ENTORNO VIRTUAL MULTIUSUARIO EN UNITY

Autor : Roberto Martín Ballester

Tutor : Ximo Cerdà Boluda

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2015-16

Valencia, 1 de julio de 2016

Resumen

En este proyecto estudiaremos cómo crear entornos multiusuario mediante una de las herramientas más utilizadas ultimamente por desarrolladores de videojuegos, el motor 3D de Unity. Descubriremos cómo nació y las características que hacen que sea tan empleado a día de hoy tanto por pequeñas como grandes empresas, y veremos el funcionamiento básico de su editor.

Analizaremos las herramientas de que dispone Unity para crear entornos multiusuario a través de internet, tanto de forma nativa, como a partir de las extensiones más utilizadas de su Asset Store, PUN (Photon Unity Networking) y Extremality Virtual World Framework.

Por último, crearemos un entorno simple de ejemplo, en el que podamos comprobar el proceso de creación y gestión de un mundo virtual y las características que ofrecen las herramientas mencionadas.

Resum

Aquest projecte estudia com crear entorns multiusuari per mitjà dels utensilis més utilitzats del moment per programadors de videojocs, el motor 3D d'Unity. Descobrirem com va nàixer i les característiques que fan que siga emprat per grans i xicotetes empreses, i vorem el seu funcionament bàsic.

Analitzarem els utensilis de que disposa Unity per fer entorns multiusuari, ja siga emprant els inclosos a l'editor o per mitjà de les extensions més utilitzades de la seua Asset Store; PUN (Photon Unity Networking) i Extremality Virtual World Framework.

Finalment, crearem un exemple simple d'entorn multiusuari, per a comprobar el procés de creació i gestió d'un món virtual i les característiques que oferixen les extensions nomenades.

Abstract

This project tries to study the posibilities we have to develop a virtual multi-user environment emplyoing the most used engine of the moment, Unity. We will find about its origins and the main funcionalities that makes this editor the most used along small and big enterprises, and we will learn about how it works.

We will take a look at the main tools Unity has to develop virtual multi-user environments, and we will explore the most used extensions as well, as PUN (Photon Unity Networking) and Extremality Virtual World Framework.

Finally, we will try to develop a simple virtual world with this tools, to learn more about its complexity and capabilities.

Índice

Capítulo 1. Introducción

En este primer capítulo explicaremos las motivaciones que nos llevan a realizar este trabajo, introduciendo el contexto actual de las tecnologías que utilizaremos y de las que hablaremos más detalladamente en el capítulo 2, y explicando en qué consistirá nuestro proyecto y cómo lo abordaremos.

1.1 Motivación

La industria de los videojuegos ha experimentado unas tasas de crecimiento muy altas en los últimos años, llegando a generar más dinero que la propia industria del cine, con una comunidad objetivo en constante crecimiento, debido al desarrollo de las tecnologías.

Éste crecimiento se ve reflejado también en la aparición de nuevas herramientas para programadores del sector. En este sentido, la aparición de Unity, un motor de edicion de bajo coste que aporta herramientas utilizadas por grandes empresas, ayuda a la inclusión en este sector de pequeñas empresas independientes.

Dentro del sector de los videojuegos, cobra cada vez más protagonismo la opción de ofrecer multijugador online. Aunque las capacidades de Unity son muy amplias, con una comunidad que desarrolla y publica continuamente nuevas herramientas a través de la plataforma Unity Asset Store, la realidad es que las posibilidades a la hora de desarrollar entornos multiusuario que utilicen este motor no son tan numerosas.

Por ello, en este trabajo nos dedicaremos al estudio de las dos extensiones más utilizadas y potentes para la creación de entornos multiusuario online, como son Photon Unity Networking y la más reciente Extremality Virtual World Framework.

1.2 Objetivos

El principal objetivo de este trabajo es estudiar las posibilidades de creación y gestión de entornos multiusuario en Unity. Principalmente, comprenderemos por qué necesitamos herramientas externas a las incluidas nativamente en el editor, y qué nos ofrecen dichas herramientas.

En primer lugar, estudiaremos el funcionamiento de ambas herramientas, PUN y Extremality, buscando conocer sus características más básicas.

A continuación, trataremos de crear un mundo virtual sencillo, estudiando así el proceso de creación y gestión de dichos entornos y sus posibilidades.

1.3 Metodología

El primer paso a la hora de realizar el estudio, será conocer la plataforma sobre la que trabajaremos, el editor de Unity. En el capítulo 2 introduciremos dicho motor, y conoceremos el método de trabajo. También explicaremos los orígenes de la extensión PUN, el por qué del desarrollo de esta herramienta y su funcionamiento, y la herramienta Extremality.

Lamentablemente, la documentación sobre Extremality es muy escasa. No obstante, dicho asset viene con un proyecto de ejemplo bastante simple. En el capítulo 3, conoceremos el funcionamiento de dicha extensión, a partir del análisis del mencionado proyecto de ejemplo.

A continuación, en el capítulo 4 procederemos a la creación de un proyecto sencillo utilizando PUN y Extremality, para comprobar el proceso de creación y gestión de un entorno multiusuario con estas herramientas.

Por último, realizaremos un análisis de las posibilidades actuales de la creación de estos entornos multiusuario en Unity, comentando también la evolución y desarrollo futuro de las herramientas utilizadas.

Capítulo 2. Estado del arte

En este capítulo conoceremos más a fondo la herramienta Unity, el motor multiplataforma objeto de nuestro estudio. Comenzaremos viendo sus orígenes, hablaremos de las características que lo hacen tan especial y explicaremos un poco su funcionamiento.

A continuación, exploraremos un poco PUN, el principal asset de Unity para conseguir conexiones en línea multiplataforma.

Por último, también echaremos un primer vistazo en este capítulo a Extremality, el asset de Unity sobre el que trabajaremos principalmente en los capítulos 3 y 4, y que pretende facilitar el uso de PUN y de Unity y proporcionar una herramienta sencilla para crear mundos virtuales.

2.1 Unity

2.1.1 Orígenes

La aventura de Unity Tecnhologies [1] empezó en el año 2004 cuando David Helgason (CEO), Nicholas Francis (CCO), y Joachim Ante (CTO) en Copenhague, Dinamarca, lanzaron su primer videojuego, 'GooBall'. El juego no tuvo el éxito esperado, pero por el camino habían creado unas herramientas muy potentes, que decidieron continuar desarrollando con la idea de democratizar el desarrollo de videojuegos, es decir, crear un motor de videojuegos que pudieran utilizar por igual tanto pequeñas como grandes empresas. Un entorno amigable para programadores, artistas y diseñadores, que pudiese llegar a diferentes plataformas sin tener que programar específicamente para cada una de ellas.

En un principio, el motor Unity estaba disponible sólo en Mac. Contaba con dos versiones, Indie y profesional, que presentaban un coste de 300 y 1.500 dólares respectivamente. No sería hasta 2008, con el lanzamiento del iPhone, cuando Unity comenzaría a crecer de forma imparable. Aprovechando el auge de los smartphones, desarrollaron compatibilidad con estas plataformas, y convirtieron su versión Indie en una versión totalmente gratuita, con la que se permitía distribuir el juego igualmente.

Los desarrolladores independientes no tardaron en abrazar la idea. Hoy en día, muchos estudios pequeños y medianos ni siquiera se plantean el crear tecnología propia desde cero (más que en los casos puntuales en los que se requiera algo muy específico y raro), es más económico y rentable adquirir un par de licencias de Unity y ponerse a trabajar. Además de la existencia de una creciente comunidad de usuarios dispuestos a echarse una mano, compartir código e intentar ayudar en los problemas habituales de desarrollo.

Poco a poco se fueron lanzando nuevas versiones, introduciendo más herramientas que los estudios de alta gama suelen tener a su disposición, con el fin de conquistar el interes de desarrolladores más exigentes a la vez que proporcionaba a empresas más pequeñas el alcanzar una calidad difícilmente asequible con otras herramientas.

2.1.2 Características principales

Como hemos visto anteriormente, una de las principales características de Unity es su compatibilidad multiplataforma, ya que permite desarrollar para PC, Mac, Linux, iOS, Android, BlackBerry, Oculus Rift, Playstation, Xbox, Wii, Web, entre otras, sin tener que realizar un trabajo de programación específico para cada plataforma.

Otra de las características que hacen de Unity un buen sistema de trabajo, es la oferta de herramientas que ofrece, que permite trabajar la programación y el diseño artístico de manera simultánea en un mismo entorno de trabajo. Unity no solo aporta herramientas para ambos sistemas de trabajo, sino que permite importar y exportar objetos con programas externos como Maya, Softimage, Blender, Cinema 4D o Photoshop, además de contar con la Asset Store, una tienda online que cuenta con gran cantidad de material para usar en proyectos propios, como modelos 3D, sonidos, partículas, sprites para menús, incluso plugins que extienden el motor.

