Plataforma de videojuegos online Mundo Isla

Online videogame platform Mundo Isla

Rodrigo Valladares Santana

Sección de Ingeniería Informática

Escuela Superior de Ingeniería y Tecnología

Trabajo de Fin de Grado

La Laguna, 6 de septiembre de 2014

D. **Pedro Antonio Toledo Delgado**, con N.I.F. 45725874B profesor Ayudante adscrito al Departamento de Ingeniería Informática de Sistemas de la Universidad de La Laguna y D.ª **Carina Soledad González González**, con N.I.F. 54064251Z profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática de Sistemas de la Universidad de La Laguna

C E R T I F I C A N

Que la presente memoria titulada:

"Plataforma de videojuegos online Mundo Isla."

ha sido realizada bajo su dirección por D. Rodrigo Valladares Santana, con N.I.F. 54112521F.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 3 de septiembre de 2014

Agradecimientos

Gracias a los tutores por su ayuda y orientación a lo largo del desarrollo de este proyecto.

Resumen

El objetivo de este proyecto es añadir una serie de funcionalidades al prototipo de videojuego Mundo Isla desarrollado en el motor *Unity3D*. Estas se concentran en dos frentes:

- Permitir que el prototipo pueda contener otros videojuegos y acceder a ellos desde su estructura de niveles.
- Crear una infraestructura online para permitir a varios jugadores conectarse a Mundo Isla y jugar cooperativamente.

Esta última funcionalidad se completó utilizando el framework *Photon Unity Networking* para permitir la conexión y la interacción de los jugadores. Además, se utilizó una base de datos SQL para cargar datos sobre el estado de los jugadores y del mundo para permitir que el mundo sea persistente.

Uno de los prototipos a integrar utilizaba el dispositivo de Microsoft Kinect para permitir utilizar en la plataforma una interfaz de control natural. Por ello, comprobar la compatibilidad de Mundo Isla con este dispositivo es otro de los objetivos de este proyecto.

Palabras clave

Mundo Isla, Proyecto SAVEH, Unity, Unity3D, Photon Unity Networking, PUN, Kinect, Zigfu

Abstract

The aim of this project is the creation of some functionalities for Mundo Isla, a videogame prototype. It was developed using the *Unity3D* engine. Basically, there are two goals:

- The transformation of Mundo Isla into a platform which is able to contain other videogames as levels.
- The creation of a prototype of an online infrastructure to allow players to connect Mundo Isla and play cooperatively.

Photon Unity3D Networking framework was used to develop the last functionality. Additionally, an SQL database was used to load and save data involving the state of players and the world to make Mundo Isla a persistent world.

One of the integrated prototypes uses the Microsoft's Kinect device as a natural interface. Because of that, checking the compatibility of Mundo Isla with this device is another of the aims of this project.

Keywords

Mundo Isla, Proyecto SAVEH, Unity, Unity3D, Photon Unity Networking, PUN, Kinect, Zigfu

Índice general

Índice de Figuras

1. INTRODUCCIÓN

1.1 Antecedentes

Mundo Isla es un prototipo de videojuego multijugador online desarrollado con la herramienta *Unity3D* [1]. El escenario principal es una isla con dos zonas: una playa y un pueblo, pero se espera ampliar el número con las herramientas desarrolladas en este proyecto.

El prototipo está diseñado inicialmente para ser utilizado por niños en situación de hospitalización o en atención domiciliaria, de manera que puedan estar en contacto con otros niños en situaciones similares durante su convalecencia. A través de esta comunicación podrán interactuar para resolver cooperativamente las misiones que son planteadas por los personajes del videojuego.

Las misiones y las zonas se pretenden ampliar integrando videojuegos desarrollados independientemente de Mundo Isla en el sistema. Actualmente son dos: el primero utiliza el dispositivo *Kinect* como método de rehabilitación y el segundo se aprovecha de la interacción que otorga el sistema para utilizar el aprendizaje colaborativo en la solución de las misiones. En este proyecto se ha integrado únicamente el primer prototipo.

Gracias a la flexibilidad de esta integración, la jugabilidad de Mundo Isla se verá enriquecida y, además, se espera que amplíe los escenarios de uso del sistema en el futuro. Para llevar esto a cabo, se utilizan las mismas herramientas incluidas en el motor *Unity3D*. Se abordará con más detalle en el apartado 4.

Por último, también se requiere la creación de una plataforma online que permita discriminar a los jugadores en grupos de usuarios distintos para que Mundo Isla pueda ser usado en diferentes ámbitos (distintos hospitales, división por edades o por tipo de enfermedad, jugadores no hospitalizados, etc.). Para ello se utilizó *Photon Unity Networking* y *MySQL*, como se explica en el apartado 5.

1.2 Alcance

El proyecto se ha dividido en las siguientes tareas.

1. Integración mediante una interfaz tradicional.
2. Integración en Mundo Isla.
3. Compatibilidad de los dispositivos de interacción.
4. Despliegue de múltiples instancias de la plataforma.
5. Despliegue multiplataforma.

1.2.1 Integración mediante una interfaz tradicional

Esta tarea consiste en crear una interfaz tradicional para cargar los distintos prototipos en lugar de integrarlos como nuevas zonas de la isla. El objetivo era que el alumno se familiarizara con el entorno de trabajo *Unity3D*, pero no se descarta que se use en otras versiones de Mundo Isla.

1.2.2 Integración en Mundo Isla

En este apartado se busca añadir la funcionalidad de cargar nuevos niveles a través de la isla mediante el desarrollo normal del videojuego, como puede ser hablar con un personaje o llegar a un punto concreto del mapeado, por ejemplo.

1.2.3 Compatibilidad de los dispositivos de interacción

Se integrará soporte para *Kinect* de manera que los videojuegos diseñados para la utilización de este dispositivo puedan ser integrados en Mundo Isla sin contratiempos. *Kinect* permite al usuario comunicarse con un ordenador mediante una interfaz natural (gestos y voz).

1.2.4 Despliegue de múltiples instancias de la plataforma.

En este caso el objetivo es crear el soporte tecnológico necesario para que se puedan crear tantas instancias de la plataforma online de Mundo Isla como se requiera. Es decir, crear distintos servidores (o entidades equivalentes) para que diferentes grupos de jugadores puedan conectarse a la plataforma pero no puedan comunicarse con los componentes de otros grupos. Esto es útil para separar usuarios por edades u hospitales donde se encuentren.

1.2.5 Despliegue multiplataforma

Por último, esta tarea trata sobre la investigación de la compatibilidad de Kinect con las distintas plataformas en las que se puede compilar un proyecto de *Unity3D* [1]. Más concretamente, con el navegador web, que es la plataforma en la que se ejecutará finalmente Mundo Isla, y con la compilación en escritorio.

1.3 Próximos apartados

En el apartado 2 se explicarán nociones del programa *Unity3D* y del framework *Photon Unity Networking* y, en el 3, se describirá detalladamente el sistema de Mundo Isla. Seguidamente, en los apartados 4, 5 y 6 se detallan en profundidad el trabajo desarrollado a lo largo del proyecto. Finalmente, en el apartado 7 se incluyen las conclusiones y líneas futuras.

2. TECNOLOGÍAS DE SOPORTE

2.1 Unity3D

Unity3D [1] es un motor de videojuego multiplataforma disponible para Microsoft Windows y OS X. Las plataformas en las que se puede publicar contenido son Windows, OS X, Linux, Xbox 360, Playstation 3, Playstation Vita, Wii, Wii U, iPad, iPhone, Android y Windows Phone [2]. Gracias al plugin web de *Unity3D*, también se pueden crear videojuegos de navegador [3].

El contenido del juego es construido desde la interfaz del programa y el gameplay se programa usando scripts. Los lenguajes disponibles son *UnityScript* (una versión de JavaScript), C# y Boo [4]. En este proyecto se ha usado principalmente el primero.

Los juegos están estructurados en escenas. Una escena es un contenedor de elementos: modelados, scripts, audio, mapeado, etc. Se pueden utilizar como menús independientes, niveles o áreas de juego a elección del desarrollador.

2.1.1 Interfaz de Unity3D

En la ilustración 1 se muestra la interfaz del motor *Unity3D* durante el desarrollo de un proyecto. Esta se encuentra dividida en cinco partes que a continuación se explicarán en el orden en el que están numeradas en esa misma ilustración [5]:

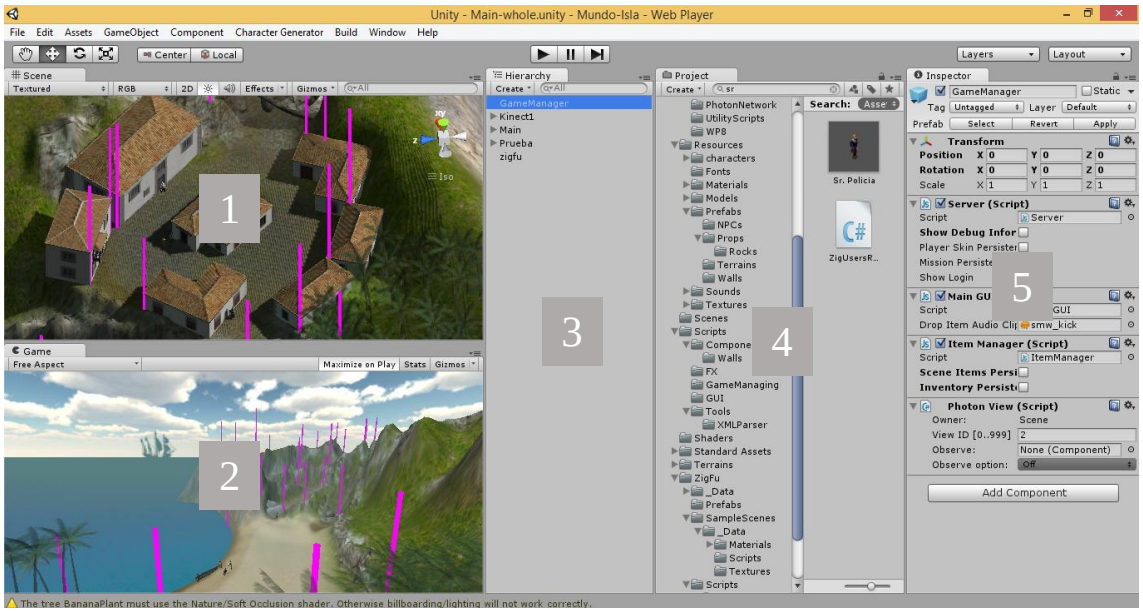

Ilustración 1 Áreas de la interfaz de Unity3D

1. Vista de escena: Es el área donde se construye visualmente la escena de un juego. Aquí se pueden añadir objetos, rotarlos, moverlos de lugar, etc. Todas las entidades que pueden ser colocadas en una escena heredan de la clase base *GameObject*.
2. Vista de juego: Aquí se previsualiza el juego. En cualquier momento puede reproducirse el juego pulsando el botón de play de la parte superior de la interfaz y jugarlo en esta vista.

3. Vista de jerarquía: Aquí se visualizan los *GameObject* que se encuentran en la escena actual y su jerarquía.
4. Vista de proyecto: En esta vista se organizan todos los recursos del proyecto (scripts, texturas, materiales, modelados, sonidos, etc.).
5. Vista de inspector: Sirve para ver y modificar los componentes de los *GameObject* seleccionados en la vista de escena, de jerarquía o de proyecto.

