Beca de Colaboración

Phiby's Adventures v2

Grupo de Aplicaciones Multimedia y Acústica

Henar Redondo Martin
2020/2021

Índice

1. Introducción

Este es un resumen de mi trabajo realizado en la beca de colaboración en el curso 2020/2021. La beca ha sido desarrollada en el grupo GAMMA (Grupo Aplicaciones MultiMedia y Acústica) de la UPM (Universidad Politécnica de Madrid). Se ha trabajado desarrollando y mejorando el videojuego Phiby's Adventures 3D v2, juego pensado para ser jugado con una Kinect para mejorar la movilidad de niños y personas con discapacidad física.

El trabajo se ha desarrollado en dos semestres y se ha colaborado con diferentes alumnos. Durante el primer semestre se mejoró la jugabilidad y rendimiento del proyecto, mientras que el trabajo realizado en el segundo semestre esta más enfocado a continuar el desarrollo del juego.

El juego está dividido en una escena principal que es un mundo abierto y diversas escenas de ejercicios. Se partió de una escena principal y un único ejercicio que consistía en recoger unas manzanas. Al finalizar mi trabajo se había mejorado y ampliado el terreno de la escena principal, se había creado otro ejercicio y se estaban desarrollando los siguientes.

2. Objetivos

Los objetivos de la beca se han ido actualizando a la vez que se iba desarrollando el proyecto y se iban teniendo necesidades diferentes. En resumen, los objetivos principales de la beca se pueden resumir en los puntos que vienen a continuación, sin embargo el trabajo realizado ha sido más extenso.

- Reorganización del proyecto, creando un nuevo proyecto.
- Mejora del rendimiento general
- Implementación de escenas de ejercicio y funcionamiento de los movimientos
- Control de los parámetros de los ejercicios

El trabajo realizado a partir de estos objetivos puede consultarse en el GDD [1]

3. Primer Semestre

El trabajo del primer semestre (septiembre-enero) se realizó con la ayuda de los compañeros Laura Caballero Ruiz y Dario Cavero Cabrera. El trabajo en este periodo se ha realizado parte de manera individual y parte de manera conjunta.

Junto con estos compañero, se realizó el semestre anterior un videojuego para la asignatura SAI (Síntesis de Animación e Imagen), y se continuó con la dinámica de trabajo en equipo establecida para ese proyecto. Aunque cada integrante del equipo tenía asignadas unas tareas ha existido una constante colaboración y ayuda entre unos y otros.

Los objetivos fijados en el primer semestre consistían en unificar el juego, corregir errores y mejorar el rendimiento del mismo. Una vez cumplidos esos objetivos, se tenía pensado continuar implementando el resto de los ejercicios, pero al final esta tarea quedó postpuesta al segundo semestre.

3.1. Mejora del rendimiento: Reestructuración

Uno de los principales problemas que existían a la hora de jugar, era que el juego iba muy ralentizado. Se estuvieron analizando las causas de esto, y como primer motivo se descubrió que cuando los árboles de las manzanas entraban en pantalla, los FPS del juego se reducían sustancialmente haciendo muy complicado avanzar por la escena.

Para intentar corregir esto, se modificaron los árboles con el editor de Unity reduciendo el numero de hojas que los componían. Para que esta reducción fuese lo menos notable posible, se amplió el tamaño de las mismas. Con este cambio se mejoró el rendimiento del juego aumentando los FPS y haciendo un poco más fácil jugar.

El siguiente elemento que se modificó para la mejora del rendimiento, fue realizar una restructuración total del proyecto. Esto se realizó a partir de un proyecto en blanco al que se le fueron añadiendo todos los objetos existentes y utilizados hasta entonces. Se creó una estructura base para tener el proyecto organizado instaurando una serie de normas para crear nuevas escenas.

Para organizar la estructura del juego, la escena principal, así como el resto de las escenas, se dividen en objetos atendiendo a las siguientes características:

- Hay un **objeto padre por cada zona** del juego, actualmente existen la celda, el bosque, la montaña y el rio.
- Dentro de estos objetos, la jerarquía se divide en objetos **interactivos** y **estáticos**. Dentro de los cuales están los objetos que tienen un comportamiento definido por script/especial, y los que no lo tienen.
- Todos los objetos, han de estar con una **posición 0,0,0 y una rotación de 0,0,0**. El único objeto que debe tener la posición es el último de cada cadena. De este modo se mejora el rendimiento del juego.