2.1.3 Introducción a su funcionamiento

En Unity se trabaja sobre proyectos, y así mismo, sobre escenas, donde se pueden visualizar los avances. [2] Cada escena puede considerarse un nivel del proyecto en cuestión, conteniendo los *GameObjects* del juego. Una de las mayores cualidades de esta plataforma es la arquitectura por componentes, que podemos definir a grandes rasgos como la capacidad de añadir o quitar componentes a los *GameObjects*, otorgándoles así nuevas funcionalidades o características, convirtiendo así dicho *GameObject* en nuestra cámara, una fuente de luz o un personaje que reaccione de una forma determinada.

Figura 1. Editor de Unity.

En la figura 1 vemos un ejemplo de la pantalla de editor de Unity. En el centro tenemos la pantalla de escena, donde podemos previsualizar los contenidos de dicha escena y editar gráficamente algunas de las características de los *GameObject* de dicha escena. Pulsando el botón de Play situado encima de esta pantalla, podremos testear nuestra escena en tiempo real sin necesidad de compilar. Dicho testeo se visualizará tambien en esta ventana, en la pestaña *Game,* visualizando la escena a través del objeto definido como cámara*.*

A la izquierda tenemos la ventana Árbol de la escena, donde podemos ver los objetos de la escena actual y su jerarquía actual. Un *GameObject* puede tener una jerarquía de otros *GameObject* como hijos, de forma que si se modificase alguno de los componentes del objeto definido como padre, como la posición, rotación o tamaño, se modificaría también el de los hijos en dicha jerarquía.

A la derecha, en la ventana Inspector, se muestran todas las características y propiedades del *GameObject* seleccionado, y permite modificar y configurar dichos atributos.

Y por último, en la ventana inferior, podemos ver los assets contenidos en nuestro proyecto. Los *assets* [3] son todos los distintos tipos de recursos que podemos utilizar, e incluyen:

- Las propias *escenas* de nuestro proyecto.

- Elemenos de Geometría 3D, tanto *mayas* como *animaciones* de dichos modelos. También incluye mayas no visibles que permiten configurar las colisiones con el motor de físicas integrado de Unity, el motor PhysX de Nvidia, y los *materiales* que definen el aspecto de dichas mayas.

- *Clips de audio*, con los que añadir músicas y efectos de sonido.

- *Scripts*, que permiten personalizar el comportamiento de los objetos del juego mediante lenguaje de programación.

- Los *prefabs* que vayamos creando a lo largo del proyecto. Los prefabs definen un prototipo de *GameObject* con unos componentes predefinidos, que sirven como prototipado para la creación de grupos de objetos basados en él.

La programación en Unity funciona bajo el *.NET framework* de Microsoft y permite programar tanto en JavaScript como en C# o Boo, de forma independiente (los scripts del juego pueden estar en cualquiera de los 3 lenguajes). En este caso, a lo largo de nuestro trabajo, utilizaremos C#. Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos se encapsulan dentro de definiciones de clase. Además, Unity utiliza un modelo de programación por componentes, lo que simplifica el paso de mensajes entre objetos de distintas clases y ofrece mucha versatilidad, ya que objetos diferentes pueden compartir scripts.

2.2 Photon Unity Networking (PUN)

[4] PUN es un asset gratuito de la Unity Asset Store que permite añadir multijugador online a nuestros proyectos de forma sencilla y entre distintas plataformas, a través de la llamada Photon Cloud, que permite conectar hasta 20 clientes sin coste alguno en un servidor gratuito, y hasta 200.000 usuarios creando un servidor propio a cambio de una cuota mensual. A su vez, dichos servidores se organizan en la creación de distintas salas internas.

La principal ventaja que supone usar PUN respecto a la herramienta multijugador nativa de Unity, reside en la arquitectura cliente-servidor (Unity utiliza originalmente conexiones P2P),que minimiza el consumo de recursos y garantiza una baja latencia y alta fiabilidad. Además, Unity Networking no cuenta con compatibilidad con dispositivos móviles, mientras que PUN+, una extensión de pago de PUN, sí cuenta con ella.

2.2.1 Funcionamiento

Una vez descargamos el PUN asset en nuestro proyecto, se lanza el setup wizard, donde se nos pide que introduzcamos el ID de nuestro servidor, o nos ayuda a crear uno si aun no tenemos ninguno. Dicha ID se guarda entonces en el script *PhotonServerSettings*, donde se incluyen los datos necesarios para que nuestro proyecto se conecte correctamente.

La clase más importante es la llamada *PhotonNetwork*, que funciona igual que la clase Network de Unity y permite gestionar todo el proceso de conexión a través de ella, como la creación o busqueda de una sala, y también la creación de *GameObjects* cuando nos unimos a dicha sala o realizamos alguna acción en particular. También asigna de forma automática un ID a cada usuario al conectarse a la sala, lo que permite reconocerlos entre ellos.

La actualización de las características de un *GameObject* según nuestras acciones se realiza mediante la clase *PhotonView*, que intercambia dicha información con el servidor. PUN incorpora scripts configurables con esta clase, de manera que al asignarlo a un *GameObject* podamos declarar de manera rápida y sencilla qué componentes queremos que se envíen, e incluso permite alterar el intervalo de actualización de dicho componente.

2.3 Extremality Virtual World Framework

Extremality Virtual World Framework [5] es un asset cuyo objetivo es ayuda a crear mundos virtuales multiplataforma de forma sencilla, proporcionando las herramientas necesarias para ello y una estructura predefinida, facilmente configurable segun nuestras necesidades y exigencias.

El asset de Extremality incluye scripts escalables y configurables, utilizados en una arquitectura que se basa en el cambio de estados de dichos scripts, que podemos modificar según el número de escenas y usuarios que necesitamos y según las características a tener en cuenta de cada uno de ellos.

Comprobaremos más a fondo el funcionamiento de esta herramienta en los siguientes capítulos, ya que supone la base sobre la que hemos trabajado en este proyecto.

Capítulo 3. Funcionamiento de Extremality

En éste capítulo trataremos de explicar a modo de introducción cómo funciona y que nos ofrece el asset de Extremality Virtual World Framework de Unity. Como vimos en el capítulo 2, se trata de un conjunto de scripts configurables, que definen una arquitectura de trabajo ya predefinida para la creación y gestión, de forma rápida y sencilla, de una sala virtual online.

3.1. Instalación

Para poder utilizar el asset de Extremality Virtual World Framework, debemos de cargar también otros assets en nuestro proyecto, tanto originales de Unity, como alguno de la Asset Store, como es el caso de PUN. Como hemos comentado en el capítulo 2, la función de esta herramienta es aportar una base fácilmente configurable con la que poder hacer uso de estos otros assets de forma sencilla. En concreto, para empezar a trabajar con Extremality, debemos seguir los siguientes pasos iniciales:

- Tras crear el proyecto desde cero, descargamos Extremality de la Asset Store.

- Desde el menú de Assets de Unity, importamos los Standard Assets (paquetes de assets que vienen por defecto con el editor de Unity) de Characters, Cameras, CrossPlatformInput y Utility.

- Por último, descargamos de la Asset Store el Photon Unity Networking, y realizamos la configuración inicial con el wizard.

En el capítulo 2 vimos que para utilizar PUN debemos registrarnos en su página y adquirir uno de los servicios que ofrecen para usar su nube. En concreto, utilizaremos el servicio gratuito, que nos permite conectar hasta 20 usuarios. La creación y gestión de dicho servidor en la nube es sencilla; la primera vez que se accede, basta con escribir una dirección de correo válida en el wizard de PUN para que se nos registre con una cuenta válida, se cree el primer servidor y se configure automáticamente en nuestro proyecto de Unity. Si ya estamos registrados, debemos acceder a la página web de Photon Cloud, donde se puede gestionar nuestros servidores, para conocer el ID de nuestro servidor, e introducirlo en el wizard inicial de PUN para que se configure la conexión con nuestro servidor de forma automática.

Con el asset de Extremality tenemos dos carpetas, Examples y Core. En Examples tenemos un proyecto de ejemplo, que nos permite ver un principio de las posibilidades de la herramienta y cómo funciona. En la carpeta Core encontramos los scripts y objetos que utilizaremos. En concreto, nos interesan los prefabs, que incluyen todos los objetos que usaremos en las distintas escenas de nuestro proyecto para crear un mundo virtual. Estos objetos ya tienen scripts asignados, que son fácilmente configurables desde la ventana Inspector del editor de Unity.