2.1.2 Otros aspectos relevantes de Unity3D

Explicar el funcionamiento de *Unity3D* no es el objetivo de esta memoria, pero puesto que hay ciertos elementos que pueden ser relevantes para el entendimiento de algunos de sus apartados, a continuación se definirán algunos entes del sistema *Unity3D* con más precisión.

GameObject

GameObject [6] es una clase de la que derivan todas las entidades de *Unity3D* que pueden colocarse en una escena. Están compuestos por componentes, los cuales pueden ser algún modelado tridimensional, un script, un archivo de audio, un emisor de partículas, etc. Los componentes de un *GameObject* se visualizan en la vista de inspector al ser este seleccionado, como se comenta en el apartado 2.1.1. Todos los *GameObject* deben tener un componente *Transform* que indica la posición en la que se encuentra ese objeto.

Por último, un *GameObject* puede contener una jerarquía de otros *GameObject* como hijos. Si es así, se presentará en la vista de jerarquía con un triángulo a la izquierda de su nombre. En la ilustración 1, *Kinect1*, *Main* y *Prueba* son ejemplos de ello.

Prefab

Un *prefab* [7] (prefabricado en inglés) es un tipo de recurso de *Unity3D* que permite almacenar un *GameObject* completo con todos sus componentes y propiedades. De esta forma se pueden crear multitud de copias de un mismo elemento fácilmente.

En Mundo Isla se utilizan *prefabs* para los ítems, los NPC y los jugadores.

Escena

Una escena [8] es un archivo independiente de *Unity3D* que contiene *GameObject*. Normalmente son fases independientes dentro de la lógica del juego, pero pueden ser zonas enteras o simplemente un menú de la interfaz.

2.2 Photon Unity Networking

Photon Unity Networking [10] (abreviado como PUN), es un plugin para *Unity3D* que permite configurar una conexión entre clientes remotos. Además, cuenta con un servicio denominado *Photon Cloud*, una serie de servidores de *Photon* a los cuales el usuario se puede conectar para que no tenga necesidad de crear un servidor por sí mismo.

La API tiene dos versiones:

- *PUN Free* (gratuita): Permite la conexión de hasta veinte usuarios concurrentemente a los servidores de *Photon Cloud*.

- *PUN+* (de pago): Permite la conexión de hasta cien usuarios a *Photon Cloud*. Además, añade la compatibilidad con las plataformas móviles iOS y Android.

En este proyecto se utilizó la primera versión. La compatibilidad con las plataformas móviles es, al menos de momento, irrelevante, y la limitación de veinte usuarios no fue un problema a lo largo de su ejecución, pero sí lo será en un futuro en la versión final. Sin embargo, en ese momento dejaría de utilizarse *Photon Cloud* y se crearía un servidor propio sin limitación de usuarios.

2.2.1 Salas

Una sala [11] de *Photon* es un ente que agrupa a un número determinado de usuarios y les permite comunicarse entre ellos fácilmente. Todos los usuarios que se encuentren en la misma sala también verán su escena sincronizada (movimiento del personaje o el estado de otros *GameObject*). Es útil para crear las distintas instancias de la plataforma Mundo Isla.

2.2.2 Alternativas a PUN

Antes de decantarse por PUN, se barajaron otras dos posibilidades: *SmartFoxServer* y *Unity Networking*.

SmartFoxServer

SmartFoxServer [12] es una plataforma que permite desarrollar videojuegos online en Adobe Flash, Unity3D, HTML5, iOS, Windows Phone, Android, Java y Windows 8, entre otros.

Tiene una documentación muy detallada y unas herramientas muy versátiles, pero no era la plataforma idónea para Mundo Isla, puesto que la cantidad de información que se envía a través de la red de *SmartFoxServer* está limitada. Esto quiere decir que su uso solo es viable en videojuegos simples o que no requieran de una alta interacción del usuario, como pueden ser juegos de estrategia por turnos o apps de móviles en 2D. La posibilidad de moverse en el entorno tridimensional de la isla provoca que la sincronización de los usuarios mediante la conexión de esta plataforma sea inviable.

Unity Networking

Unity Networking [13] es la infraestructua de red que *Unity3D* trae por defecto. PUN es una ampliación de esta infraestructura, por lo que está más limitada que *Photon*:

- No cuenta con servidores en los que probar el desarrollo (PUN cuenta con *Photon Cloud*).
- La compatibilidad con los dispositivos móviles no es completa. Sin embargo, PUN+ sí cuenta con ello.
- No tiene un concepto de sala o una entidad similar. Esto complica la creación de distintas instancias de la plataforma Mundo Isla.
- PUN en general tiene una API más amigable, puesto que gran parte de la sincronización se hace de forma transparente al desarrollador.

PUN finalmente se consideró la mejor opción debido a que cuenta con una API amigable y bien documentada, además de numerosos tutoriales de ejemplo. Además, gracias a las salas y a la

sincronización transparente, facilita enormemente la creación de múltiples instancias de la plataforma Mundo Isla. Por otra parte, también fue decisivo que contara con un ancho de banda óptimo para la sincronización del videojuego.

3. MUNDO ISLA

3.1 Introducción

Mundo Isla es un videojuego multijugador online parte del proyecto SAVEH [14]. Los usuarios que se conecten a esta plataforma deberán cooperar entre sí dentro del entorno de juego para poder resolver misiones en conjunto. Cada jugador tiene un avatar personalizable como su representación en la isla.

3.2 Descripción de las pantallas

Mundo Isla consta de varias pantallas que permiten al usuario interactuar con el sistema. En los siguientes apartados se presentarán con detalle.

3.2.1 Pantalla de login

En esta pantalla el usuario introduce sus datos de usuario y contraseña.

Ilustración 2 Login

En el caso de que se introduzca algún dato erróneo o el usuario con el que se quiere iniciar sesión ya esté conectado, aparecerá un mensaje de error alertando al usuario.

Ilustración 3 Login incorrecto o usuario ya conectado

3.2.2 Pantalla de bienvenida

Cuando el usuario introduzca sus datos de inicio correctamente, se mostrará la pantalla de bienvenida. En esta interfaz aparecerán dos opciones: Personalización y Jugar. A la derecha aparece el avatar del jugador.

Ilustración 4 Pantalla de bienvenida

3.2.3 Pantalla de personalización

Hacer clic en el botón Personalización de la ilustración 4 llevará a esta pantalla. En este caso, la interfaz muestra una serie de opciones de personalización del avatar del jugador:

- Permite elegir entre varios personajes aleatorios pulsando las flechas a la derecha de la opción “Personaje”.
- Se puede cambiar ciertas partes del cuerpo individualmente con los controles debajo de Personaje, en este orden: la forma de la cara, el color de los ojos, el color y la forma del pelo, la ropa de la parte superior del cuerpo, los pantalones y, finalmente, el calzado. Como se puede ver en la ilustración 5, a la derecha volverá a aparecer el avatar del personaje, esta vez con los cambios actualizados.
- Por último, en la parte inferior izquierda el jugador puede guardar los cambios hechos a su personaje o cancelar el proceso de personalización.

Ilustración 5 Pantalla de personalización

3.2.4 Interfaz durante el juego

Cuando el jugador pulsa Jugar en la pantalla de la ilustración 4, el usuario ya puede controlar a su personaje en la isla (ilustración 6). La cámara se sitúa detrás del personaje, que puede moverla pulsando y manteniendo el botón derecho del ratón. Para moverse, se pulsan las teclas WASD o las teclas de dirección del teclado.

En este momento, se muestran varias opciones en la interfaz en la parte inferior de la misma: Chat, Diario, Inventario y Menú. A continuación se explican las funciones de cada una de ellas:

- Chat: El chat consta de un espacio en blanco para escribir y un botón Enviar a su derecha. En la parte superior se muestra lo que han escrito él y otros personajes de Mundo Isla (ilustración 7).
- Diario: Muestra las misiones del personaje y su estado (ilustración 8).
- Inventario: Muestra los ítems que ha recogido el usuario (ilustración 9). Pulsando encima de cada icono, el usuario dejará caer ese ítem en la escena.
- Menú: Muestra de nuevo las opciones de la ilustración 4.

Ilustración 6 Interfaz durante el juego

Ilustración 7 Chat

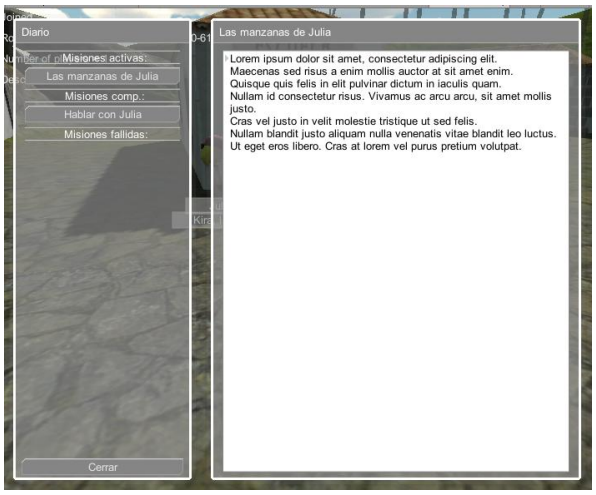

Ilustración 8 Misiones

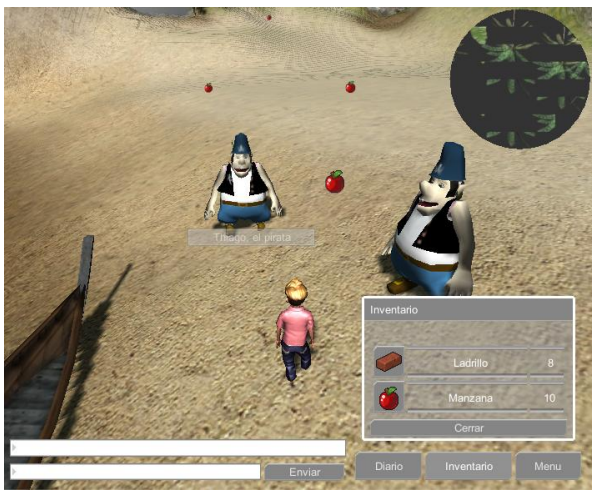

Ilustración 9 Inventario

3.3. NPC

En la isla hay una serie de NPC (Non-Player Character, Personaje No Jugador en español) con los que el usuario puede entablar una conversación. De esta forma se reciben información sobre

la isla o nuevas misiones con las que el jugador puede avanzar el desarrollo de su personaje. Los NPC tienen una etiqueta con su nombre justo debajo de su modelado.

Para hablar con un NPC, el jugador debe pulsar el botón izquierdo del ratón cuando esté a cierta distancia de ellos. En ese momento, se mostrará un diálogo en el que en ocasiones el jugador debe elegir varias respuestas para comunicarse (ilustración 10).

3.4 Ítems

Los ítems son una parte importante del sistema de juego de Mundo Isla. Se distinguen fácilmente en el mapa porque son pequeñas texturas bidimensionales con una etiqueta en su parte superior que indica su nombre.

El jugador obtiene los ítems pasando por encima de ellos o hablando con un NPC. Cuando alguna de estas dos cosas ocurre, ese ítem va a su inventario. Los jugadores también pueden dejar caer ítems (ver 3.2.4).

Ilustración 10 Diálogo

Ilustración 11 El jugador deja caer un ítem

4. INTEGRACIÓN DE OTROS PROTOTIPOS

4.1 Introducción

La creación de un método para integrar otros videojuegos dentro de Mundo Isla se requería para crear misiones dentro del sistema fácilmente. De esta manera, se enriquecería en gran medida la jugabilidad y la variedad de la plataforma, amén de poder ser utilizada en distintos ámbitos del inicial (niños en estado de hospitalización).