Los otros objetos padre son el mundo abierto, los personajes, el Canvas, y otros que tienen parámetros del juego, como el GameManager(Figura 1).

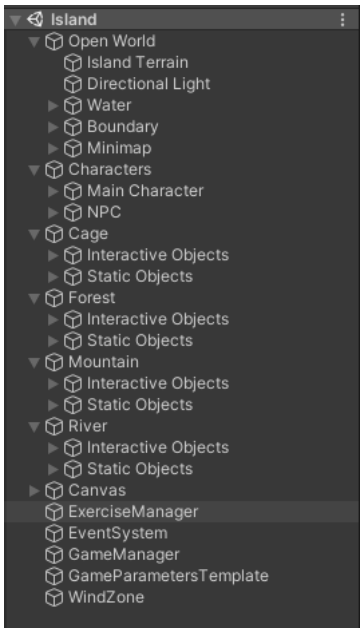

Figura 1. Objetos de la escena principal

Open World:

Incluye todos los objetos generales del juego, el terreno, la luz, el agua, los límites que corresponden a los checkpoints, y el mini mapa. Aquí deben incluirse los elementos generales para toda la isla. La disposición de los objetos sigue la misma estructura que los anteriormente mencionados(Figura 2). Los objetos padres tienen una posición y rotación 0,0,0.

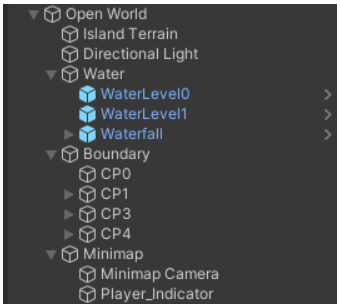

Figura 2. Objetos de Open World

Characters

Aquí aparecen dos categorías, el personaje principal y los NPC. La filosofía de posición y rotación de los objetos es la misma que para los objetos anteriores(Figura 3).

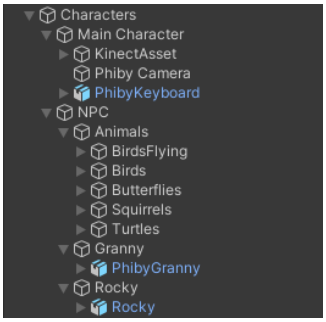

Figura 3. Objetos dentro de Character

Si las nuevas escenas creadas siguen la misma organización y estructura, se vuelve más fácil pasar objetos de una escena a otra, por tanto, se ha decidido que esta es la organización óptima que utilizar.

Una vez realizada esta estructuración, se analizaron los scripts que existían en el juego, se hizo una limpieza de los mismos y se organizaron y analizaron para tener una especie de base de datos con los mismos. Esta información puede encontrarse en el GDD del videojuego.

3.2. Creación de iconos

Otra de las acciones realizadas fue crear unos iconos para indicar en el modo de juego que se está jugando. Ya que, aunque el juego está pensado para ser jugado exclusivamente en el modo Kinect, al poder pasar de un modo a otro no viene mal tener un indicador del modo al que se está pasando. Para ello se crearon dos iconos los cuales se muestran en la Figura 4.

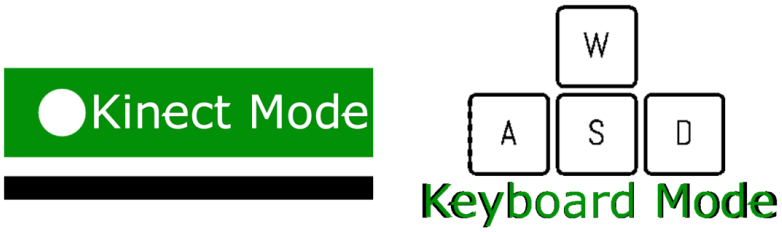

Figura 4. Iconos Kinect y Teclado

3.3. Mejora Rocky

A la hora de jugar con la Kinect, en la parte del juego en la que aparece Rocky por primera vez, había una serie de problemas de jugabilidad, por ello, se decidió modificar esta parte de la escena para darle mas sentido a la historia.