3.2. Funciones principales

Para comenzar a trabajar en un nuevo proyecto, necesitamos una primera escena, donde crearemos la interfaz inicial de nuestra aplicación, y en la cual también configuraremos el flujo de trabajo en que se basa Extremality para la creación y gestión de nuestras salas online. En esta escena inicial, por la configuración de algunos de los scripts preprogramados que controlan la arquitectura de trabajo de Extremality, debe de nombrarse *Extremality SDK Build.* A continuación, añadiremos el prefab llamado *Extremality*, que es el que contiene todos los *GameObject* que nos permitirán configurar nuestra aplicación y cómo se conecta con los servidores, y cada uno de los cuales pasamos a explicar más detalladamente.

3.2.1. Arquitectura de trabajo interna de Extremality

Como hemos mencionado previamente, Extremality es una herramienta con una arquitectura de trabajo basada en estados. Dicho flujo de trabajo viene preconfigurado y controlado principalmente mediante 3 elementos.

En primer lugar, tenemos el *XRProcessManager* y el *XRSettingsManager,* cuyos scripts asignados simplemente indican la ubicación de los archivos de la aplicación a nivel local y de los demás usuarios conectados al Photon Cloud, lo que permite comprobar que sus versiones sean la misma y no hayan incompatibilidades.

A continuación, tenemos el *XRSimulationManager*, que tampoco necesitamos configurar pero es una de las bases de Extremality, ya que controla el flujo de trabajo, intercambiando información entre los distintos scripts y realizando las llamadas a los distintos métodos de cada uno cuando es necesario, y encargándose de que sólo esté activo en cada momento el script adecuado, deshabilitando los que no se requieran en cada momento.

3.2.2. Gestión de la interfaz

El siguiente objeto que encontramos es el *XRUIManager*, cuyo script controla principalmente toda la interfaz de nuestro programa, tanto en el menú al iniciar la aplicación como la interfaz que encontramos dentro de una sala online.

Al tratarse del elemento que controla la interfaz, nos encontramos con que presenta una jerarquía padre-hijo con un *Canvas*, que como sabemos es el *GameObject* que controla los elementos de la UI, y a su vez con distintos paneles que se pueden configurar gráficamente desde el editor de Unity. Dichos paneles se corresponden con la ventana de login en la que se nos pedirá información de usuario para entrar en el servidor (nombre y, si lo exigimos, password), el panel de fondo del menú principal, el listado de los escenarios con los que podremos crear salas de nuestra aplicación, otra lista con las salas ya creadas dentro de nuestro servidor, y un último listado con los diferentes modelos que podemos elegir para nuestro personaje, si existen varias posibilidades. Si añadimos otras opciones previas a la conexión con una sala en nuestra aplicación, también se realizará su elemento de la UI añadiendo paneles a este canvas. También encontramos en esta jerarquía los elementos de la interfaz del chat una vez entramos en una sala.

Tras configurarlos a nuestro gusto, si pretendemos obtener información de entrada o salida respecto de alguno de estos paneles, cuadros de texto o botones, como por ejemplo la información de login, debemos configurar en el script asignado a *XRUIManager* su función en el recuadro asignado, como podemos ver en la siguiente imagen:

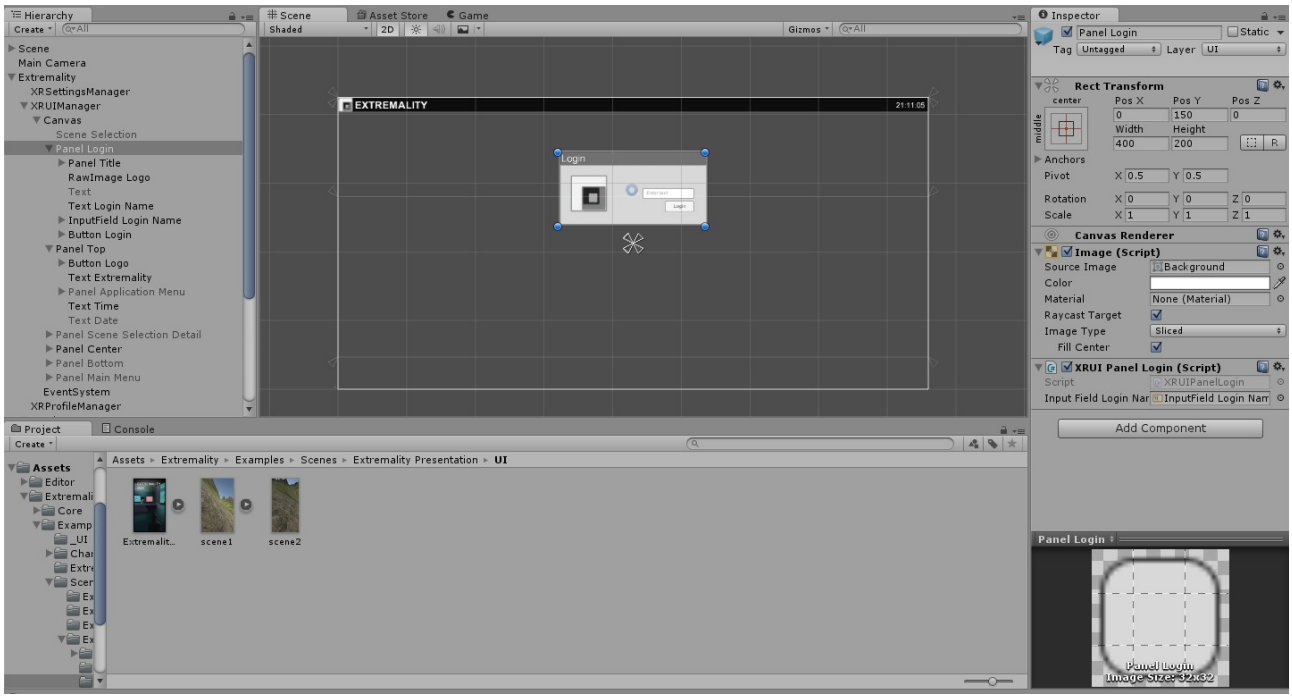

Figura 2. Edición de la interfaz gráfica, con la configuración del XRUI Manager a la derecha.

3.2.3. Gestión del entorno online.

El control de la interacción de los usuarios entre sí y con el servidor se realiza mediante el siguiente grupo de elementos.

En primer lugar, nos encontramos con el *XRProfileManager*, un script cuya funcionalidad básica es guardar nombre e información de usuario, como podría ser una foto de perfil, e intercambiarla con el servidor cuando dicha información es solicitada.

El siguiente elemento es el *XRNetworkManager*, cuyo script es el encargado de gestionar la conexión con la Photon Cloud y el intercambio de información con el servidor, es decir, el proceso de conexión y desconexión con el servidor y de crear o unirse a una nueva sala. Principalmente, todas acciones necesarias que empleen el método PhotonNetwork se encontrarán en este script.

Nos encontramos ahora con el *XRInstanceManager*, elemento que no requiere ninguna configuración y cuya función es intercambiar información con el servidor sobre las salas creadas actualmente y sus usuarios.

Por último, la función del *XRChatManager* es controlar los comandos de PUN necesarios para el intercambio de mensajes escritos con los demás usuarios que se encuentren en la misma sala, incluyendo tanto mensajes de destino global como mensajes privados a un solo usuario, todo gestionado por dicho script sin tener que configurar nada.

3.2.4. Gestión de cámaras

El *XRCameraManager* es el encargado del funcionamiento de las cámaras. Aunque el comportamiento de la cámara o grupo de cámaras se modifica y configura dentro de cada escena, el objetivo de este script es asignar como objetivo de la camara en nuestra aplicación a nuestro personaje cuando nos conectemos a una sala. En este script además debemos configurar el número de cámaras que se utilizará en cada escena cuando se utilice más de una, e indicar cuál es la principal.

3.2.5. Gestión de personajes.

El *XRCharacterManager* es un objeto más interesante, ya que es en su script asocidado donde indicaremos la cantidad de modelos de personaje entre los que podemos escoger uno para nosotros. Existe un primer parámetro configurable desde el Inspector, llamado *Size,* en el que introducimos el número de elementos de la lista, es decir, el número de modelos. A continuación, para cada modelo debemos elegir su nombre, una imagen, y el *prefab* que se utilizará para dicho personaje. También se puede añadir una pequeña descripción y una imagen previa.