Para poder integrar otros videojuegos dentro de Mundo Isla era necesario crear un sistema que cargue nuevos niveles e instancie al jugador dentro de ellos. En *Unity3D* no es complicado, ya que el motor cuenta con el concepto de escena (apartado 2.1.2). Aún así, la implementación de PUN trajo consigo problemas a la hora de la carga de escenas, por lo que tuvieron que buscarse modelos alternativos.

En los apartados subsiguientes se detallarán las distintas soluciones a este problema que se probaron.

4.2 Modelo inicial de varias escenas en una sala

El primer modelo que se llevó a cabo para la integración de los prototipos es el común en *Unity3D*: las fases de un juego son equivalentes a escenas. Esta organización fue viable hasta que se comenzó a utilizar *Photon* para la plataforma online en el proyecto (ver apartado 5).

4.2.1 Explicación del problema encontrado con Photon

Primeramente, para la explicación se asume que hay dos jugadores en Mundo Isla. Inicialmente, ambos se autentifican en la plataforma y sus *prefab* de jugador se instancian en la escena inicial.

El problema ocurre cuando el primer jugador carga una nueva escena y el segundo jugador le sigue: a la hora de instanciar al segundo jugador en la máquina del primer jugador, el primero podía ver al segundo, mientras que el segundo no podía ver al primero. En la ilustración 12 y 13 se puede ver el problema.

Ilustración 12 Máquina del jugador uno y del jugador dos durante la instanciación inicial

Ilustración 13 Máquina del jugador 1 y del jugador 2 cuando ambos cambian de escena y ocurre el error

El error se identificó como un problema del sistema *Photon* en sí mismo. Este no cuenta con el conocimiento de la escena en la que se encuentra cada jugador, por lo que cuando un jugador cambiaba de escena y se instanciaba, *Photon* lo instancia en cada cliente, independientemente de en qué escena se encontrara este respecto al otro jugador. Luego esto provocaba errores de sincronización al cargar nuevas escenas debido al motor de *Unity3D*, que elimina todos los *GameObject* que se encuentran en la escena anterior. Por ello no aparecía el primer jugador en la segunda escena. Definitivamente, este modelo se mostró inviable.

4.3 Modelo de una sala por escena

En el apartado anterior se utilizaba una única sala en la que se agrupaban todas las escenas, mientras que en este se opta por crear una sala por cada escena. De esta forma, cada vez que un jugador cargue un nuevo nivel, deberá conectarse a una nueva sala. Esto provocaba ralentizaciones a la hora de cargar nuevas escenas, puesto que conllevaba una gran carga de red, pero eliminaba el problema comentado en el apartado anterior.

Sin embargo, el hecho de que los jugadores en distintas escenas no tengan la posibilidad de comunicarse impide la misma concepción de Mundo Isla, que pretende ser un juego multijugador online cooperativo.

4.4 Modelo de varios niveles dentro de una escena

Este modelo tiene su razón de ser en el hecho de que el problema de no visualizar al otro jugador es debido a la instanciación del personaje en la segunda escena. Para eliminar ese problema, se evita su causa: la instanciación misma. Para ello, en lugar de crear varias escenas, se crea únicamente una y dentro de ella se recrean todos los demás niveles.

Dentro de una escena de *Unity3D* se puede crear una jerarquía de *GameObject* para poder facilitar la lógica de la única escena que contiene los niveles. En la ilustración 14 se muestra un ejemplo de esa jerarquía.

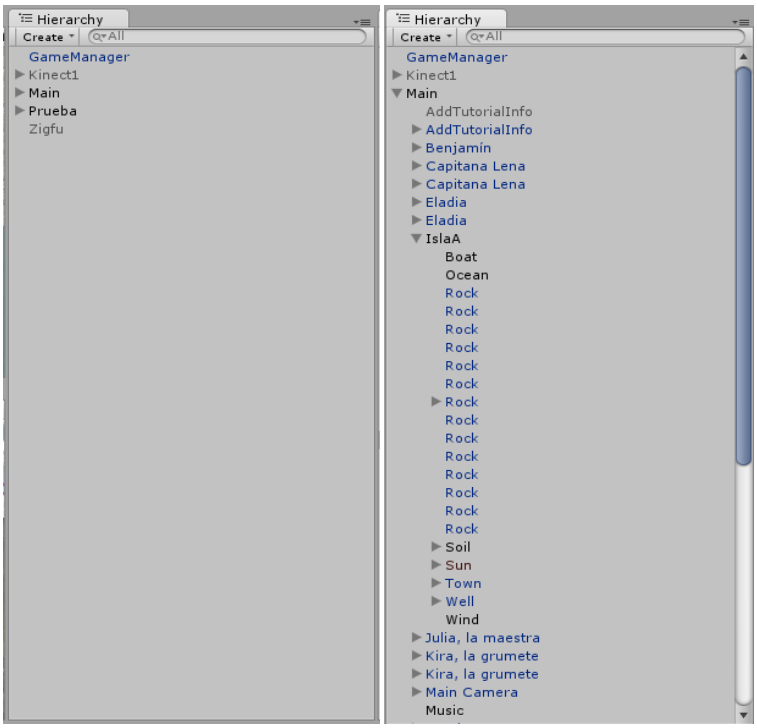

Ilustración 14 A la izquierda, jerarquía dentro de una escena Unity3D, a la derecha está Main desplegado

Cada nivel está separado de los demás mediante un *GameObject* vacío en el primer nivel de la jerarquía de la vista de escena. De este modo, se puede observar en la ilustración 14 que se tienen tres zonas separadas: *Kinect1*, *Main* y *Prueba*.

4.4.1 El modelo se desecha

El modelo era robusto en cierta medida, puesto que el tiempo de carga entre niveles era mínimo y no había errores debidos a la instanciación de *Photon*. Pese a lo anterior, presenta dos grandes carencias:

- Posibles errores en el futuro cuando Mundo Isla aumente de envergadura: Las escenas de *Unity3D* tienen un límite para el que la renderización de los *GameObject* falla debido a la precisión de las coordenadas en punto flotante. En el momento actual de la isla no entrañaba ningún problema, pero era posible que en un futuro este modelo se tornara inviable.
- Dificultad para integrar nuevos prototipos: La creación de nuevos niveles en este modelo no es algo intuitivo. Los nuevos desarrolladores que comenzaran a trabajar con Mundo Isla se encontrarían con una manera tosca de crearlos, puesto que todo se realizaría en una misma escena.

Debido a estos problemas, se optó por buscar un modelo con una lógica más simple.

4.5 Modelo de desconexión al cargar una nueva escena

Como se comentó en el apartado 4.2, instanciar a los dos jugadores en la escena inicial no presentaba ningún problema. En este caso, lo que se pretende es recrear el estado de Mundo Isla en ese momento. Para ello y cada vez que se cargue una nueva escena, el jugador se

desconectará de la sala y se conectará nuevamente. Esto resulta satisfactorio, ya que el comentado error que se quería subsanar no se repite y, además, se mantiene la simpleza lógica de las escenas y los niveles del apartado 4.2.

Pese a lo anterior, este modelo tiene un problema: el tiempo de carga entre escenas, debido a la reconexión, es sumamente alto. Una solución a esto, o al menos una molestia menor, sería crear una pantalla de carga entre las escenas.

Este modelo se terminó convirtiendo en el final. Aunque los tiempos de carga son muy altos, la simplificación del modelo para los nuevos desarrolladores y la mayor escalabilidad respecto al modelo de una sola escena lo convierten en el más adecuado.

4.6 Ejemplos de uso del cambio de nivel

Los cambios de nivel pueden implementarse en multitud de situaciones: completar una misión, hablar con un personaje, recoger cierto ítem, abrir una puerta, etc. Un ejemplo de un cambio de nivel es la historia que se describe a continuación.

Ilustración 15 El jugador se encuentra con el NPC

Ilustración 16 El jugador habla con el NPC

Como se puede ver en las imágenes, primeramente el usuario se encuentra con el NPC que va a activar el evento de cambio de nivel (ilustración 15) para acto seguido hablar con él. Le pregunta si desea ir a la playa, para lo que el usuario tiene dos opciones: aceptar la oferta o permanecer en ese nivel (ilustración 16). Si el jugador elige la primera opción, será transportado al nivel deseado (ilustración 17).

En este caso, el jugador permanece en la isla. Pero el cambio de niveles, como se ha comentado anteriormente, es también útil para cargar los otros prototipos de juegos que se quieren integrar en Mundo Isla.

Ilustración 17 El jugador finalmente viaja a la playa

En la siguiente historia, el jugador abre un videojuego que usa Kinect. El videojuego consiste en una serie de ejercicios físicos usando el dispositivo de Microsoft. En este caso, también se habla con un NPC (ilustración 18 e ilustración 19), el cual le pregunta si quiere realizar ejercicio. En el caso de que acepte, se abriría el nivel que aparece en la ilustración 20.

Ilustración 18 El jugador se encuentra con el NPC

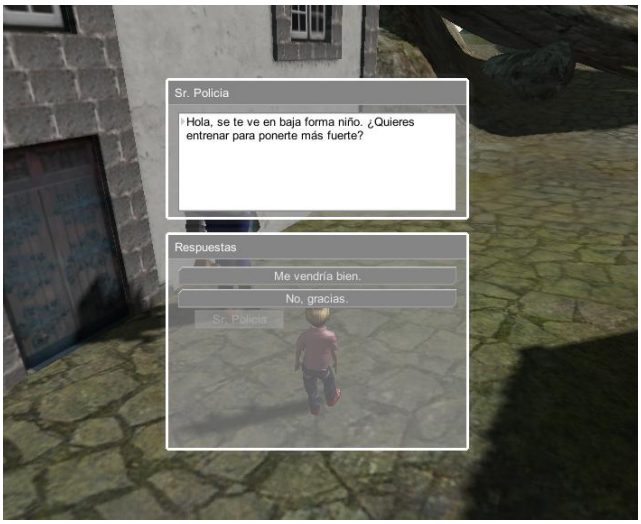

Ilustración 19 El NPC le pregunta si quiere hacer ejercicio

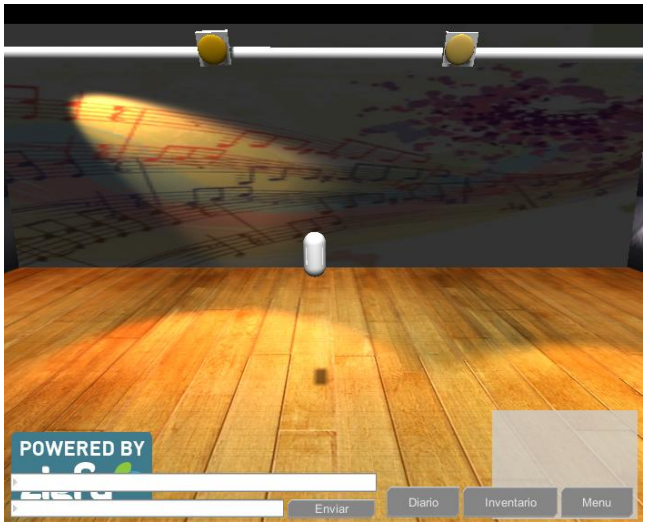

Ilustración 20 Se abre un prototipo que usa Kinect

5. CREACIÓN DE LA PLATAFORMA ONLINE

5.1 Introducción

Mundo Isla necesitaba una infraestructura online para crear distintas instancias de la plataforma y permitir la conexión de distintos grupos de usuarios. La herramienta elegida fue *Photon Unity Networking* (PUN), una plataforma para videojuegos multijugador especialmente construida para funcionar con *Unity3D*.