Ahora este está encerrado en la cueva que Ra ha derrumbado. Este le ha robado su superpoder y tirado al bosque para que no pueda salir de la cueva. Con los prefabs de las rocas, se ha cubierto la entrada de la cueva (*RockyTrap*). Cuando Phiby se acerca a ella, Rocky, le cuenta lo que ha sucedido, y Phiby tiene que ir a recuperar el super poder de Rocky que se encuentra en el bosque. Cuando Phiby se acerca con el superpoder, las rocas que tapaban la entrada de la cueva explotan. Para crear ese efecto, se ha creado una animación, "RockExplosion", que se activa cuando se acerca Phiby con un Halo activado, que simula el superpoder(Figura 5).

Esta animación se ha creado haciendo uso de Animation del editor de Unity, cambiando la posición de las rocas en función del tiempo.

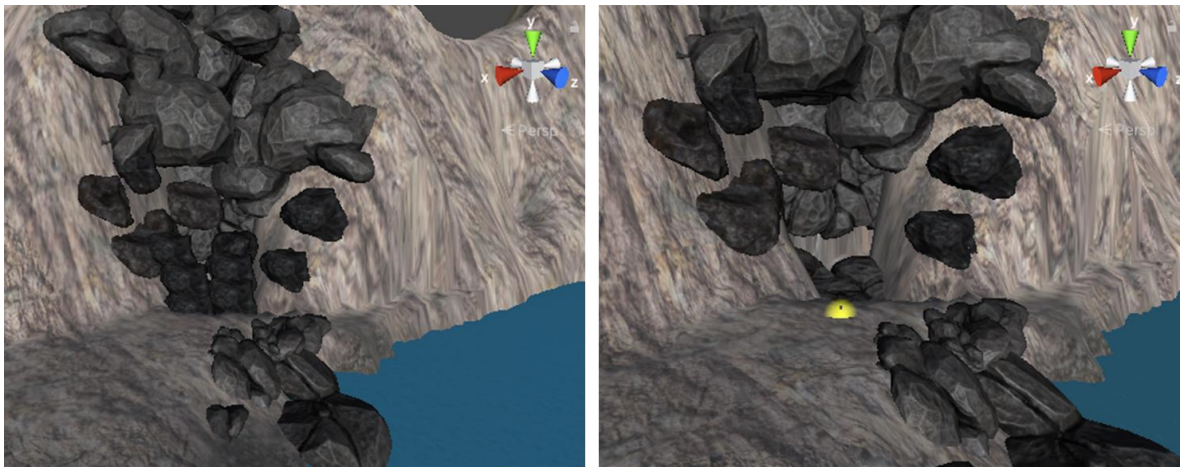

Figura 5. Nuevo escenario en el que se encuentra Rocky

El superpoder consiste en una seta luminosa, la cual está escondida, pero a la vez brilla con un haz de luz dorado (Figura 6).Una vez que se recoge, Phiby se ilumina del mismo color simulando que ahora mismo él tiene el super poder.