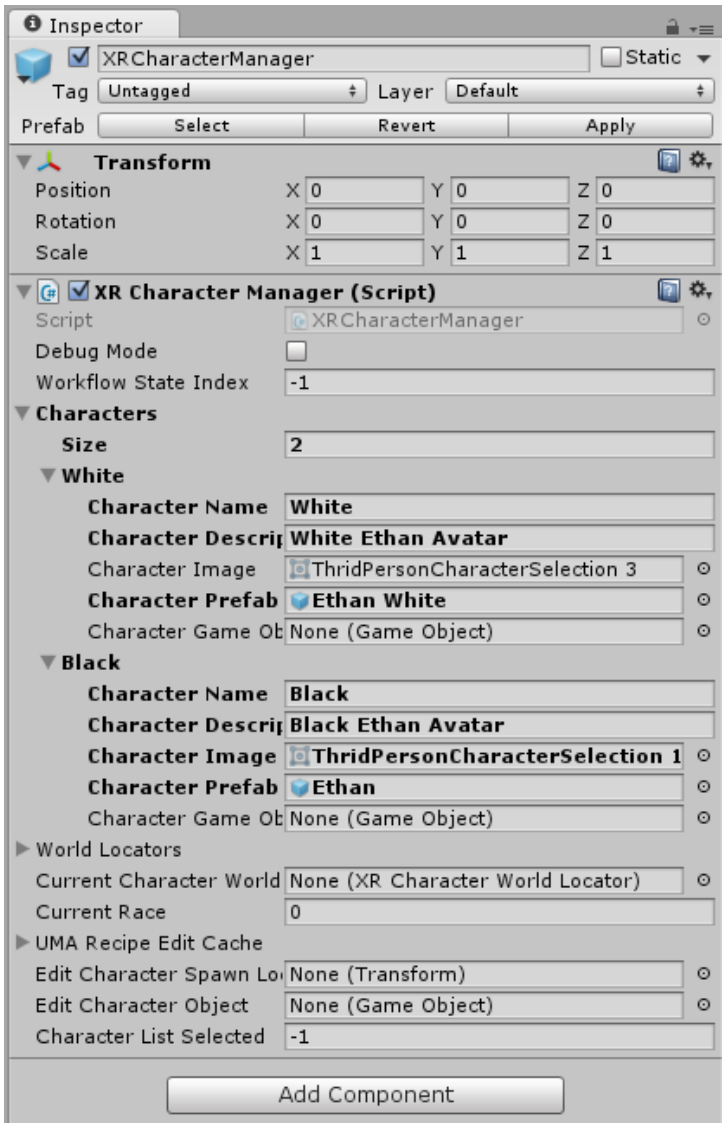

Figura 3. Configuración de XR Character Manager.

3.2.6. Gestión de escenarios

El *XRSceneManager* funciona de forma similar al elemento anterior, y sirve para indicar las escenas que se utilizarán en nuestra aplicación. Tras indicar el tamaño de la lista, para cada escena indicamos el nombre de la escena para nuestro programa, el nombre que recibe en nuestro editor, y podemos añadir una descripción o una imagen previa. También debemos indicar, en *SceneType*, si se trata de una escena independiente de este flujo de trabajo, o si se trata de una escena que inicia un flujo de trabajo distinto de Extremality (como podría suceder, por ejemplo, si queremos cambiar la nube de Photon a la que nos conectamos, para disponer de más espacio para usuarios), en cuyo caso deberíamos indicar también el nombre de la escena con la nueva configuración de Extremality en *AssetBundleName*.

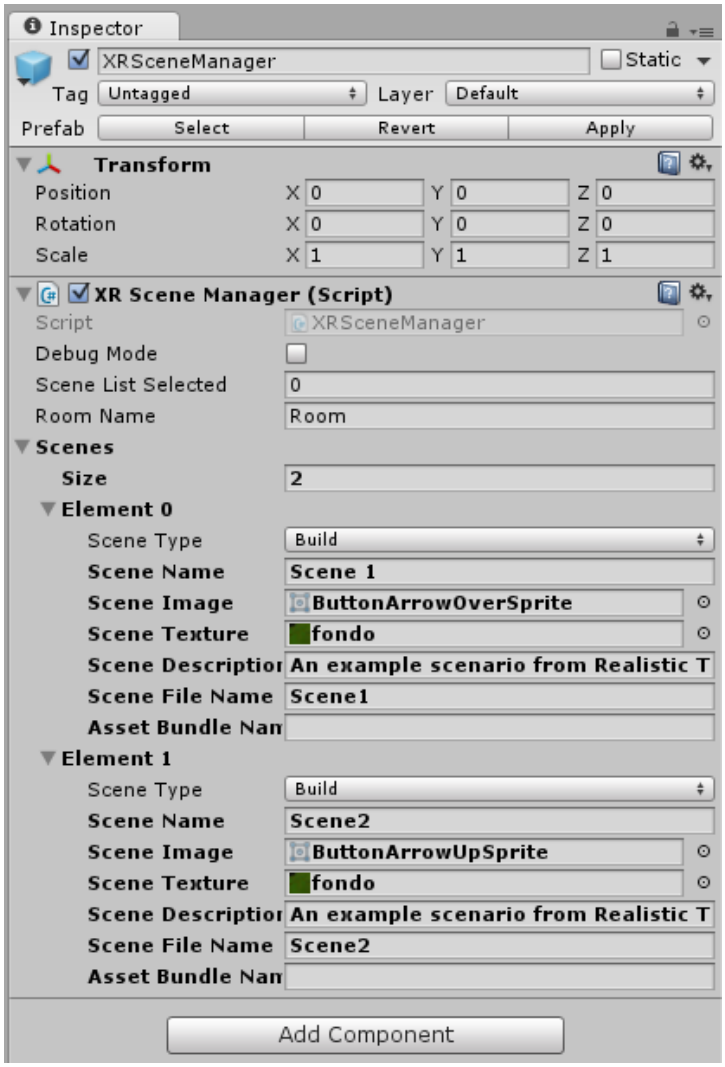

Figura 4. Configuración del script XR Scene Manager.

3.3. Otros aspectos

Una vez hemos configurado todo lo necesario en esta escena principal, aún debemos tener presente algunos detalles de configuración, tanto en el diseño de las escenas que utilizaremos como en los personajes y demás elementos interactivos que hayan en nuestras escenas.

3.3.1. Configuración de escenas

En el caso de las escenas, debemos añadir dos objetos a cada una de ellas: el *SpawnStart* y *XRLevelManager*. El *SpawnStart* es simplemente un *GameObject* con solo un vector de posición, donde aparecerá nuestro personaje al entrar en la sala. El *XRLevelManager* es el objeto encargado de crear tu personaje a partir del prefab preseleccionado, asignarle un ID dentro del servidor, y adjudicarnos el control de dicho personaje. También deberemos de asignar el script *XR Camera* a nuestra cámara principal, para que se comunique con el *XRCameraManager* de nuestra escena principal, y de esta forma se le indique a qué personaje debe definir como *target* para seguirlo.

3.3.2. Configuración de elementos interactivos

En cuanto a los elementos con los que se vaya a interactuar en nuestra escena, debemos asignarles el script *XR Character Definition* y el *Photon View* correspondiente para intercambiar la información necesaria con el servidor. Por ejemplo, en el caso de nuestros personajes, tendrán asignados el *Photon Transform View* y el *Photon Animator View*, para que se intercambie información respecto a la posición y la animación de los personajes. Éstos scripts son propios de PUN, y se pueden configurar de forma fácil desde el Inspector del editor de Unity, para indicar qué parametros de dichos componentes compartir y el intervalo de actualización de dicha información. En la siguiente imagen podemos observar la configuración de dicho ejemplo:

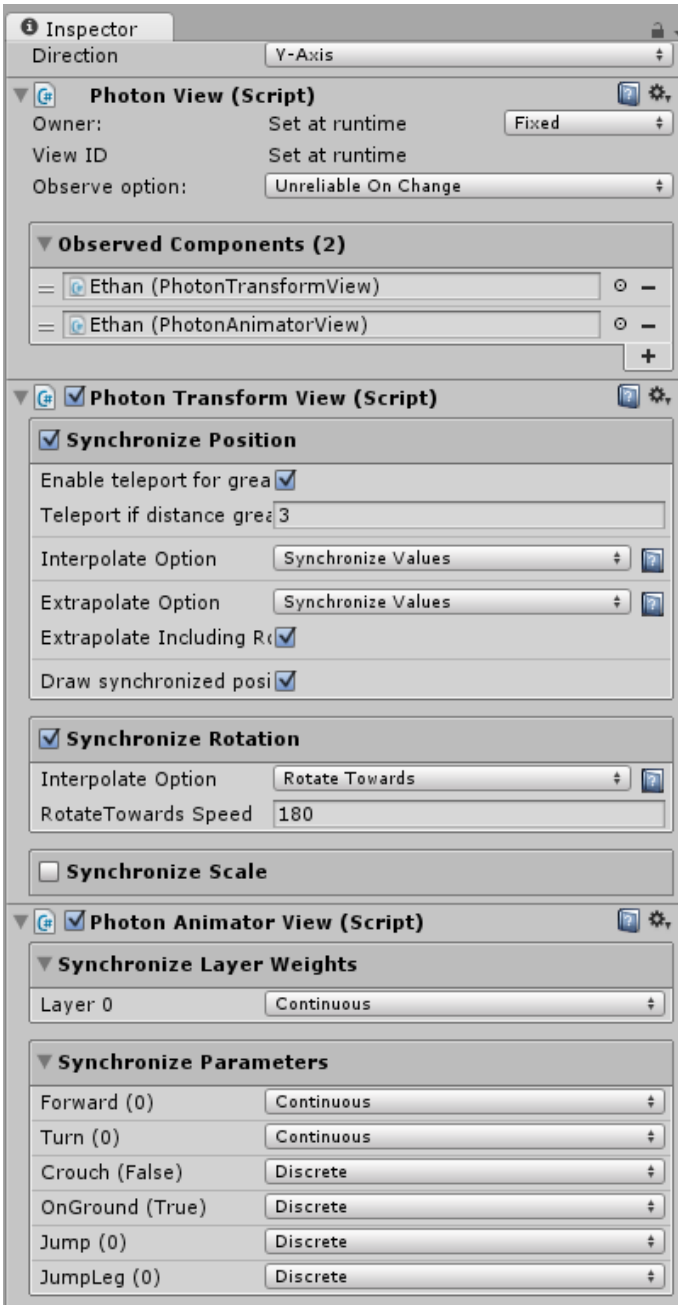

Figura 5. Ejemplo de configuración de Photon View en un personaje.

Como hemos visto, la función principal de todos los elementos de Extremality es intentar gestionar todo lo necesario para desarrollar un entorno online de forma sencilla y autónoma, configurando sólo algunas opciones y utilizando todo el potencial del Asset de Photon para Unity.

Capítulo 4. Creación de un mundo virtual en Unity.

En este capítulo, crearemos un entorno multiusuario en Unity mediante los assets de PUN y Extremality, siguiendo las indicaciones explicadas en el capítulo anterior. No dedicaremos mucho esfuerzo a la creación del escenario y el modelado de los personajes, ya que el objeto del estudio no es sino la creación del entorno y sus posibilidades.

4.1. Creación del servidor en Photon Cloud.

Como comentamos en el punto 2.2, uno de los primeros pasos es registrarnos en el servicio de Photon Cloud para poder configurar nuestro servidor en su nube. Mientras instalamos el asset de Virtual World Framework de Extremality, siguiendo los pasos indicados en el apartado 3.1 de este informe, nos salta el wizard de PUN, mostrando una pantalla parecida a la de la siguiente imagen:

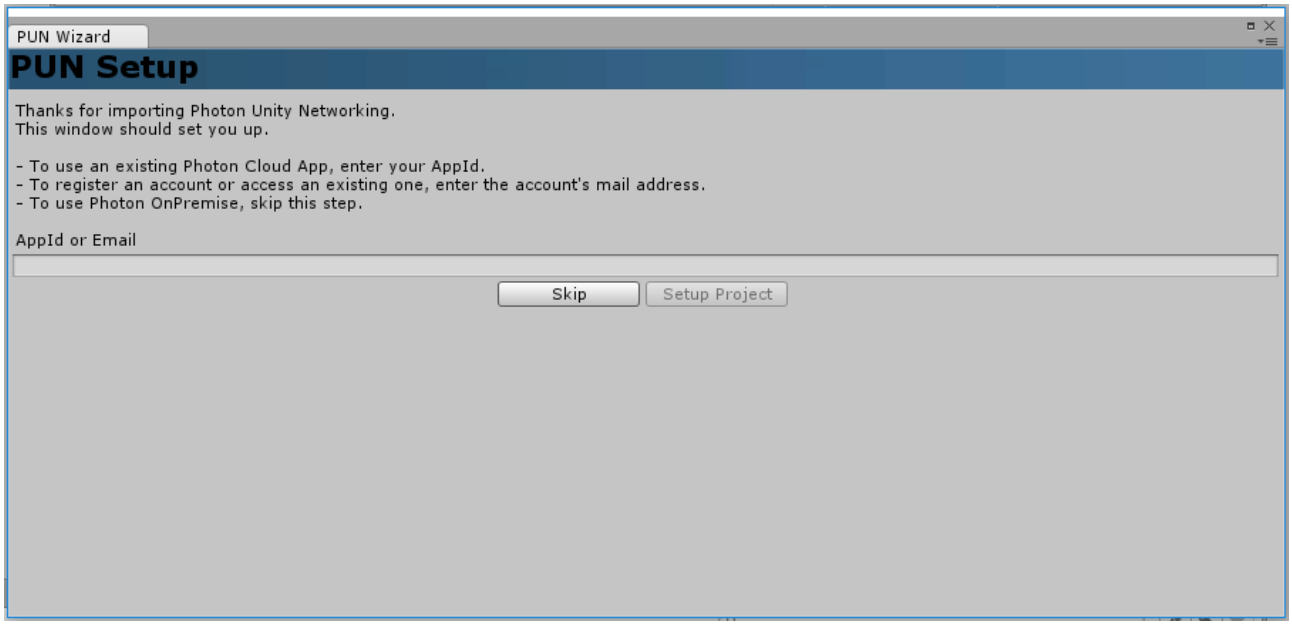

Figura 6. Wizard de configuración del asset PUN.

Si no estamos registrados en su servicio, basta con introducir la dirección de correo con la que queremos registrarnos. De esta forma, se creará automáticamente un servidor con el servicio gratuito, que consta de conexión para hasta 20 usuarios y un máximo de 3 salas, y se configurarán los archivos de nuestro proyecto de Unity con la AppID que se asocie a dicho servidor tras ser creado. En caso de querer gestionarlo, o estar ya registrado y querer buscar el AppID asociado a nuestro servidor, basta con acudir a su página web, photonengine.com, y logearse con la dirección escogida y una contraseña que encontraremos en nuestro correo. Tras ello, en la sección Dashboard, encontramos un listado de nuestro servidores en photon, de una forma similar a la que observamos en la figura siguiente:

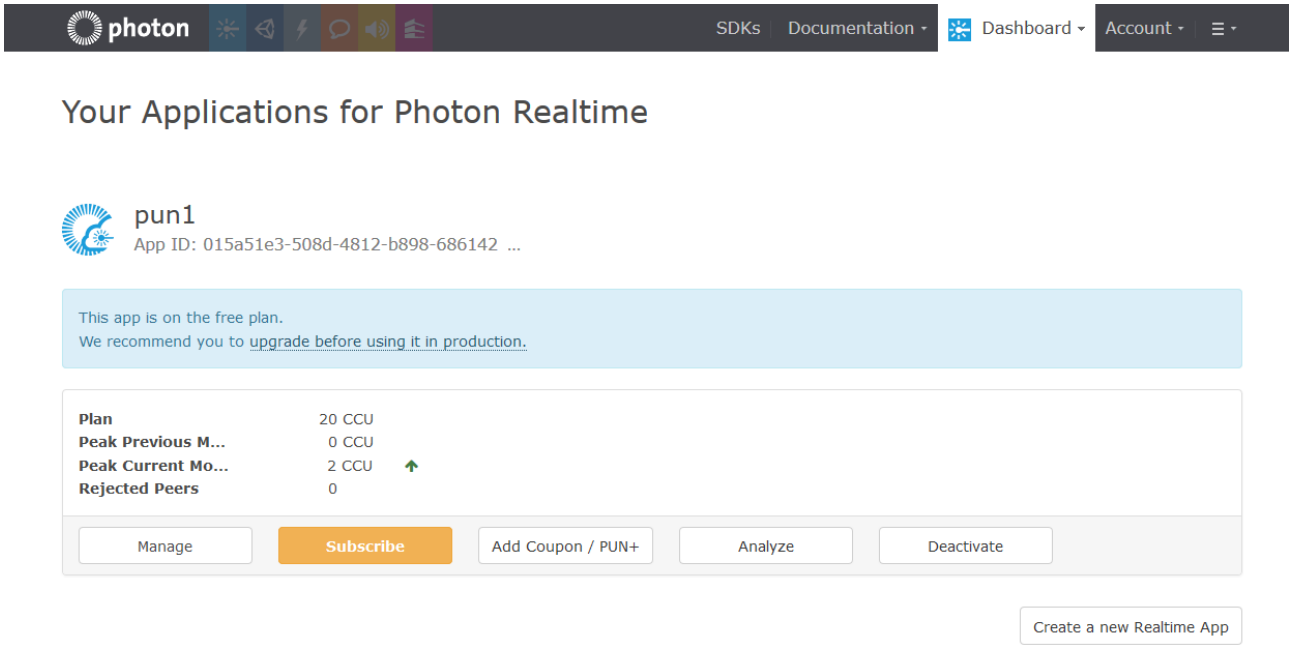

Figura 7. Gestión del servidor en photon desde la web.