También era necesaria la persistencia del mundo. Por ello fue necesario crear una base de datos con la información relativa a las escenas y a los jugadores (posición de los ítems en el mundo, estado de las misiones de los jugadores y el inventario, etc.).

El proyecto se ha probado utilizando un servidor local para guardar la base de datos y los servidores de *Photon* en la nube para la conexión de los jugadores. Sin embargo, el código es fácilmente escalable para utilizar un servidor online propio en el momento en el que Mundo Isla salga de la fase de prototipado.

5.2 Base de datos

Un elemento importante de la plataforma online es la base de datos. Sin ella, Mundo Isla no sería persistente y cada vez que el usuario entrara a la isla, su inventario, misiones y avatar se verían reiniciados. También es necesaria para el registro de los usuarios.

En cuanto al diseño, se usó *Dia* [15] para el diagrama entidad-relación y *MySQL Workbench* [16] para el diagrama relacional. Para la infraestructura del servidor se usó *XAMPP* [17] con la propia máquina como servidor local.

En los apartados siguientes se explica la base de datos utilizando el diagrama entidad-relación y el diagrama en el modelo relacional, respectivamente.

5.2.1 Modelo entidad-relación

Jugador

La entidad *Jugador* representa a un usuario de Mundo Isla.

Cada usuario debe tener su nombre y contraseña para poder entrar a la plataforma. En *Avatar* se guarda la información gráfica de su personaje (que puede ser personalizado, como se explica en el apartado 3.2.3) y en *X*, *Y* y *Z*, unas coordenadas espaciales. Estas coordenadas se pueden utilizar para saber en qué posición estuvo el jugador la última vez que se conectó. Por último, *Última conexión* guarda la fecha en la que se conectó el usuario por última vez a la plataforma.

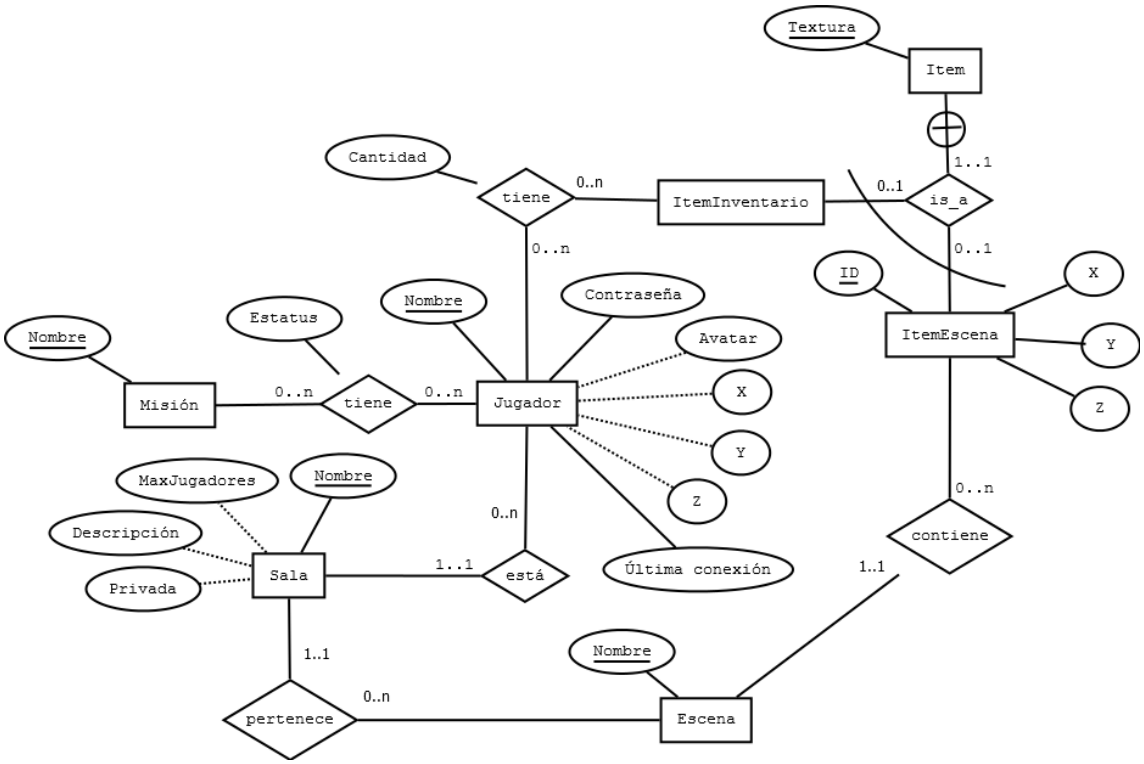

Ilustración 21 Base de datos en el modelo entidad-relación

Misión

Jugador está relacionado con *Misión* mediante *tiene*. Ambos lados de la relación tienen cardinalidad de cero a muchos, es decir: una misión puede ser tenida por ningún o varios jugadores y viceversa. También se guarda el *estatus* de la misión, el estado en el que se encuentra la misión para ese jugador. Puede ser: fallida, completada, en proceso u otros estados personalizados para cada misión que elija el desarrollador.

Sala

El jugador debe saber a qué sala de Mundo Isla conectarse, por lo que *Jugador* está relacionado con *Sala*. Esta relación implica que una sala puede no tener ningún o varios jugadores registrados en ella, mientras que un jugador sólo puede pertenecer a una sala.

Las salas se distinguen por un nombre. También pueden contener información sobre el número máximo de jugadores, una descripción y una indicación de si es privada o no.

Escena

En *Escena* se guarda la información sobre las esccenas de Mundo Isla. Esta entidad únicamente cuenta con un nombre.

Una *Escena* necesita conocer en que sala se implementa, ya que un mismo nivel en dos salas distintas puede haber sido alterado de formas diferentes; y por ello se relaciona con *Sala*. Esta relación refleja que una *Escena* solo puede tener una *Sala* asociada, mientras que una *Sala* tiene varias escenas.

Item

Un *Item* puede estar en el inventario o en un nivel. Dependiendo de si está en un lugar u otro requiere distinta información, por eso existe una entidad *Item* que se especializa en una entidad *ItemInventario* o en *ItemEscena*. La entidad padre solo tiene un atributo *Textura*, que es el nombre de la textura del objeto.

ItemInventario

Un *Jugador tiene* ningún o varios *ItemInventario*. *ItemInventario* representa una clase de ítem, como puede ser una manzana, un ladrillo o una espada. Por ello, en *tiene* también se encuentra la cantidad de ese ítem en el inventario del jugador.

ItemEscena

ItemEscena, sin embargo, representa un ítem concreto que está en el suelo de una escena. Por ello añade varios atributos más: un *ID*, que es un ID que PUN le asigna a los objetos, y las coordenadas X, Y y Z. Al igual que en *Jugador*, estas coordenadas representan una posición en una escena.

Finalmente, una *Escena contiene* ninguno o varios *ItemEscena*. Sin embargo, un *ItemEscena* solo puede pertenecer a una *Escena*, ya que es un ítem concreto.

5.2.2 Modelo relacional

En la ilustración 22 se ve la implementación final de la base de datos en el modelo relacional. En este caso se usan nombres en inglés para estar acorde a los scripts de Mundo Isla, cuyos nombres de clases, métodos y variables se encuentran en ese idioma.

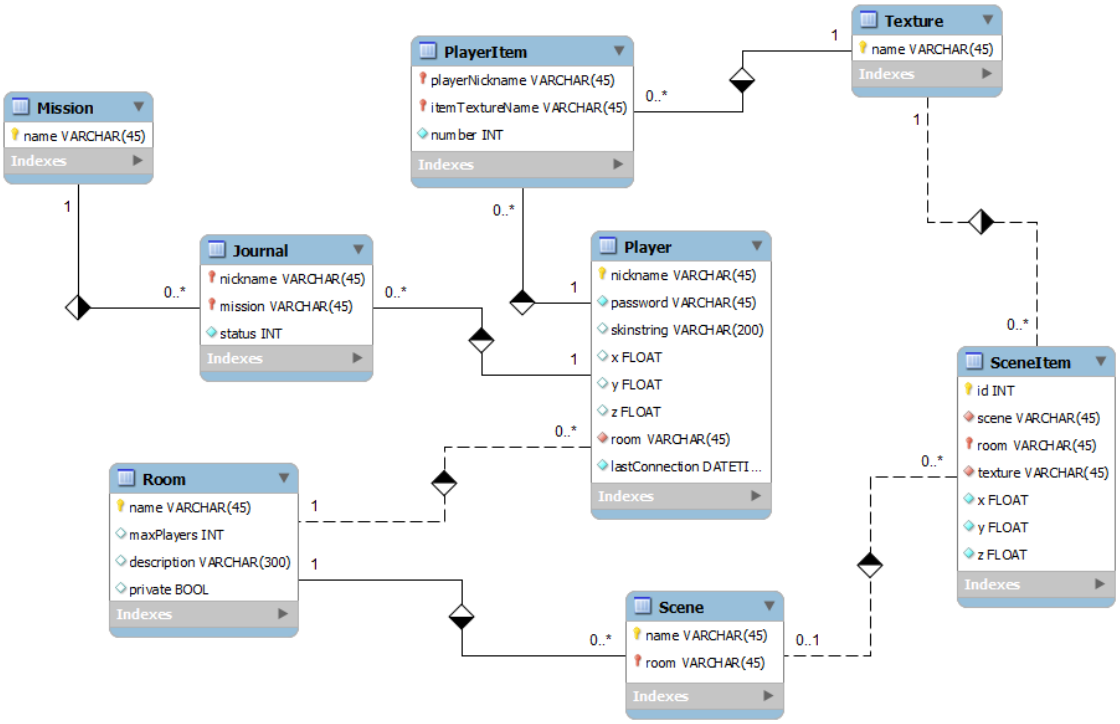

Ilustración 22 Base de datos en el modelo relacional

Player

Player almacena la información relativa a un jugador. *nickname* es su clave principal y a su vez su nombre de usuario. En *password* se guarda su contraseña y en *skinstring* la información de su avatar representada como una cadena de caracteres. Este último puede ser nulo para permitir al usuario elegir un avatar cuando inicie sesión por primera vez.

Los siguientes tres atributos, *x*, *y* y *z*, representan coordenadas espaciales en punto flotante de la posición del usuario en una escena. El siguiente atributo, *room*, es una clave ajena que referencia al atributo *name* de la tabla *Room*. Consiste en la sala en la que el usuario se conectará al iniciar sesión en Mundo Isla.

Por último, *lastConnection* guarda la fecha en la que se conectó el usuario por última vez. Esta se actualiza cada minuto en el proyecto *Unity3D*. La utilidad de este valor viene dado por el hecho de que dependiendo de su valor se puede asegurar o no que el usuario esté conectado en este momento a la plataforma y así permitir que pueda hacer login o rechazarlo.

Journal y Mission

Journal es una tabla que relaciona *Mission* con *Player*. Guarda información sobre una misión de un jugador y su estado. Sus dos claves principales, *nickname* y *mission* son, a su vez, claves ajenas que referencian a *nickname* de la tabla *Player* y a *name* de la tabla *Mission*, respectivamente. El último atributo de *Journal* es *status*, un entero con el estado que tiene esa misión para el jugador. Puede ser -1 (fallida), 0 (completada) o mayor que 0 (en proceso).