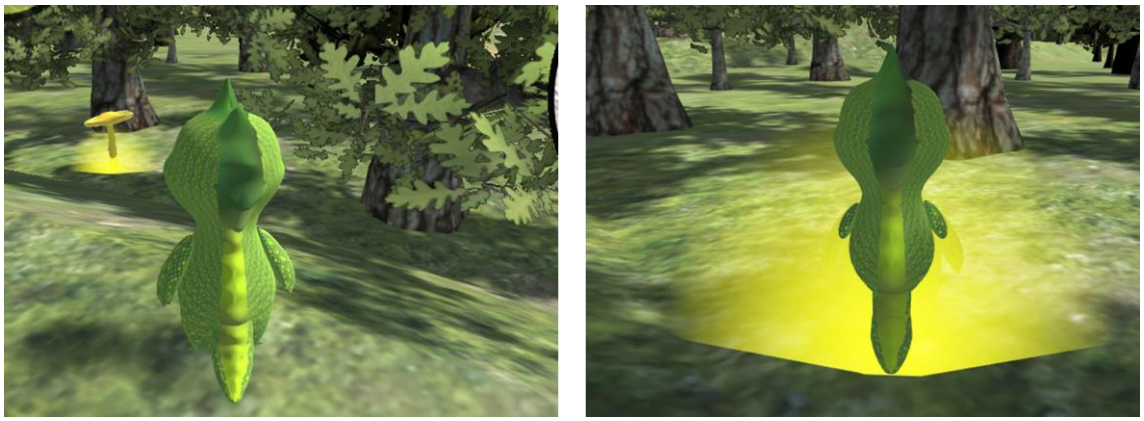

Figura 6. SuperPower antes y después de ser recogido por Phiby

Cuando se acerca a Rocky y las piedras explotan, Rocky sigue a Phiby y cuando esté está lo suficientemente cerca, Phiby se ilumina ya que Rocky está compartiendo la superfuerza con su hermano (Figura 7).

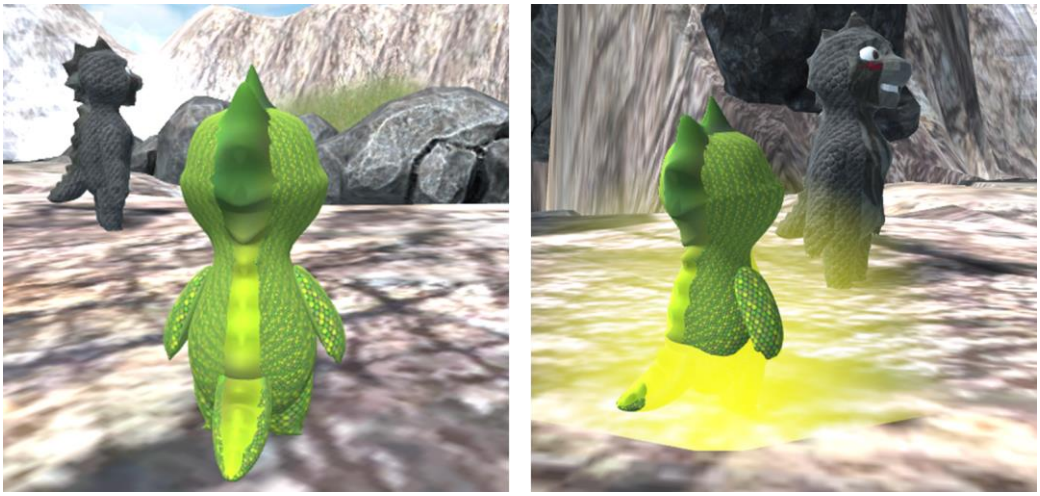

Figura 7. Phiby cuando está lejos y cerca de Rocky, sin y con el super poder de fuerza.

Solo se podrá entrar en el ejercicio del puente, si Rocky se encuentra cerca de Phiby. Este comportamiento se determina tanto en el script SuperPower.cs como en el RockyController.cs

3.4. Doblaje

Como última parte del semestre, se realizaron nuevos doblajes, ya que la mecánica del juego había cambiado durante este periodo de tiempo. Por tanto, la voz de la abuela cambió. Este doblaje se realizó usando AdobeAudition.

creando un archivo de audio con una velocidad de muestreo de 48 kHz, estéreo y con una profundidad de 32 bits. Para modificar el tono de voz primero se ha realizado una puerta de ruido para eliminar el posible ruido de fondo. A continuación se ha ecualizado el archivo con un ecualizar gráfico, amplificando las frecuencias mas graves, después se ha realizado una deformación de 2 semitonos y finalmente se ha normalizado la señal en 0dB. En la siguiente figura se muestran cada una de las ventanas aplicadas para llegar a la resolución final de la voz.

Todo este proceso queda reflejado en la Figura 8 por si es necesaria su recreación en un futuro.

