mando con gameplay o únicamente gameplay. Para la demostración de este proyecto, se ha optado por el envío de la imágen de un mando de manera exclusiva. Este mando puede verse en la figura XXXXX .

Esta imágen llega al dispositivo Android tras realizarse una compresión a formato PNG en el lado del servidor. Este array de bytes que llega es convertido a un tipo Bitmap mediante *BitmapFactory* y es usado como fondo de la Actividad. Este proceso se realiza constantemente ya que en caso de que se quiera utilizar para gameplay, este requiere de fluidez. En el capítulo siguiente se tratan las pruebas con usuario donde se hace un estudio de cuánto tarda esta imágen en procesarse.
La arquitectura de este proyecto de Android puede verse en la figura XXXX.

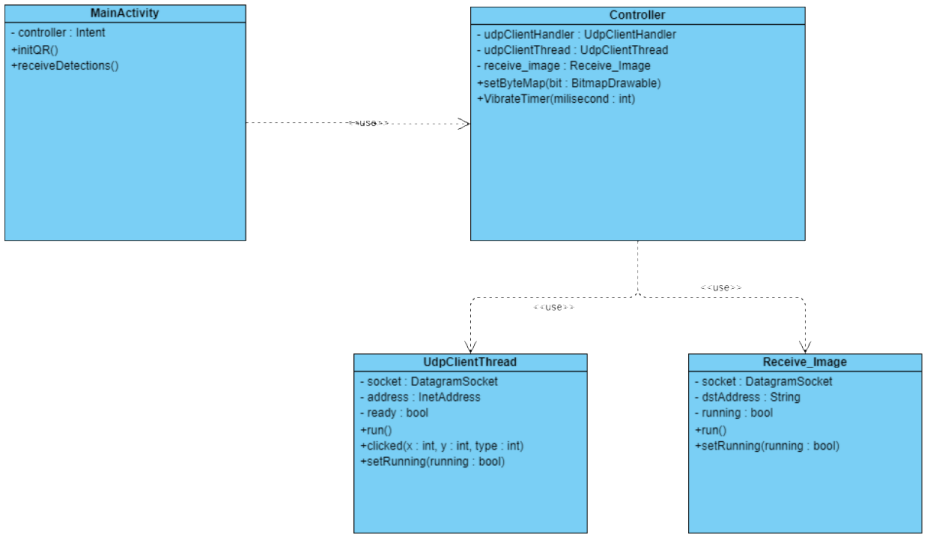

Figura 4.4: Arquitectura de la aplicación Android

MainActivity y **Controller** son las 2 actividades que se han comentado anteriormente que existen en este proyecto de Android. La función de MainActivity es la de la lectura del QR. Su diseño tiene como único propósito el de mostrar la cámara para la lectura del QR. Se encarga de pedir los permisos necesarios para el uso de la cámara. Cuando el QR es correcto, guarda la información y antes de terminar ordena la creación de una segunda actividad con la información del QR. La información del QR es de tipo

String y tiene como formato *IP:Puerto*.
Además de la información del QR y las imágenes, hay otro dato que interviene en la comunicación entre las 2 aplicaciones planteadas para este proyecto. Estos datos son las pulsaciones de la pantalla del dispositivo Android. En estos mensajes se envían los datos de las coodenadas (x,y) donde el usuario ha pulsado y además el tipo de gesto que ha hecho (pulsar o levantar). La disposición de los bytes que se envían puede verse en la figura XXX. Un mensaje que únicamente se realiza al inicio de la conexión es el del envío de las dimensiones de la pantalla que tiene el dispositivo Android. La disposición de los bytes de este mensaje puede verse en la figura XXX.