Room

En esta tabla se guardan los datos de una sala. La clave principal es el nombre de la sala, *name*. Los atributos sucesivos (todos con posibilidad de ser nulos) son *maxPlayers* (guarda el número máximo de jugadores que puede soportar esa sala), *description* (una descripción de la sala) y el booleano *private* (indica si la sala es visible o no).

Scene

La tabla *scene* (que representa un nivel dentro de Mundo Isla) contiene el nombre del nivel en *name* y el nombre de la sala asociada en *room*. Ambas son claves principales y *room* también una clave ajena que referencia a *name* de la tabla *Room*.

PlayerItem

PlayerItem es un ítem de jugador. Sus claves principales son *playerNickname* y *itemTextureName*, ambas a su vez claves ajenas. La primera referencia a *nickname* de la tabla *Player* y la segunda, a *name* de *Texture*. El tercer atributo es *number*, información sobre el número de ese tipo de ítems que tiene el jugador en su inventario.

SceneItem

La tabla *SceneItem* guarda la información de ítems en una escena. Sus claves principales son *id* y *room*. La segunda es una clave ajena que referencia a *room* de la tabla *Scene*. La primera clave representa el ID que otorga PUN cuando se instancia un objeto en la escena.

scene es una clave ajena que referencia al atributo *name* de *Scene* e indica el nivel en el que se encuentra el ítem. *texture* es otra clave ajena, y referencia a *name* de *Texture*; en este caso la textura que se usa para renderizar el ítem en la escena. Para terminar, los tres últimos atributos son las coordenadas tridimensionales que indican la posición del ítem.

5.3 Comunicación con el servidor de base de datos

Unity3D tiene herramientas para comunicarse directamente con un servidor de base de datos SQL, una mala opción que puede entrañar graves problemas de seguridad. A pesar de todo, en la versión *WebPlayer* de *Unity3D* no está permitido utilizar esas herramientas, puesto que hacen uso del sistema de archivos y desde el navegador no puede accederse a él como medida de seguridad.

En todo caso, lo mejor es hacer uso de scripts externos para acceder a la base de datos. En este proyecto se prefirió el uso del lenguaje PHP. En la ilustración 23 se puede ver la estructura de los directorios donde se encuentran esos scripts.

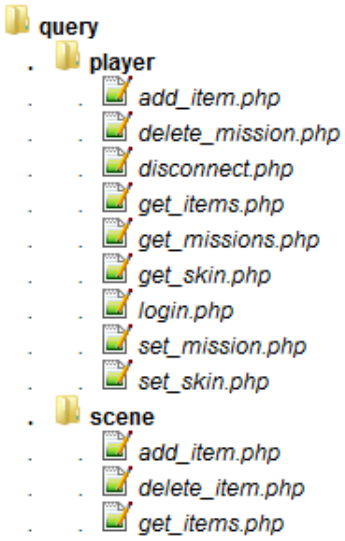

Ilustración 23 Estructura de scripts de consultas

5.4 Ficheros en el servidor

Los scripts no son los únicos ficheros que se encuentran en el servidor local, sino que hay multitud de archivos de diversa índole. A continuación se describe por orden el contenido de cada carpeta y fichero que se muestra en la ilustración 24:

- *Asset bundles*: Un *asset bundle* es un tipo de recurso de *Unity3D* que puedes ser enviado a través de una conexión web fácilmente. En esta carpeta se guardan ficheros de este tipo y, en este caso, están relacionados con la skin del personaje.
- Ficheros de configuración del servidor.
- Ficheros del prototipo de Kinect.
- Diálogos de los NPC.
- Ficheros de idioma.
- Scripts de PHP para consultar la base de datos.
- Logs del servidor.

- Varios ficheros de textos.
- Texturas de los ítems.
- *crossdomain.xml*, un fichero necesario para la conexión con el servidor.

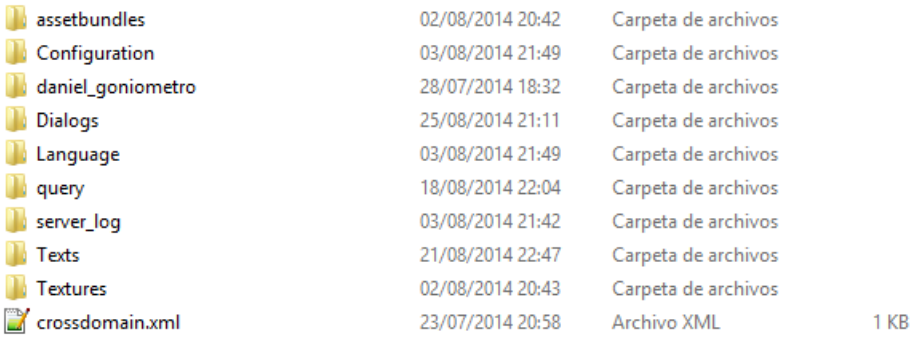

Ilustración 24 Ficheros en el servidor local

6. INTEGRACIÓN EN WEBPLAYER

6.1 Introducción

Otra de las tareas del proyecto era comprobar la compatibilidad del prototipo a integrar con la plataforma *WebPlayer*. El prototipo usaba *Zigfu* [18], un conjunto de librerías que permite utilizar Kinect en *Unity3D*. En un principio, las versiones compiladas para PC de *Unity3D* no tenían ningún tipo de problema, pero en *WebPlayer*, al estar limitado por el navegador podría presentar errores.

Los errores fueron básicamente dos: el primero relacionado con la compatibilidad de *Zigfu* con el *WebPlayer* y el segundo, con las limitaciones del navegador. Ambos problemas y su solución se explicarán detalladamente en los dos siguientes apartados.

6.2 Incompatibilidad de Zigfu con las opciones por defecto de compilación del WebPlayer

Cuando se compila para el *WebPlayer* de *Unity3D* y se ejecuta, el primer problema que ocurre es que al intentar abrir las librerías de Zigfu, el programa deja de responder. Es un error común de todos los proyectos de *Zigfu*. En el siguiente apartado se explica cómo solucionarlo.

6.2.1 Solución al problema

Cuando se compila para el *WebPlayer*, *Unity3D* crea dos archivos: un HTML y un archivo con extensión *unity3d*.

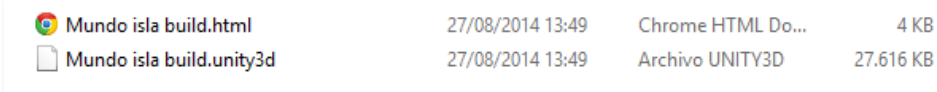

Ilustración 25 Resultado de la compilación para WebPlayer

El problema viene dado por el HTML que crea *Unity3D* por defecto. Para poder ejecutar el proyecto en sí (que es el archivo con extensión *unity3d*), el HTML descarga un JavaScript denominado *UnityObject2.js*. Sin embargo, para poder abrir un proyecto que use *Zigfu*, se necesita usar *UnityObject.js*.

```
8        <!--
9        var unityObjectUrl = "http://webplayer.unity3d.com/download_webplayer-3.x/3.0/uo/UnityObject2.js";
10       if (document.location.protocol == 'https:')
11           unityObjectUrl = unityObjectUrl.replace("http://", "https://ssl-");
```

Ilustración 26 Extracto del código del HTML por defecto

Esto último se puede comprobar al ver el código de alguno de los proyectos de ejemplo que está colgado en la página de *Zigfu*. Como se puede ver en la ilustración 27, los ejemplos usan una versión anterior de ese script.

```
117  </div>
118
119      <script type="text/javascript" src="./blockman3rdperson_files/UnityObject.js"></script>
120
121  <script type="text/javascript">
```

Ilustración 27 Extracto del código del HTML de un ejemplo de la página web de Zigfu

Por tanto, la solución era cambiar el script que usa el HTML para ejecutar el archivo *Unity3D*.

6.3 Imposibilidad de usar el sistema de archivos desde el WebPlayer

Algunos prototipos utilizaban inicialmente el sistema de ficheros debido a que no estaban pensados para ser usado en el *WebPlayer*, sino como aplicaciones de escritorio. Esto resultó otro problema, ya que es algo que no está permitido por temas de seguridad.

6.3.1 Solución al problema

Concretamente, uno de los prototipos accedía a un fichero para guardar los resultados de un ejercicio. Para poder realizar esta acción en el *WebPlayer*, se optó por el mismo enfoque usado a la hora de conectarse a la base de datos: se creó un script en PHP denominado *guardarResultado* que edita el fichero de resultados (llamado *resultados2*). De esta forma no se accede al sistema de ficheros, sino que desde *Unity3D* se llama al fichero PHP como si fuera una URL y es este el que lo edita. Esta práctica es un estándar para los proyectos en *WebPlayer*.

Finalmente, para la lectura de ficheros, simplemente ese fichero se mueve al servidor local. En el caso que nos ocupa, es un fichero denominado *ejer* con los datos de un ejercicio.

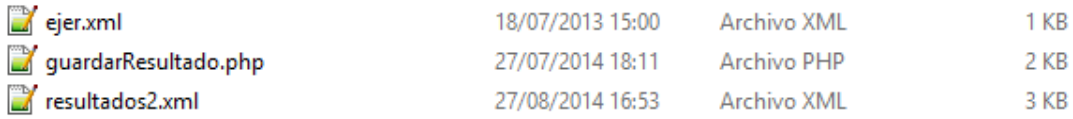

Ilustración 28 Ficheros XML y PHP del prototipo de Kinect en el servidor local

7. CONCLUSIÓN

El proyecto, como se ha comentado anteriormente, pasó por varias fases antes de ser terminado:

- La primera consistió en la integración de otros prototipos dentro de Mundo Isla. No estuvo exenta de problemas, puesto que tuvo que ser rehecha desde el principio en varias ocasiones debido a incompatibilidades con la siguiente tarea. Sin embargo, terminó cumpliendo su objetivo, al permitir la integración de otros videojuegos de una forma simple de cara al futuro desarrollador.
- Seguidamente se realizó la plataforma online. En este caso no hubo grandes problemas, únicamente ciertos cambios en el modelo de niveles que obligaron a transformar ligeramente la estructura de la base de datos.
- Por último, se comprobó la compatibilidad de Kinect con la plataforma *WebPlayer*. Esta resultó ser la fase más simple, puesto que sólo se encontraron dos errores de compatibilidad y solo uno de ellos estaba relacionado con el uso del mismo dispositivo.

7.1 Futuro a corto plazo de Mundo Isla

El futuro de Mundo Isla a corto plazo pasa por tener en cuenta ciertos aspectos o mejorar otros. Uno de ellos, relacionado con este proyecto, es el tema de la seguridad de la base de datos. Se ignoró en este desarrollo, pero es imprescindible la creación de cierta infraestructura para impedir el acceso no autorizado a la base de datos del servidor.

Otro aspecto a mejorar, ya comentado en apartados anteriores, es la creación de una pantalla de carga durante el cambio entre escenas. Es de vital importancia para que el usuario final tenga en todo momento conocimiento de lo que ocurre en el videojuego.