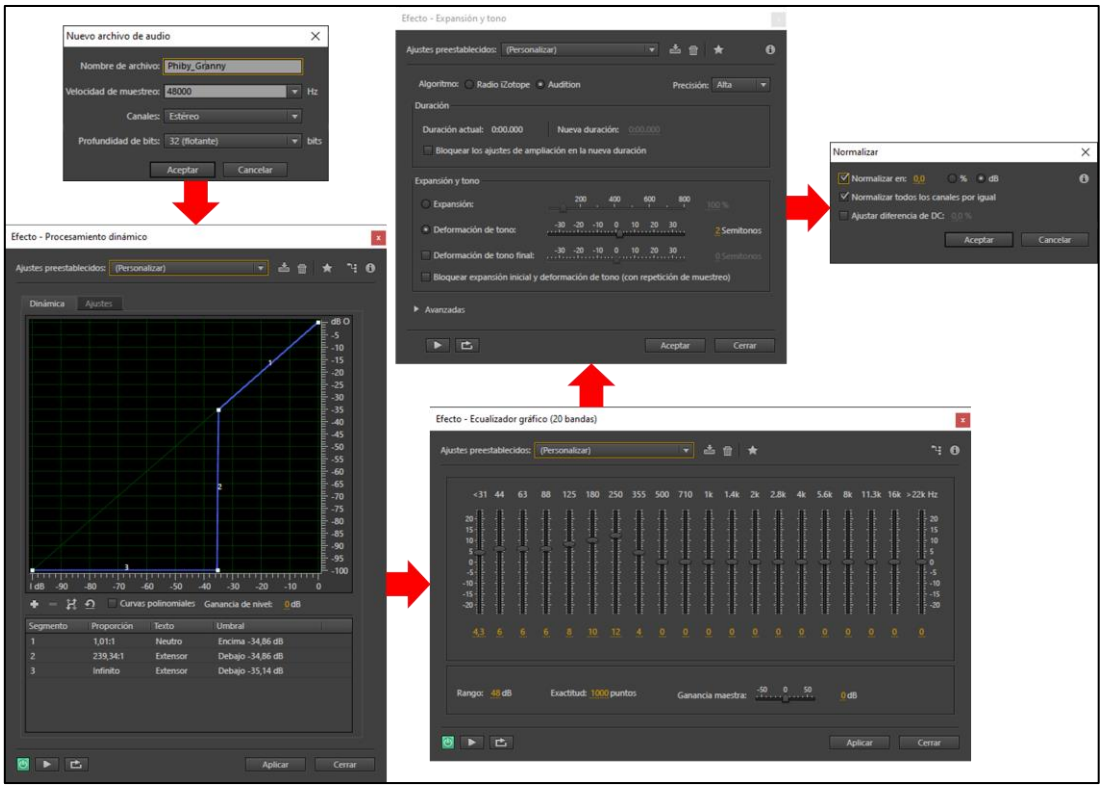

Figura 8. Creación del doblaje de Granny

3.5. Otros

Se han realizado otras tareas de menor tamaño que no forman parte de una categoría específica. Entre ellas se encuentran:

- La creación de un script para dar movimiento a los pájaros que al inicio del proyecto estaban estáticos (Birds.cs).
- Además, se ha modificado justo a los compañeros Laura y Dario, el terreno de la isla para mejorar la jugabilidad.

- Traducción y revisión de los scripts. Se han añadido comentarios y los que estaban en castellano se han traducido al inglés.
- Redacción de la documentación del GDD con el trabajo realizado durante el semestre.

4. Segundo Semestre

La segunda parte de la beca se ha desarrollado en el segundo semestre (enero-mayo). Durante estos meses se ha trabajado junto a Marta Sanz Gómez alumna que estaba realizando el Proyecto de Fin de Grado. Este semestre el trabajo se ha realizado de manera más individual en comparación con lo realizado en el primer semestre. Ambas han colaborado juntas, pero no se ha llegado a trabajar en la misma tarea al mismo tiempo.

Las tareas principales de este semestre se basan en continuar con los ejercicios y a partir de esta primera tarea, derivó la tarea de tener un sistema de tratamiento de parámetros más ordenado y lógico del previamente implementado, que no tenía en cuenta la necesidad de diferentes parámetros para diferentes ejercicios.

4.1. Creación del ejercicio ClimbPhiby

Este semestre se inició creando un ejercicio nuevo, ClimbPhiby. Se ha diseñado el ejercicio de trepar la jaula. Una vez se ha entrado en el minijuego, Phiby aparece en una posición estática sobre la que irá escalando la jaula. Los parámetros que se tienen en cuenta en este ejercicio son el tiempo, y el número mínimo de movimientos, cuyo valor ha de ser pequeño, ya que este ejercicio ha de ser sencillo y rápido.