4.4. Aplicación PC

Como ya se ha mencionado en el capítulo 2, la aplicación de escritorio que contiene el servidor y el juego está desarrollada en Unity. Unity es un motor de videojuegos basado en entidades y componentes. Estos componentes están concebidos para ser características independientes que se añaden a las entidades. Por su parte las entidades por si mismas no realizan ninguna acción hasta que no se le añaden componentes. Este sistema anima a integrar partes independientes a la arquitectura general de un proyecto sin necesidad de hacer cambios drásticos, es por esto que la idea principal de este proyecto era la de desarrollar una serie de scripts que añadiesen la posibilidad de utilizar un dispositivo Android para el control de un videojuego ya terminado. Para el inicio de la conexión se ha utilizado la librería de ZXing, la cual tiene su versión para diferentes lenguajes. En el caso de C# es suficiente con tener la DLL en el directorio del proyecto. ZXing es una librería que soporta tanto la generación como la decodificación de códigos de barras. Algunos de los diferentes formatos que soporta son:

- Códigos QR
- PDF 417
- EAN (European Article Number)
- UPC (Universal Product Code)
- Aztec Code

Un código QR suele utilizarse para la conexión con una URL ya que es más rápido y fácil leer un código QR que escribir una URL completa en un dispositivo móvil. El QR utilizado en este proyecto no es estático ya que no contiene ninguna URL, lo que contiene son tanto la IP como el puerto al que debe conectarse la aplicación móvil para poder jugar al juego. Es por esto que no sirve cualquier aplicación genérica de lectura de QRs y se ha tenido que integrar dentro de la aplicación móvil que se utiliza para controlar el videojuego. Una vez este código se ha creado con la IP del ordenador al que conectarse y el puerto habilitado para el intercambio de datos, el QR se genera como si formase parte de la interfaz del usuario hasta que este se conecte.
La primera acción que se realiza desde el PC una vez se ha realizado la conexión es la del envío del tiempo de vibración en caso de que los desarrolladores del juego quieran dar ese feedback al usuario. Este mensaje es un único valor de tipo *int* que indica el tiempo de vibración. Inmediatamente después comienza el envío de imágenes, estas imágenes son enviadas 1 vez por cada frame del juego. La decisión de enviar estas imágenes es porque se

da la posibilidad de ver gameplay fluido en el dispositivo móvil donde se está jugando. Dentro de Unity esto es posible gracias al uso de múltiples cámaras dentro de la misma escena. Para este proyecto se ha propuesto el uso de una cámara auxiliar que sirva para decidir lo que el usuario va a ver en su teléfono móvil. Al igual que se explicó en el apartado anterior con Android, Unity dispone de varias funciones que son llamadas directamente por el entorno. Unity dispone de la función **LateUpdate()**, la cual se llama al final de cada frame del videojuego. Gracias a esta función, el motor nos asegura que todas las transformaciones físicas se han realizado, por lo tanto, la cámara auxiliar puede sacar una instantánea de lo que está viendo y enviarsela al usuario por red. Para que esta imágen pese lo menos posible, se la somete a una compresión PNG. Esta compresión PNG está basado en algoritmo de compresión de datos sin pérdidas conocido como ***DEFLATE o Deflación***. Para que esta conversión sea posible, la API de Unity proporciona la función **EncodeToPNG()**. [1] Esta función devuelve una array de bytes preparados para ser enviados directamente al dispositivo móvil por red.
Desde Android se envían las coordenadas de las pulsaciones para que dependiendo del mando, esto signifique si se ha pulsado un botón. Esta notificación se realiza siguiendo el **patrón listener**. Este patrón de diseño de software define que el cambio del estado de un objeto es notificado a todos los miembros dependientes. El objetivo de usar este patrón de diseño es el de definir una dependendia uno a muchos. Esto es una modificación del patrón **Observador** ya que en este caso un elemento no quiere estar pendiente de otro sino que se le avise automáticamente.