Ya fuera de lo desarrollado en este proyecto, es necesario pulir algunos aspectos de la jugabilidad. La cámara no funciona bien en algunos escenarios de la isla y es algo que puede frustrar al jugador, que siente que no tiene el control del personaje. Por último, la velocidad del jugador tampoco es la adecuada y debería aumentarse, debido a que el terreno de la isla es muy amplio. Otra solución a esto último sería reestructurar la isla en sí (algo quizás demasiado complicado) o, por otra parte, incluir formas de que el recorrido entre ciertos puntos de la isla sea más ameno o más rápido, bien enriqueciendo el entorno de la isla o bien creando modos de transporte alternativos, respectivamente.

7.2 Futuro a largo plazo de Mundo Isla

Inicialmente, Mundo Isla estaba pensado para que su público fueran niños en estado de hospitalización, pero ya en la introducción se habló de que este objetivo podría cambiar en un futuro. Esto no sería muy complicado, puesto que la integración de otros videojuegos puede cambiar la temática de la isla a gran escala. La viabilidad de este objetivo ya no recaería en el sistema de Mundo Isla en sí, sino en la habilidad de imaginar qué misiones podrían tener cabida dentro de la isla.

Siendo ambiciosos, si se quisiera llevar el videojuego a terrenos comerciales, habría que retocar aspectos visuales del sistema. Gráficamente necesita mejorar para ser atractivo a un mayor público, puesto que la estética de los personajes no se está acorde a los escenarios; siendo los

primeros poco realistas y el último, en exceso. Para arreglarlo, un simple cambio en las texturas del entorno sería la solución.

Por último, la banda sonora tampoco se encontraría a la altura de un videojuego comercial. Esto ya conllevaría una medida más complicada al necesitar una gran inversión para pagar profesionales de la música, o bien, como recurso más barato, se podría optar por el uso de música sin copyright.

7. CONCLUSION

As it has been explained before, this project has passed through different phases until it was finished:

- The first one was the prototype integration in Mundo Isla. It was a problematic task because it needed to be redone several times due to the incompatibilities with the next task. However, the aim was definitely accomplished, since the system allows the integration of prototypes in a simple way with a view to future developers.
- Next to that the online platform was created. In this case there was no important problems, only some changes related to the level model that forced some transformations in the database's structure.
- Finally, the compatibility of the *WebPlayer* platform with Kinect was checked. This was the simpler phase, since only two compatibility errors were found and only one of them was related to the device.

7.1 Mundo Isla's short-term future

The short-term future of Mundo Isla will involve the improvement of some aspects. One of them that is directly related to this project is the database security. In this development, it was ignored, but the creation of an infrastructure to block unauthorized access to the server is totally essential.

Another point to improve is the creation of a load screen to show during the scene loading. Is really important to assure that the final user would have knowledge of whatever is happening in the videogame.

It is also necessary to improve gameplay. The camera does not works well in some parts of the scenery of the island and it would be a frustration to the final player because he or she will lose the character control. To conclude, the speed of the character should be increased because the area of the island is too big. Another solution to this could be the restructuration of the island (maybe too complex to achieve). It would be also possible to heighten the island scene or creating some vehicles to speed up travels.

7.2 Mundo Isla's long-term future

Initially, Mundo Isla was thought to have hospitalized children as public, but this aim could change in the future. This would not be complicated, since the videogame integration can change the island theme in different ways.

As conclusion, if the videogame would be commercialised, some graphic details need to be improved. The aesthetic of the characters is not consistent with the scene: the first ones are not realistic, but the second is. A simple change in textured would be fine. Finally, the music also needs improvement. This would be complicated if a professional musician is paid to do the music, but the use of copyright free music would be fine too.

8. BIBLIOGRAFÍA

[1] Unity Technologies. “Unity - Game Engine”. unity3d.com unity3d.com/es (04/09/14)

[2] Unity Technologies. “Unity - Multiplatform - Publish your game to over 10 platforms”. unity3d.com unity3d.com/unity/multiplatform (04/09/14)

[3] Unity Technologies. “Unity - Web Player game development”. unity3d.com unity3d.com/unity/multiplatform/web (04/09/14)

[4] Unity Technologies. “Unity - Scripting - Script your gameplay in a world-leading programming environment". unity3d.com unity3d.com/es/unity/workflow/scripting (04/09/14)

[5] Collado, David. “Empezando en Unity3D”. trinit.es trinit.es/unity/tutoriales/manuales/2%20-%20Introducci%F3n.pdf (04/09/14)

[6] Unity Technologies. “Unity - Scripting API: GameObject”. docs.unity3d.com docs.unity3d.com/ScriptReference/GameObject.html (04/09/14)

[7] Unity Technologies. “Unity - Manual: Prefabs”. docs.unity3d.com docs.unity3d.com/Manual/Prefabs.html (04/09/14)

[8] Unity Technologies. “Unity - Manual: Creating Scenes”. docs.unity3d.com docs.unity3d.com/Manual/CreatingScenes.html (04/09/14)

[9] Unity Technologies. “Unity - Manual: Coroutines”. docs.unity3d.com docs.unity3d.com/Manual/Coroutines.html (04/09/14)

[10] Exit Games. “Photon Unity Networking Intro”. doc.exitgames.com doc.exitgames.com/en/pun/current/getting-started/pun-intro (04/09/14)

[11] Exit Games. “Photon Unity Networking: Room Class Reference”. doc-api.exitgames.com doc-api.exitgames.com/en/realtime/current/pun/doc/class_room.html (04/09/14)

[12] gotoAndPlay() “SmartFoxServer: massive multiplayer game server for Flash, Unity 3D, HTML5, iPhone/iPad and Android games, MMO, virtual worlds and communities”. smartfoxserver.com www.smartfoxserver.com (04/09/14)

[13] Unity Technologies. “Unity - Manual: Network Reference Guide” docs.unity3d.com docs.unity3d.com/Manual/NetworkReferenceGuide.html (04/09/14)

[14] SAVEH. “Servicio de Apoyo Virtual Educativo Hospitalario” saveh.es www.saveh.es/wordpress/ (04/09/14)

[15] Dia. “Dia draws your structured diagrams: Free Windows, Mac OS X and Linux version of the popular open source program”. dia-installer.de dia-installer.de (04/09/14)

[16] ORACLE. “MySQL :: MySQL Workbench”. mysql.com www.mysql.com/products/workbench (04/09/14)

[17] Apache Friends. “XAMPP Installers and Downloads for Apache Friends”. apachefriends.org www.apachefriends.org/es/index.html (04/09/14)

[18] Motion Arcade. “Zigfu - Kinect Development in HTML, Unity3D and Flash”. zigfu.com zigfu.com/ (04/09/14)

ANEXOS

Anexo 1: Manual para la creación de una instancia de Mundo Isla en el servidor local

MANUAL PARA LA CREACIÓN DE UNA INSTANCIA DE LA PLATAFORMA ONLINE EN UN SERVIDOR LOCAL

En este manual se explicará cómo crear una instancia de la plataforma online de Mundo Isla en un servidor local para poder ejecutar el videojuego.

1. Descarga e instalación de XAMPP y primera ejecución

Para la prueba de la plataforma online de Mundo Isla se utilizó XAMPP, un servidor *Apache* que contiene *MySQL*, PHP y *Perl* ya integrados, entre otras funcionalidades.

Para descargar el software, hay que entrar a su página web (https://www.apachefriends.org/es) y pulsar sobre el botón verde de *Descargar*.

Imagen 1

Una vez abierto el asistente de instalación (imagen 2) y tras pulsar *next* se mostrará una pantalla que pregunta qué componentes de XAMPP se quieren instalar (imagen 3). En el caso de Mundo Isla, basta con *Apache*, *MySQL* y PHP.

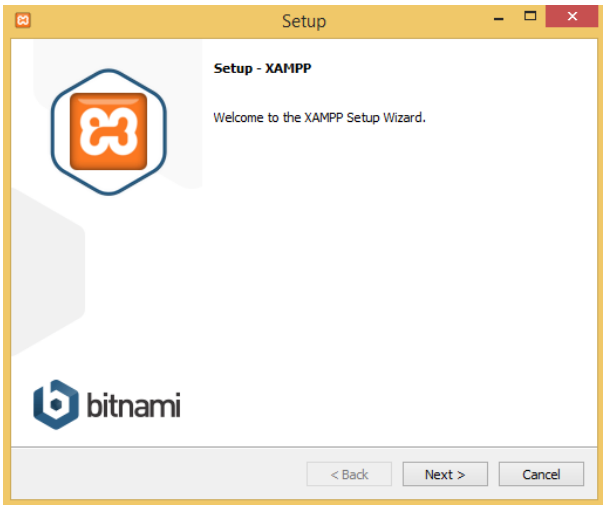

Imagen 2

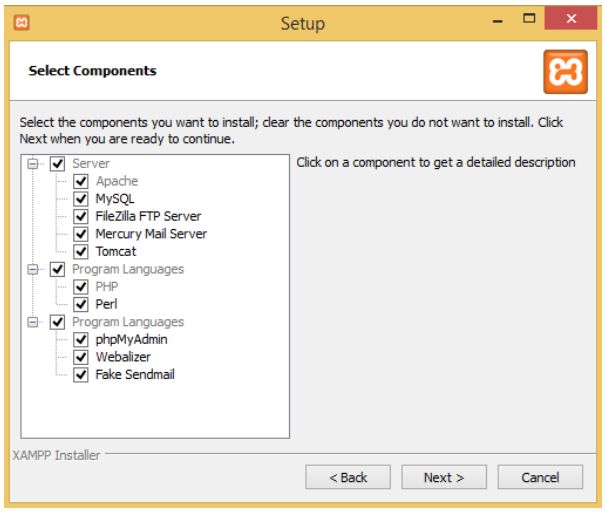

Imagen 3

Tras pulsar *next* un par de ocasiones más, la instalación debería terminar con éxito.

Tras ello se dispondrá a ejecutar el programa (imagen 4). En este momento se inicará el servidor *Apache* y *MySQL* pulsando en sus respectivos botones Start.

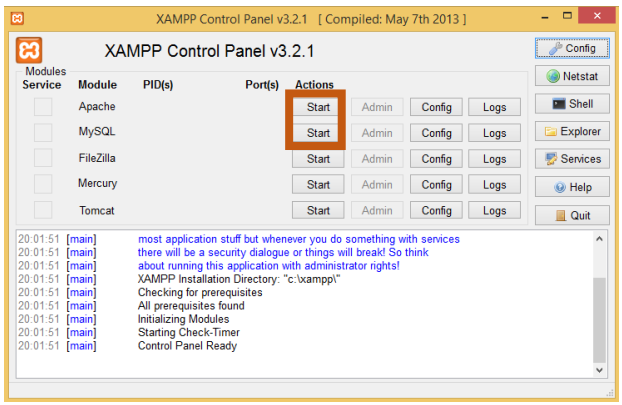

Imagen 4

2. Importación de la base de datos

Para poder importar la base de datos en XAMPP, deberá haber recibido los archivos que aparecen en la imagen 5.

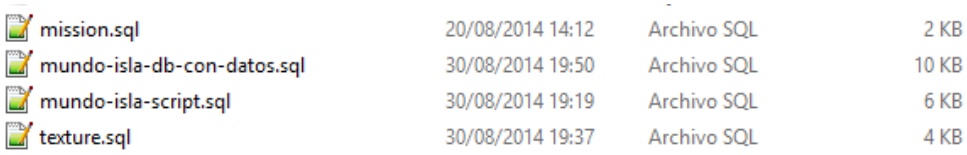

Imagen 5

Para obtener la estructura de la base de datos de Mundo Isla sin ningún dato introducido, desde *PHPMyAdmin* hay que importar el script llamado *mundo-isla-script.sql*. Para ello, desde el navegador se escribe *localhost/phpmyadmin* para entrar en el entorno de administración de la base de datos (imagen 6). A continuación se procederá a importar el archivo pulsando la pestaña

Importar y luego *Seleccionar archivo* para buscar el fichero en el equipo. Finalmente, se pulsará *Continuar*.