Para superar el ejercicio, se tiene en cuenta la distancia máxima que la persona pueda levantar cada uno de los brazos (maxLeft y maxRight) que actualmente está definido a 4 (era el valor que se obtenía si Henar Redondo levantaba los brazos). Este valor es obtenido del character IKTarget tanto L como R.

La energía que se descuenta de cada movimiento es 1, ya que no se quiere que nada más empezar el juego el jugador tenga que conseguir más energía. Para superar el ejercicio, hay que levantar los brazos alternadamente, no es necesario que estos estén en una posición especial.

Para superar el ejercicio, existen dos colliders, uno de ellos se encuentran en cada final del brazo de Phiby (Figura 9, izquierda) y un trigger (Figura 9, derecha) el cual hace de Target en el ejercicio.Levantando los brazos de manera alternada derecha-izquierda cuando el collider del brazo entre en el trigger del target tanto Phiby como el GameTarget ascenderán. El número de movimientos que se realizan están definidos en los parámetros y la cantidad de altura que sube Phiby es proporcional a la altura de la jaula y al número de movimientos.

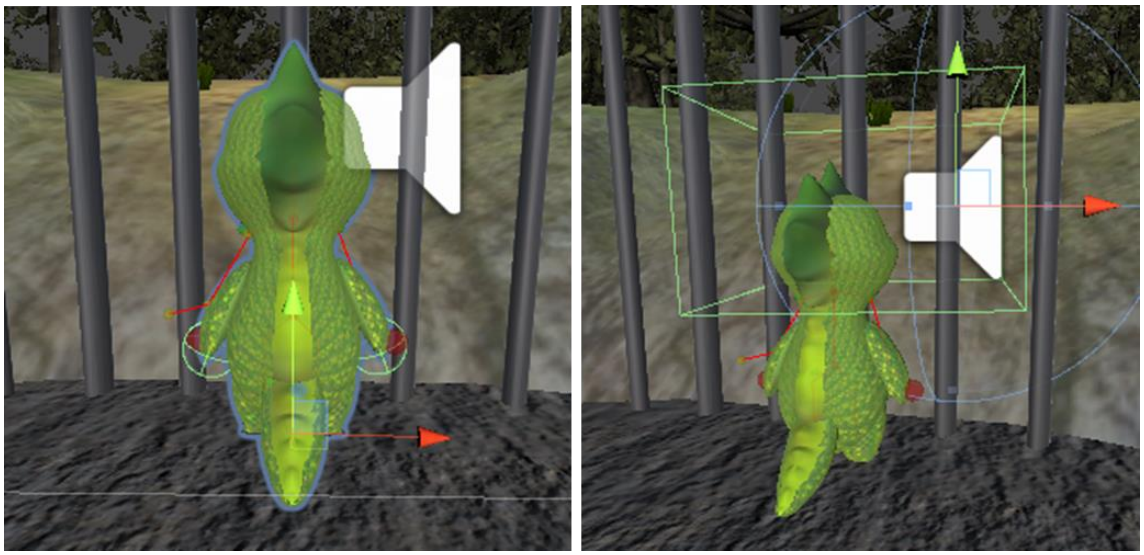

Figura 9. ClimbCage colliders y trigger/target (izquierda y derecha respectivamente)

4.2. Mejora del ejercicio Apple Tree

Para superar el ejercicio Apple Tree Climb, hay que hacer un movimiento específico que consiste en levantar el brazo haciendo una circunferencia y situando el brazo cerca de la cabeza(lo más pegado posible al tronco/eje del jugador). Para conseguir este objetivo, está definido por script las distancias mínimas para considerar el movimiento correcto, además, dentro de la escena el final del brazo de Phiby tiene que entrar en un trigger definido como el Target. Dicho Target era de un tamaño muy reducido por lo tanto era muy complicado realizar el movimiento exacto. Para reducir un poco la dificultad del ejercicio se ha aumentado el tamaño de dicha esfera.

4.3. Parámetros de ejercicios

Por último y como finalización de la beca, se ha creado un sistema de scripts para el tratamiento de los parámetros de los ejercicios tanto por defecto como los descargados de la web. Se ha partido de unos scripts y filosofía previa y se ha actualizado, ya que originalmente el videojuego solo tenía el ejercicio de las manzanas y todos los parámetros y configuraciones se basaban en este. Los objetos que intervienen en este proceso son el DataManager, KinectReceiver y el GameManager o el MenuIntro.