Pulsando en *Manage*, tenemos acceso a la AppID completa que debemos introducir en el wizard de PUN, y también podemos realizar algunas gestiones básicas, como cambiar el nombre del servidor, el tipo de suscripción, o comprobar el número de usuarios y salas creadas en las últimas 48 horas.

4.2. Creación de las escenas

El siguiente paso será crear y configurar las escenas de nuestro proyecto, para el cuál desarrollaremos tan sólo 3 escenas: la escena principal, con la configuración de Extremality y su interfaz gráfica, como vimos en el capítulo 3, y dos posibles escenarios para nuestro mundo virtual.

4.2.1. Escena principal

Nuestra escena principal deberá llamarse *Extremality SDK Build,* como explicamos en el punto 3.2, y contendrá el prefab de Extremality analizado en dicho apartado, compuesto por los elementos que gestionan y configuran el flujo de trabajo de nuestro proyecto.

En cuanto a la interfaz de nuestro proyecto, configuraremos un menú principal bastante simple. Presentaremos una ventana principal, en la que se nos pedirá un nombre de usuario que nos identificará cuando nos conectemos a una sala del servidor. Este panel sera enlazado con la opción llamada *Panel Login* en el *XRUIManager.*

Una vez conectados al servidor, aparecerá nuestro menú principal, en el editor llamado *Panel Center.* Por defecto, este panel se divide a su vez en tres partes: *Panel Main Menu, Panel Top* y un *Panel Selection* para cada tipo de elemento configurable. En el *Panel Main Menu,* en la parte inferior de la pantalla, se nos dispondrán mas o menos botones según las características que activemos. También en este panel se encuentra el botón para conectarse al servidor con las opciones escogidas.

En el *Panel Selection*, se dispondrán en forma de lista tantos botones como opciones dispongamos, dentro de la opción activada en *Panel Bottom*. Hay que crear un *Panel Selection* para cada opción, asignarle el script *XRUI Panel Selection* correspondiente, y asignarle en la opción *Panel Content* del inspector a que botón del *Panel Main Menu* corresponde dicho panel. Con este script, se creará automáticamente un botón por cada opción de dicha lista. Actualmente, Extremality cuenta con opciones para incluir listas seleccionables de escenas, configurable en el elemento *XR Scene Manager*, modelos de avatar, configurable en *XR Character Manager*, y salas ya creadas en el servidor, información a la que se accede en *XR Instance Manager*. También ofrece opción para una lista más, llamada por defecto *Panel Role Selection* y configurable de forma análoga a las escenas y avatares en el script llamado *XR Role Manager*, no inicializada por defecto. Aunque no utilizaremos esta opción en nuestro proyecto, ofrece la posibilidad de añadir una característica seleccionable más, ya sea sobre un componente del modelo de nuestro personaje o una característica del escenario. Además, con unos conocimientos un poco más avanzados del lenguaje C# que utiliza Unity, podemos partir de estos scripts para crear tantas opciones configurables como deseemos.

Por último, en *Panel Top* se nos mostrará información de cada elemento de la lista, como el nombre de la escena o avatar y una pequeña descripción del elemento. Ésta información es recibida también desde el script que configura la lista (por ejemplo, desde el *XR Scene Manager* en el caso de las escenas).

Se puede editar fácilmente la colocación de estos tres paneles en la pantalla, y escoger entre una disposición vertical u horizontal de ellas. Para nuestra aplicación, dejaremos en la parte inferior el *Panel Main Menu*, con los botones de selección en la izquierda y el botón de conexión al servidor en la esquina inferior derecha, los *Panel Selection* en la parte central de la pantalla, y dispondremos del *Panel Top* en la parte superior, como se puede observar en la siguiente imagen:

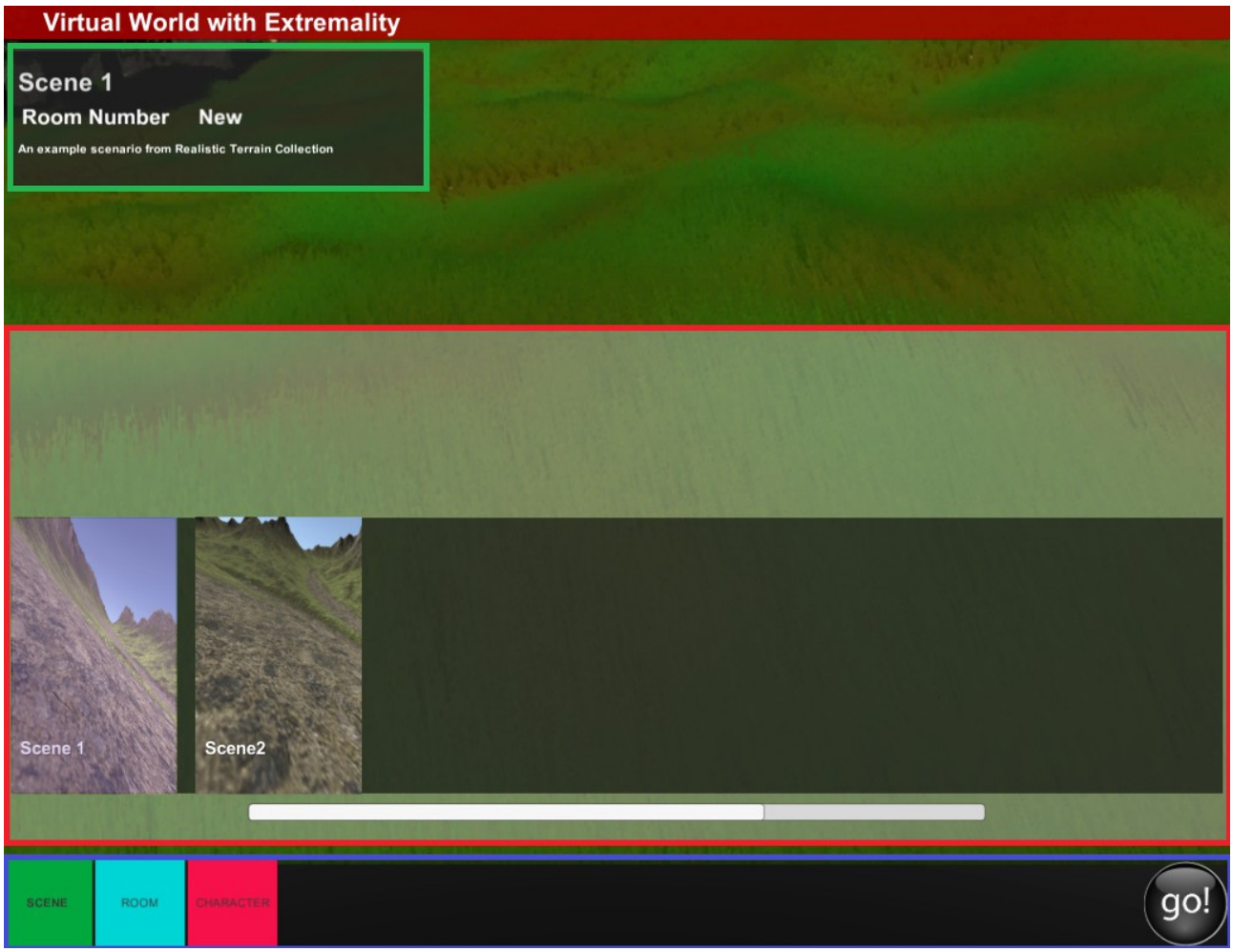

Figura 8. En el cuadro rojo, nuestro *Panel Selection;* en el cuadro azul, el *Panel Main Menu*; y encuadrado en verde el *Panel Top.*

En la figura 8 apreciamos que para nuestro proyecto estarán activadas las opciones predefinidas, *Scene*, *Room* y *Characters. .* Si activamos la opcion *Scene* en el *Panel Main Menu,* se nos listarán los dos escenarios posibles que crearemos para crear una sala nueva; en la opción *Room* se nos mostrarán las salas ya creadas en nuestro servidor para poder unirnos a ellas, y en la opción *Characters*, se dispondrán los distintos modelos que podemos escoger para nuestro avatar. Por último, en *Panel Top* se mostrará información básica sobre la opción escogida, como el nombre del escenario, sala o avatar, o una pequeña descripción de estos.

4.2.2. Escenarios

Para simplificar el proceso de creación, utilizaremos en nuestro proyecto tan sólo dos escenas, que corresponderán a dos posibles escenarios para la creación de una sala. Dichos escenarios serán creados mediante el asset Realistic Terrain Collection. Se trata de un asset gratuito de la Unity Store, que presenta una colección de terrenos previamente creados, con posibilidad de escoger entre texturas en alta y baja resolución. Para nuestro proyecto, escogeremos un terreno en alta resolución para nuestro primer escenario y otro en baja resolución para el segundo.