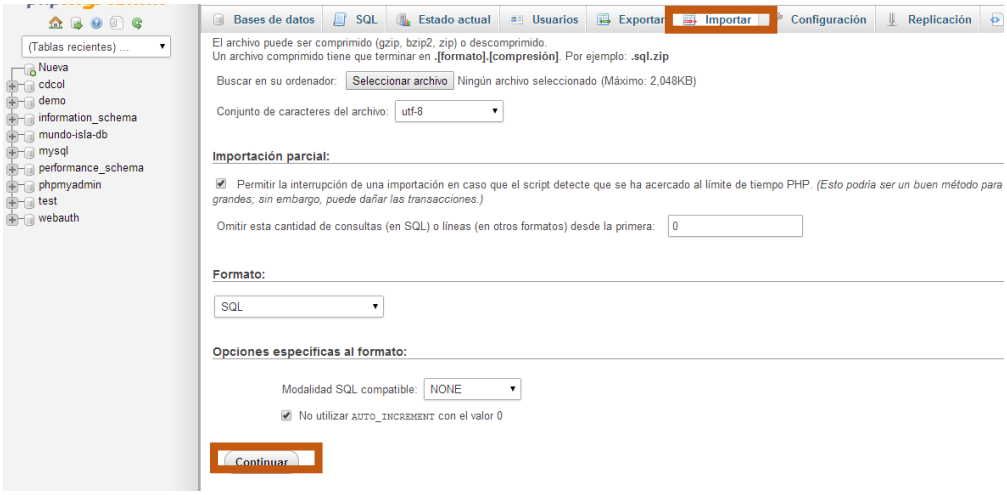

Imagen 6

En lugar de lo anterior, se puede importar directamente la base de datos con todos sus datos en el fichero *mundo-isla-db-con-datos.sql*. En este caso contiene toda la información de las misiones, texturas y escenas además de una sala y dos jugadores de prueba.

Si se importó la base de datos vacía, lo siguiente que se debe realizar es seleccionar *mundo-isla-db* a la izquierda y pulsar la pestaña *Importar*. De esta manera se pueden importar tablas a la base de datos de Mundo Isla desde un script externo. Los ficheros que se importarán son: *mission.sql* y *texture.sql*. Estos contienen los nombres de las misiones y las texturas que se encuentran en el servidor.

3 Creación de una sala

Antes de poder ejecutar Mundo Isla, se necesita añadir a la base de datos la información de al menos una sala y los niveles que se vayan a utilizar (de momento son *Main*, *Prueba* y *Kinect1*).

Para poder añadir la información de la sala, se debe seleccionar la base de datos de mundo isla a la izquierda y luego *room* en el submenú que aparece justo debajo. Pulsando a continuación la pestaña *insertar*, se mostrará la ventana que aparece en la imagen 7.

En esa ventana se añade toda la información necesaria para crear una escena:

- *name*: nombre de la escena.
- *maxPlayers*: número máximo de jugadores que se pueden conectar a esa sala.
- *description*: una breve descripción de la sala.
- *private*: un 0 si la sala es visible a los jugadores y un 1 si está oculta.

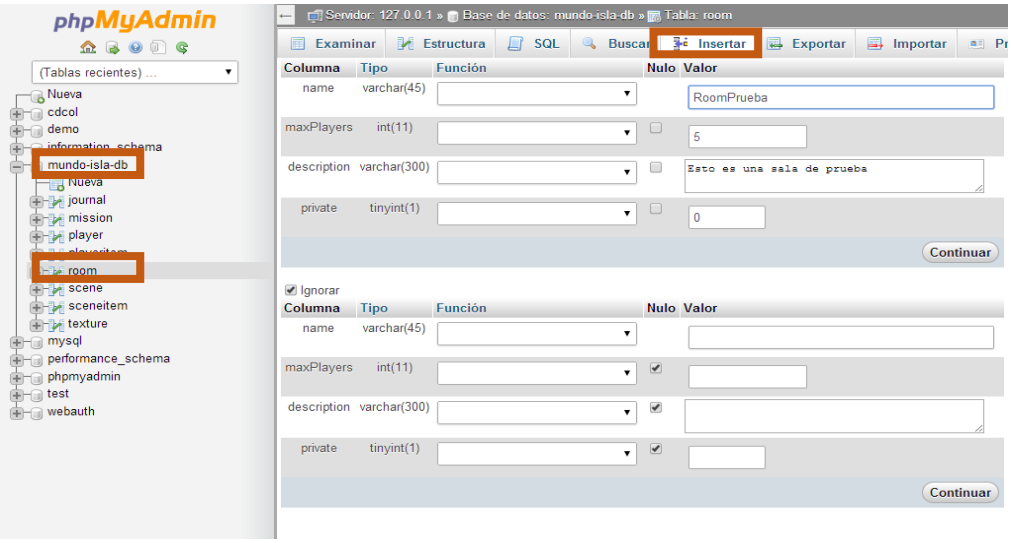

Imagen 7

4 Creación de los niveles

Para crear la información de los niveles, se debe llevar a cabo el mismo procedimiento que con las salas pero con la tabla *scene*. En este caso, la información a rellenar es únicamente el nombre de la escena y la sala asociada (imagen 8). Para poder ejecutar Mundo Isla en el estado actual, se debería añadir información para las salas *Main*, *Prueba* y *Kinect1*.

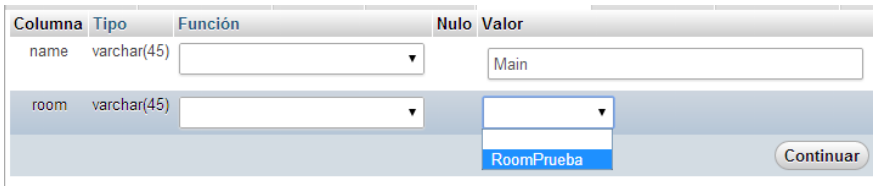

Imagen 8

5 Creación de los usuarios

Sólo resta dar de alta a los usuarios en la base de datos. Para ello, se deben insertar los siguientes datos en la tabla *player*:

- *nickname*: Nombre del jugador.
- *password*: Contraseña del jugador.
- *skinstring*: Avatar del jugador. Es preferible dejarlo en blanco y que luego el jugador elija su avatar en las opciones de personalización.
- *x*, *y* y *z*: Posición del jugador. De momento no se utiliza en Mundo Isla, pero es posible que se requiera en versiones siguientes. Dejar en blanco.
- *room*: Sala a la que este jugador está asociado.
- *lastConnection*: Última conexión del jugador. Se puede poner cualquier valor que sea igual o anterior al instante en el que se da de alta al jugador.

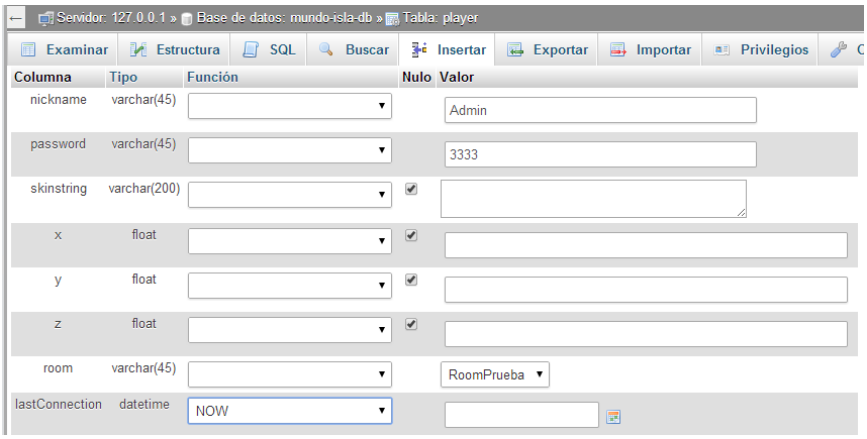

Imagen 9

Tras el paso anterior, ya casi se puede utilizar Mundo Isla con total normalidad. Opcionalmente se pueden añadir ítems a algún nivel rellenando la tabla *sceneitem*, puesto que en este estado los niveles no contendrán ninguno. Sin embargo, es preferible que sean añadidos por algún jugador que ejecute el juego y los suelte desde su inventario, debido a que desde el administrador de la base de datos no es posible saber en qué posición será colocado cada ítem.

6. Conexión con Photon

Antes de poder ejecutar Mundo Isla, se debe crear una cuenta en *Photon*. Para ello, hay que entrar a su página web (https://www.exitgames.com/en/PUN) y pulsar sobre *Sign Up*.

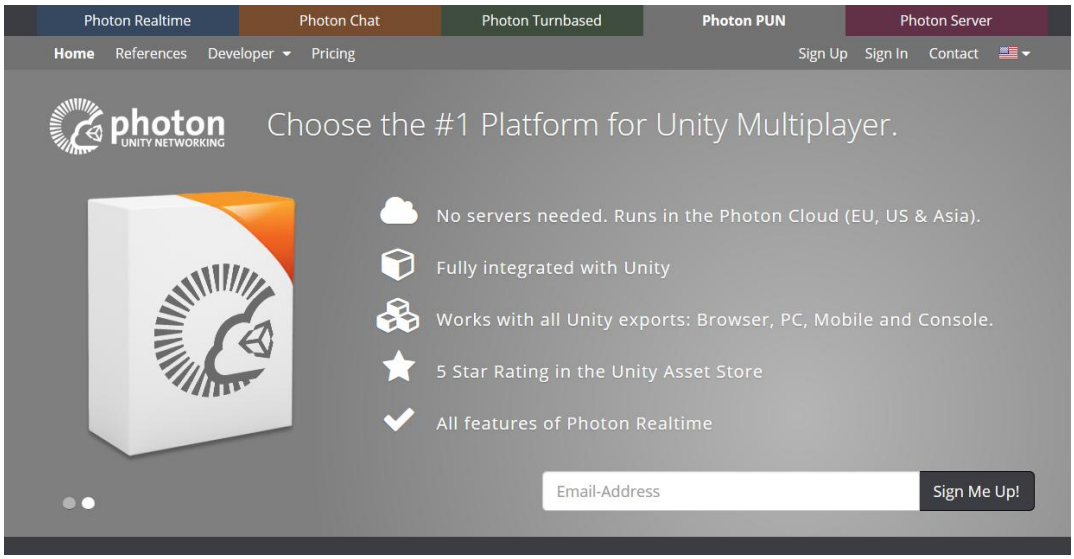

Imagen 10

Tras lo anterior, aparecerá la pantalla de la imagen 11. En el recuadro señalado deberá introducirse una dirección de email y pulsar el botón amarillo a la derecha con la palabra *Sign Up* en su interior. De esta forma se enviará un correo de confirmación a esa dirección.