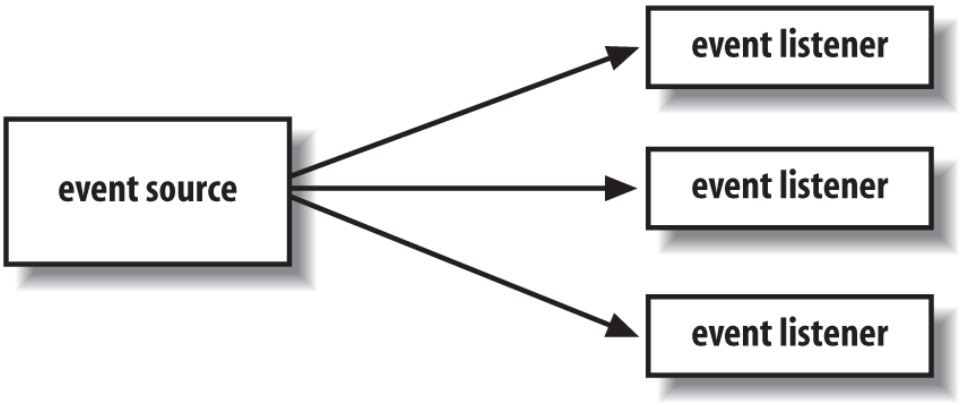

Figura 4.5: Patrón Listener

La librería realizada para este proyecto utiliza este patrón. Para esto se ha implementado una interfaz llamada **InputMobileInterface** que sirve para avisar al objeto que implemente dicha interfaz de:

[1] https://docs.unity3d.com/ScriptReference/ImageConversion.EncodeToPNG.html

- Recepción de una pulsación por parte del usuario.
- Valores del alto y ancho en pixeles de la pantalla del teléfono móvil del usuario que se ha conectado.
- Notificación del final de la conexión.

Al tratarse una interfaz pública, cualquier clase puede convertirse en Listener de InputMobileInterface heredando de esta e implementando los 3 métodos mencionados anteriormente. En el capítulo siguiente se explicará de manera más detallada los pasos a seguir para la inclusión de esta serie de scripts en un juego ya terminado junto con las pruebas con usuarios. La librería desarrollada consta de 3 scripts que se comunican entre si para realizar las funciones que se han descrito en este apartado.

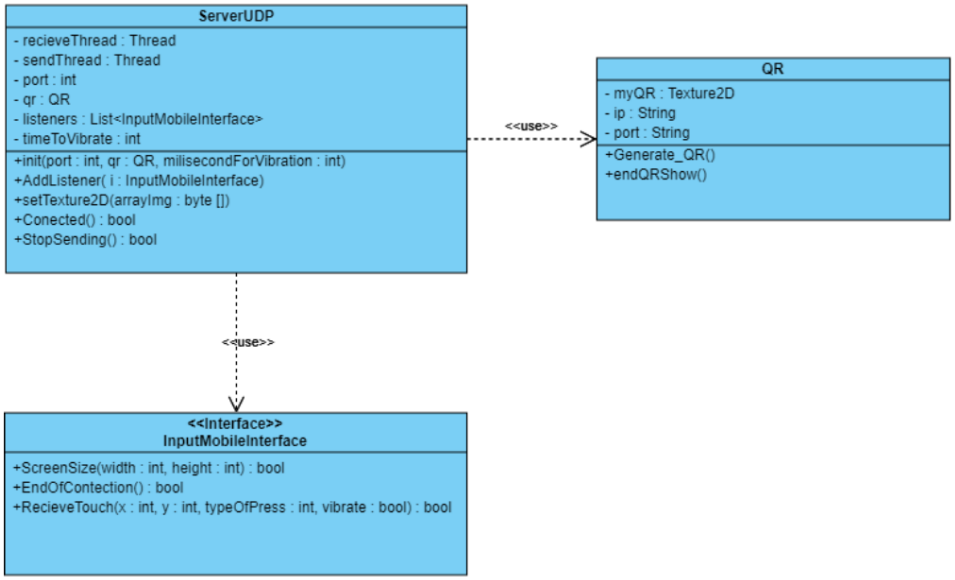

Figura 4.6: Arquitectura de la librería implementada para Unity

La clase **ServerUDP** es la clase principal y es de la cual se requiere tener una instancia en el proyecto. Esta clase es la encargada de guardar y registrar listeners. Además se encarga de utilizar la clase **QR** que implementa la funcionalidad para generar el código QR correcto con el uso de la librería ZXing.Net. Por último, la clase ServerUDP tiene como cometido la inicialización del socket y la de inicar los hilos de recepción y envío de datos al dispositivo móvil cuando este se conecte. Una vez lleguen los datos de las pulsaciones, esta clase también se encarga de avisar a todos los listeners que se hayan registrado.

Capítulo 5

Pruebas con usuarios

5.1. ?Â¿

Capítulo 6

Conclusión y trabajo futuro

6.1. Â¿?