Como vimos en el punto 3.3.1 de este informe, debemos configurar algunos elementos en ambas escenas para el correcto funcionamiento de la aplicación.

En primer lugar, debemos añadir un *GameObject* que se componga sólo de un vector de posición llamado *SpawnStart*, y el elemento *XRLevelManager* de Extremality, que será el encargado de crear nuestro personaje en dicho *SpawnStart*, y de adjudicarnos su control y un ID dentro de la sala.

También debemos configurar la cámara principal de nuestra escena, añadiendo el script *XR Camera* para que se pueda comunicar con el *XR Camera Manager* de la escena principal, que le indicará, entre otras cosas, que defina nuestro avatar como *target*, es decir, que le siga a una cierta distancia. Para que esta opción sea posible, tenemos que tener en cuenta también que la cámara no tenga ningun objeto definido como *target* de forma predeterminada, y que la opción *Camera Third Person* del script *XR Camera Manager* tampoco tenga asignado ningún objeto. Como añadido, para nuestra aplicación hemos optado por añadir a la cámara el script *XR Smooth Mouse Look*, que permite mover la cámara libremente desde un punto, sin seguir la perspectiva de nuestro avatar, mientras pulsamos el botón izquierdo de nuestro ratón. Dicho punto, debe ser definido por un *GameObject* compuesto por un vector de posición, que situaremos justo detrás de la cámara.

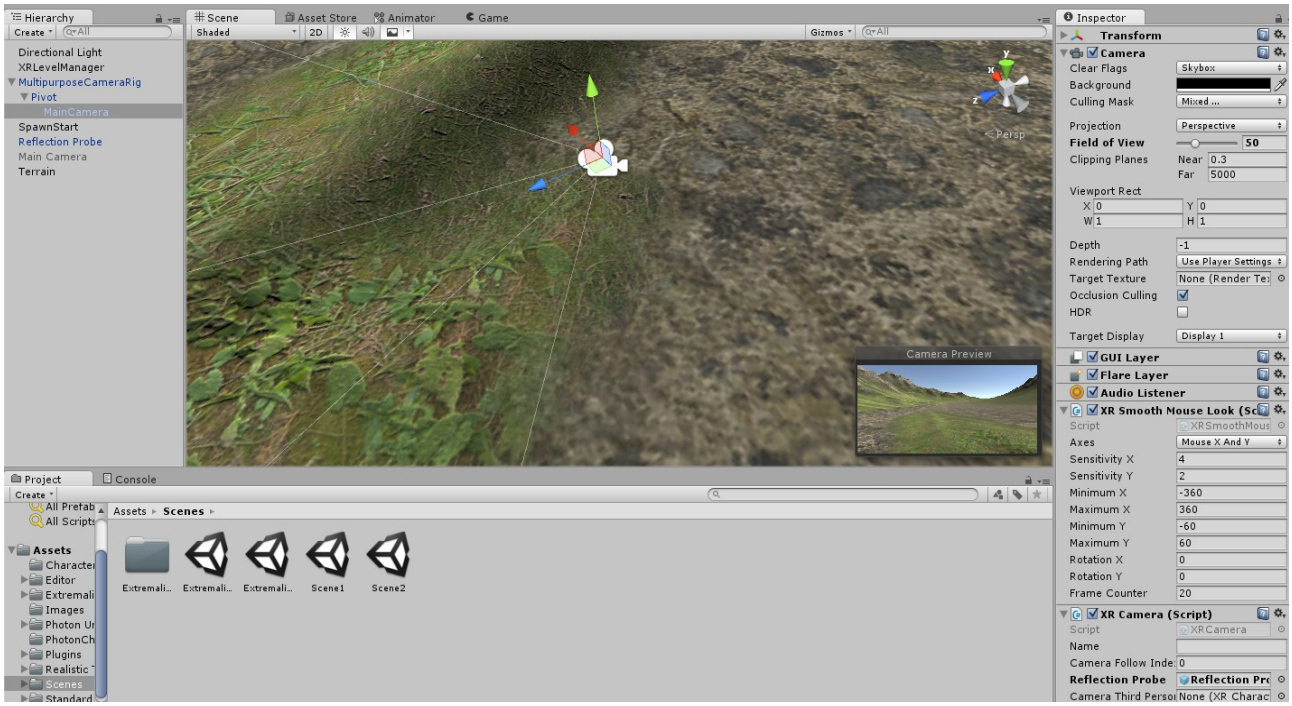

Figura 10. Configuración de la cámara.

Por último, recordar que debemos configurar adecuadamente el script *XR Scene Manager* de la escena principal de nuestro proyecto, como explicamos en el apartado 3.2.6 de este informe.

4.2.3. Configuración de los avatares.

Para la representación de los usuarios en nuestro mundo virtual, y por facilitar un poco el proceso creativo de nuestro proyecto, nos serviremos del avatar Ethan, contenido en los Standard Assets que vienen por defecto. De esta manera, dispondremos sin problemas de un modelo en 3D con scripts que permiten controlarlo y con sus propios archivos de animación. Para poder distinguir entre los distintos usuarios de nuestra aplicación, crearemos 3 prefabs distintos de dicho avatar, con texturas de color negro, blanco y marrón.

Figura 11. Modelos utilizados para los personajes de los usuarios.

A parte de los scripts ya mencionados para poder controlar dichos avatares, debemos recordar añadir los scripts *XR Character Definition* y los correspondientes *Photon View*, en este caso *Photon Transform View y Photon Animator View*, y configurarlos adecuadamente, como vimos en el apartado 3.3.2.

Por último, añadiremos los prefabs el script *XR Character Name*, que hará que se muestre el nombre del resto de usuarios encima de su avatar. Ya sólo queda configurar adecuadamente el *XR Character Manager* de la escena principal, siguiendo la descripción dada en el punto 3.2.5.

Capítulo 5. Presentación y análisis de nuestro entorno virtual

En este capítulo comprobaremos el correcto funcionamiento de nuestra aplicación. Para ello, ejecutaremos nuestro programa desde tres terminales. Cada uno, escogerá un modelo de avatar distinto, y escogeremos como nombres de usuario 'Marrón', 'Blanco' y 'Negro' respectivamente.

El primer paso, será comprobar que, tras crear una sala con uno de los usuarios, efectivamente aparece dicha sala en la lista para el resto de usuarios, y que podemos unirnos a ella. Para ello, crearemos una nueva sala con el usuario 'Marron', de forma que será el usuario escogido como 'admin' o propietario de la sala, y acudiremos al listado de salas ya creadas desde los otros dos usuarios, pulsando sobre el botón *Room* como explicamos en el capítulo 4.

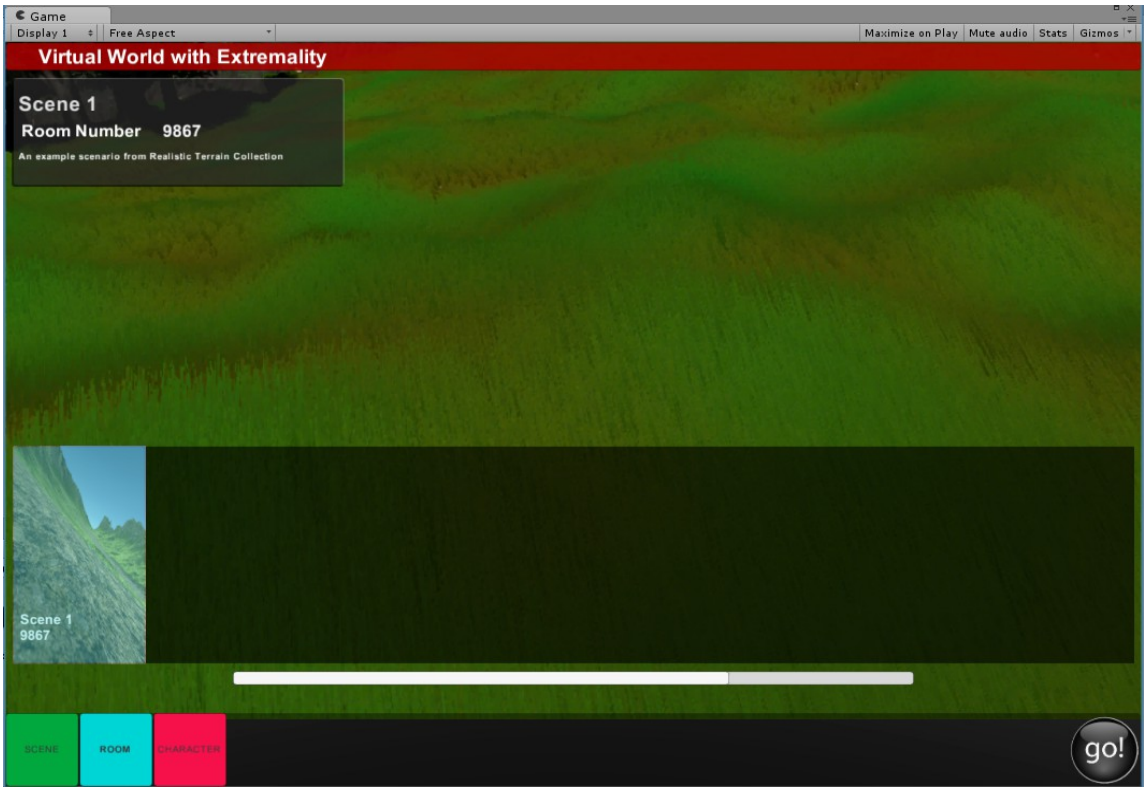

Figura 12. Lista de salas creadas.