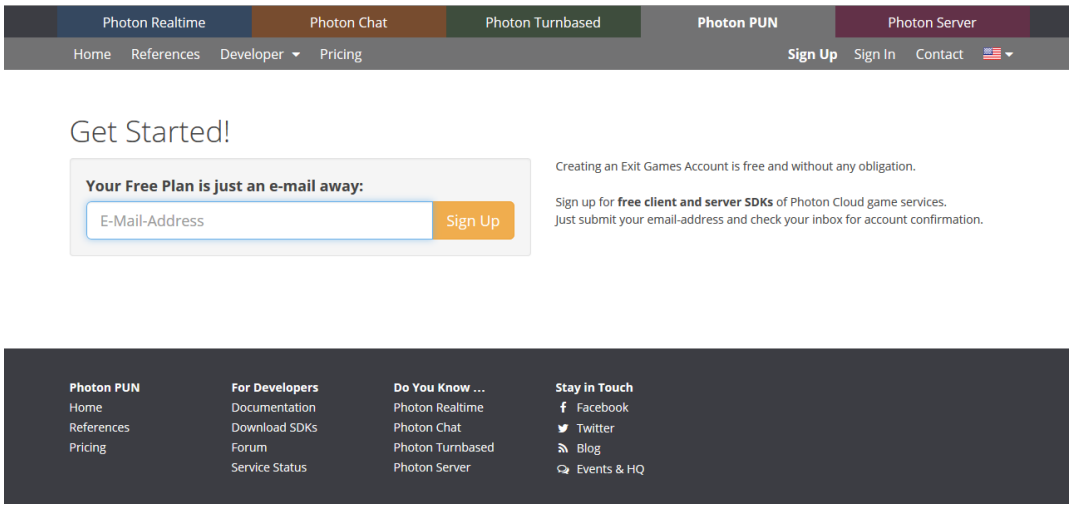

Imagen 11

En el correo de confirmación habrá un link que abrirá la pantalla de la imagen 12. Es el momento de elegir la contraseña de la cuenta.

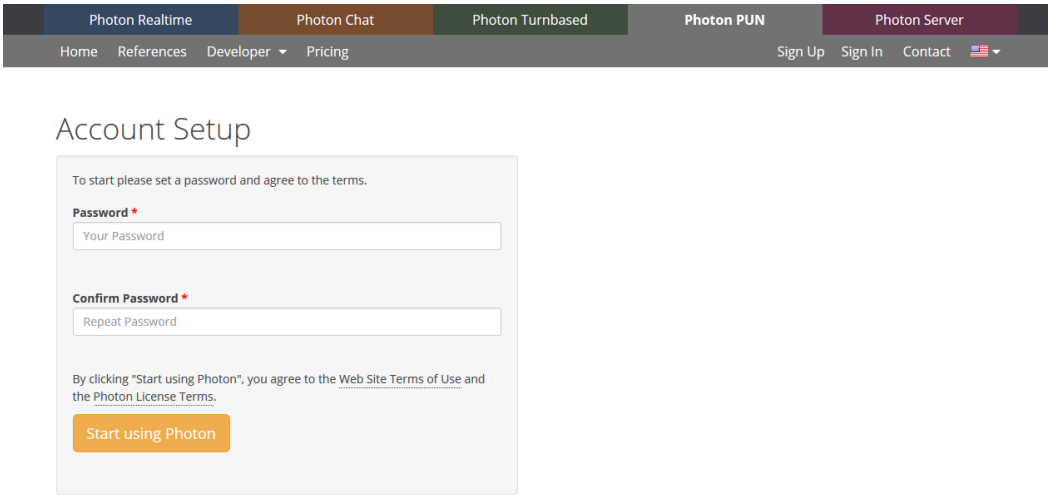

Imagen 12

Finalmente, se activará la cuenta (imagen 13).

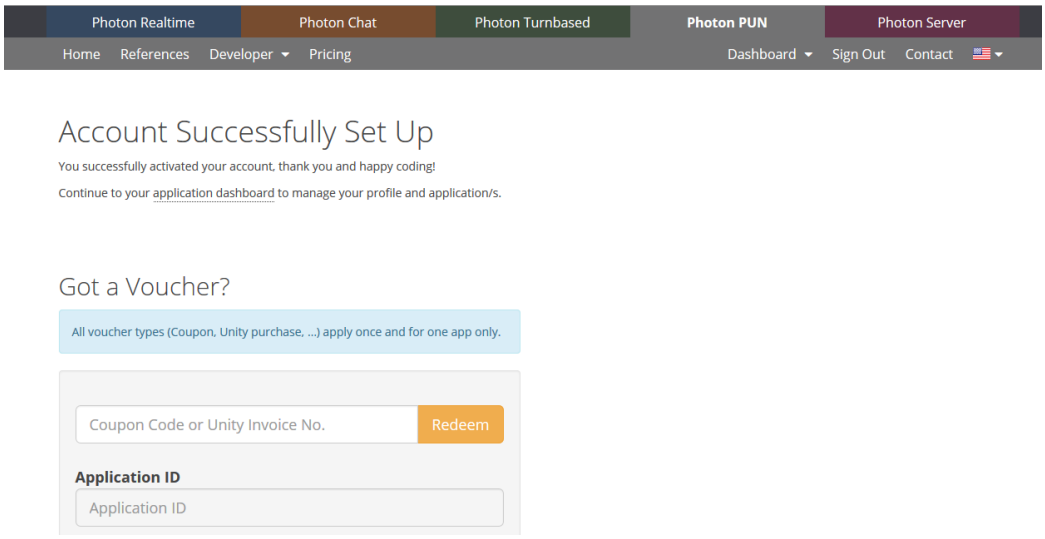

Imagen 13

Para poder utilizar *Photon Cloud*, es necesario obtener el *Application id* de una aplicación de la cuenta de *Photon*. Para ello, se requiere entrar en el submenú *My Applications* del menú *Dashboard* (imagen 14). El código alfanumérico debajo de *Application id* es lo que se necesita.

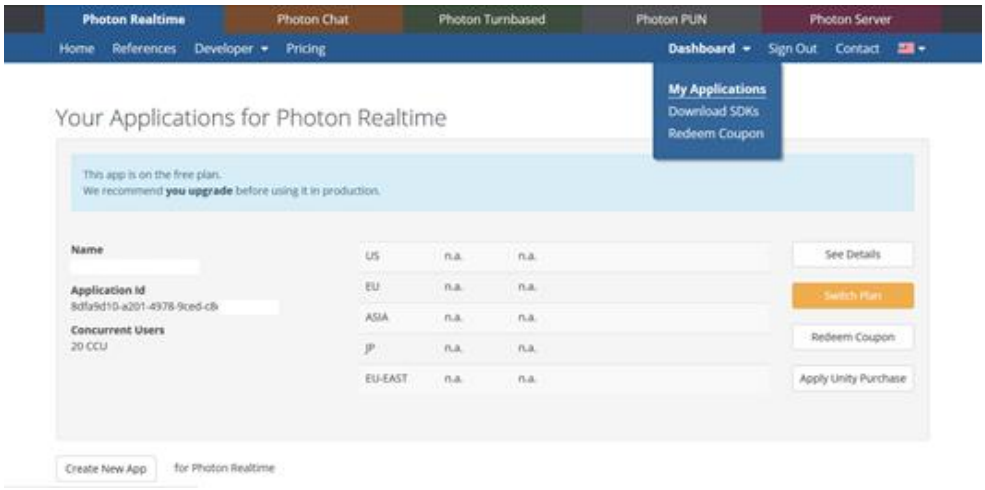

Imagen 14

A continuación se abre el proyecto en *Unity3D*. Cuando se ejecute por primera vez aparecerá la pantalla de la imagen 15. Si no es así, hay que pulsar en la opción *Window/Photon Unity Networking/PUN Wizard* de la barra de herramientas del programa.

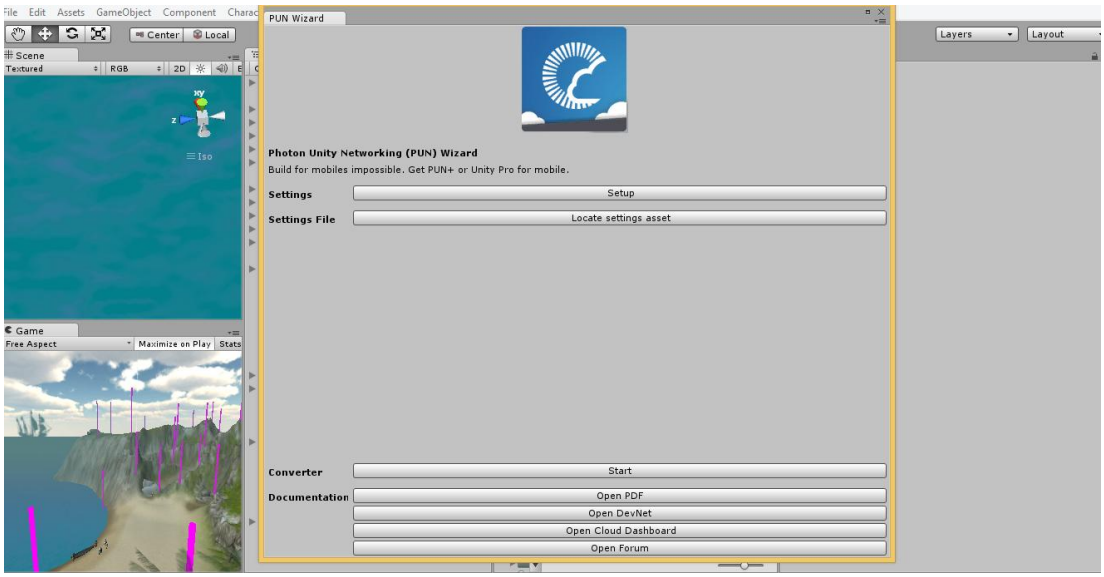

Imagen 15

El siguiente paso es pulsar el botón *Setup* de la imagen 15. Esto abrirá la ventana de la imagen 16. Y ya, finalmente, al introducir el *Application id* en el recuadro debajo de *Your AppId* y pulsando *Save*, ya es posible ejecutar Mundo Isla sin problemas.

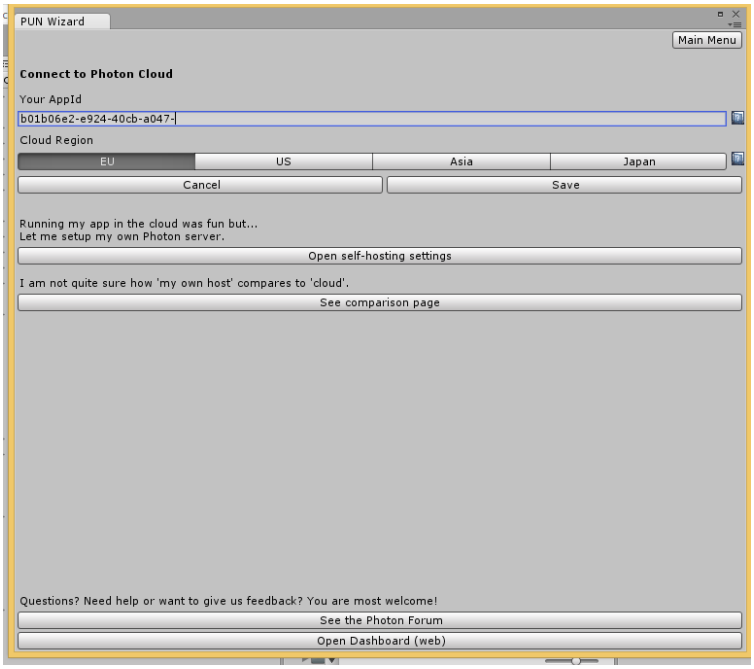

Imagen 16