Una vez hemos comprobado el correcto intercambio de información con el servidor respecto a las salas, pasamos a unirnos a la sala ya creada con los demás usuarios. Vemos entonces, que efectivamente los tres usuarios están dentro de la misma sala, que podemos ver su movimiento en tiempo real, que la cámara se fija detrás de nuestro avatar, y que aparece un pequeño texto con el nombre encima del avatar de los demás usuarios.

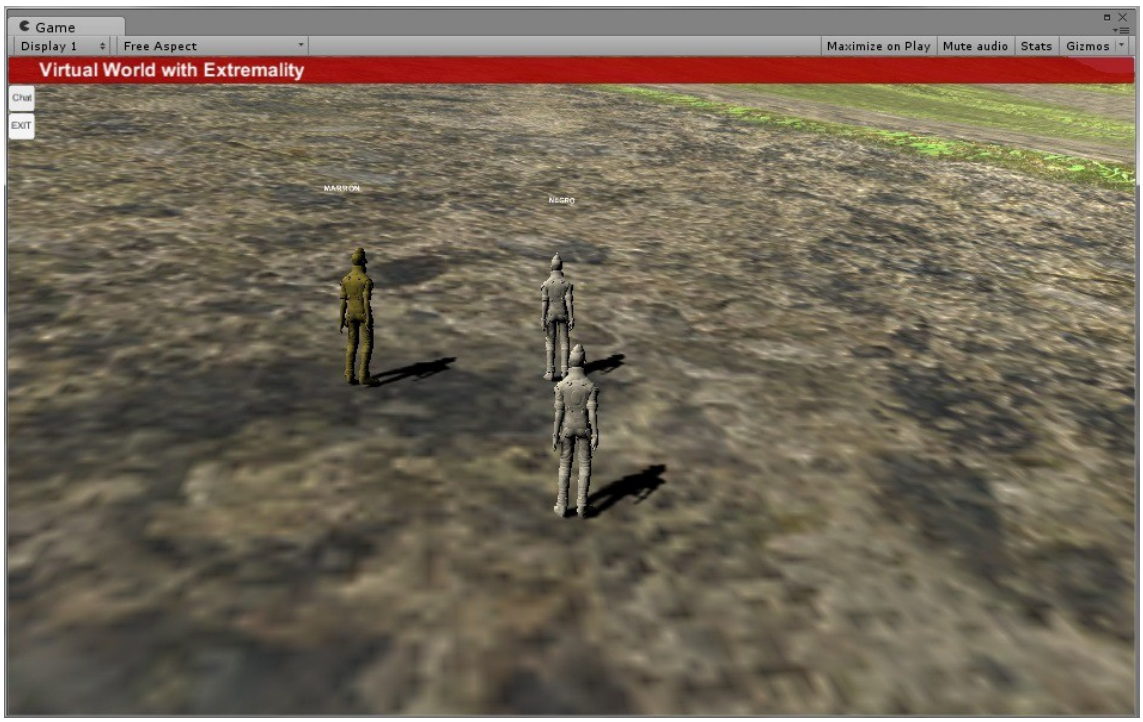

Figura 13. Vista desde el usuario Blanco.

También podemos comprobar, manteniendo el botón izquierdo del ratón y arrastrándolo hacia la derecha, que funciona la posibilidad que asignamos a la cámara de poder moverla libremente.

Figura 14. Movimiento libre de la cámara.

Tanto en la figura 13 como en la figura 14, podemos ver a la izquierda dos botones de la interfaz, uno llamado *Chat* y otro llamado *Exit.* Si pulsamos sobre el segundo, volveremos a la pantalla principal, en la que elegimos a qué sala unirnos y con qué modelo de avatar. Pulsando sobre el botón de *Chat,* aparecerá en la esquina inferior izquierda un nuevo panel, en el que veremos a la izquierda un listado de los usuarios conectados en la sala, y a la derecha se mostrarán los mensajes globales que se han enviado. En la parte inferior de este panel, tendremos un cuadro de texto donde escribir un mensaje de chat a todos los usuarios presentes en la sala, que se enviará tras pulsar el botón *Send.*

Tanto los botones de la parte superior izquierda como el panel desplegable del chat, al igual que el resto de elementos de la interfaz gráfica de nuestra aplicación, se pueden configurar en el *UIManager* de la escena principal de nuestro proyecto. En concreto, los botones pertenecen al llamado *Panel Application Menu,* y el desplegable del chat se puede editar con el *Panel Bottom.*

Por último, comprobaremos el funcionamiento del chat del que acabamos de hablar. Desde el usuario 'Blanco', enviaremos un mensaje por el chat global. Después, el usuario 'Marrón' abandonará la sala. No sólo debería de aparecer en el chat un mensaje de que dicho usuario ha salido de la sala actual, sino que, al tratarse del usuario con el que se creó dicha sala, también saldrá un mensaje de asignación de nuevo 'admin' de la sala. Estos mensajes los vemos en la siguiente imagen, desde el usuario 'Negro', lo que nos permite reafirmar el funcionamiento del chat.

Figura 15. Ventana de chat desde usuario 'Negro'.

Así pues, hemos comprobado que, con pocos y sencillos pasos, se puede crear un entorno multiusuario virtual en Unity con funciones básicas utilizando la herramienta de Extremality Virtual World Framework.

Capítulo 6. Conclusiones del estudio y líneas futuras.

En el último capítulo de nuestro trabajo, repasaremos los resultados de nuestro estudio y comentaremos el plan de desarrollo futuro de Unity y las herramientas analizadas y sus posibles utilidades.

6.1. Resumen del estudio realizado.

Como comentábamos en el primer capítulo del capítulo de nuestro trabajo, el objetivo principal era estudiar las distintas posibilidades de las que disponemos en el ecosistema de trabajo de Unity para crear entornos multiusuario.

A lo largo del capítulo 2, hemos analizado por qué las herramientas de interconexión que vienen internas con el editor de Unity son insuficientes, y cuáles son las más utilizadas. También hemos comentado los principios básicos del funcionamiento de la extensión más empleada para ello, Photon Unity Networking.

En el capítulo 3, hemos analizado el comportamiento del asset Virtual World Framework de Extremality, una herramienta potente que permite crear y gestionar un entorno multiusuario de forma aparentemente sencilla, pero del cual no existe documentación que describa su usa.

Por último, hemos utilizado las conclusiones del análisis del capítulo 3 para crear un sencillo prototipo de entorno virtual. Hemos comprobado la utilidad de la herramienta de Extremality para crear un servidor y realizar gestiones básicas, y cómo proporciona una base fácilmente configurable para dicha gestión. También hemos analizado las posibilidades que ofrece, como la gestión de un chat escrito multiusuario y privado dentro de dicho entorno.

6.2. Líneas futuras.

Como hemos visto, Extremality proporciona una base sencilla de usar y configurar para crear entornos virtuales multiusuario. No obstante, la falta de documentación sobre su funcionamiento, y las limitaciones a la hora de configurar algunos aspectos, pueden llegar a entorpecer su uso.

En el momento de realización de este trabajo, los desarrolladores de la herramienta PUN están trabajando en la posibilidad de implementar un chat de voz a través de Unity, mientras que los creadores de Extremality barajan la posibilidad de compartir videos y presentaciones dentro de sus entornos virtuales. Estas dos opciones añadirían muchas posibilidades al motor de Unity, no sólo con aplicaciones en el ámbito de los videojuegos, sino también con posibilidades en las comunicaciones.

BIBLIOGRAFÍA

[1] “History of the Unity Engine”.
seraphinacorazza.wordpress.com/2013/02/14/history-of-the-unity-engine-

[2] Collado, David. “Empezando en Unity3D”.

[3] Dr. Edward Lavieri. “Getting Started with Unity 5”.

[4] Exit Games. “Photon Unity Networking Intro”.
doc.photonengine.com/en-us/pun/current/getting-started/pun-intro

[5] Extremality
extremality.com/home