Para controlar los parámetros, se ha creado un objeto llamado DataManager, cuyo objetivo es contener y gestionar la utilización de los parámetros. Este objeto es creado por GameManager.cs y MenuIntro.cs y es un objeto que no se destruye. Dentro de este objeto, se encuentran tres scripts diferentes:

- `ConfigSettings.cs` Este scripts es el que almacena y trata los datos que se encuentran en las listas. Al inicio del juego, carga los parámetros por defecto, y estos se actualizan una vez han sido descargados de la Kinect a partir del script `SettingManager.cs`. Este script está pensado para contener los métodos que sean necesarios para el tratamiento de las listas y los parámetros de las mismas.
- `GameState.cs`: antiguamente era `GameSettingSustain.cs` pero se ha modificado el nombre ya que no era aclaratorio. En este script se controlan los datos de la partida se está jugando.

- `ExerciseResults.cs` es el encargado de crear una lista con formato ResultList (nueva lista definida también en `WebClasses.cs`) para dársela al kinectReceiver y que este la envíe al middleware y así obtener los resultados de cada ejercicio. Cuando se realiza y termina un ejercicio, siempre se van a enviar los resultados obtenidos.

Para almacenar los parámetros, se hace uso de listas. Anteriormente se utilizaba un diccionario, pero se ha decido usar listas ya que son más accesibles y visuales, y al no necesitar una gran cantidad de parámetros no es necesario hacer uso de diccionarios. Estas listas están definidas en la clase `WebClasses.cs` y el script en el cual se van a almacenar los parámetros es el `ConfigSettings.cs`, en la variable pública ExerciseList.

Los parámetros por defecto se encuentran en un archivo JSON `DefaultParameters.json` (Figura 10) que tiene la misma estructura que un ExerciseList. Dichos parámetros se encuentran en el directorio: Assets/Resources/Parameters. Esta carpeta sería la idónea para en un futuro almacenar y guardar cualquier tipo de archivo que contenga algo relacionado con los parámetros.

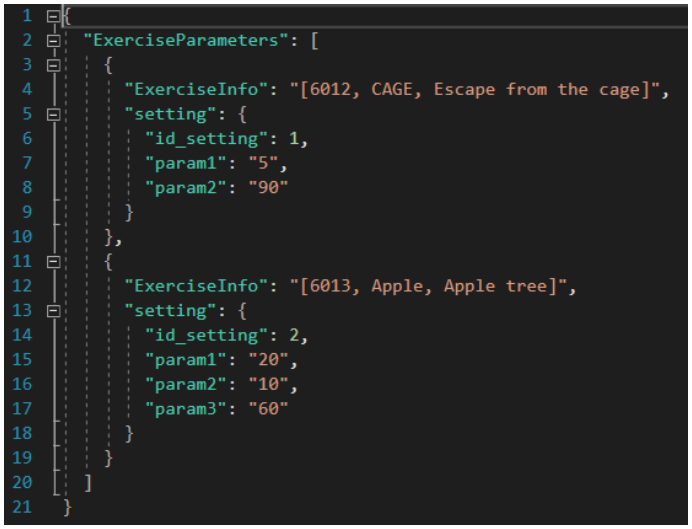

Figura 10. DefaultParameters.json

Por otra parte, el script `exerciseManager.cs` ha sido modificado, actualmente es el encargado de almacenar los datos del ejercicio que actualmente se está jugando, cuando se accede al ejercicio, este script almacena los datos del ejercicio a jugar, y cuando se finaliza se eliminan. Anteriormente, este script se creaba cada vez que se accedía a la escena Island, ahora se crea una única vez y se reutiliza.

A este script, acceden los scripts de `ClimbMovementController.cs` y `EventMng.cs` para así utilizar los parámetros necesarios para definir el ejercicio.

Una vez que se ha completado el ejercicio, el script `ExerciseResult.cs`, es el encargado de enviar los datos que se han obtenido al KinectReceiver y elimina los parámetros del juego actual del `exerciseManager.cs`.

Como resumen a continuación se muestran los pasos que se van ejecutando para cargar la configuración de parámetros (Figura 11):

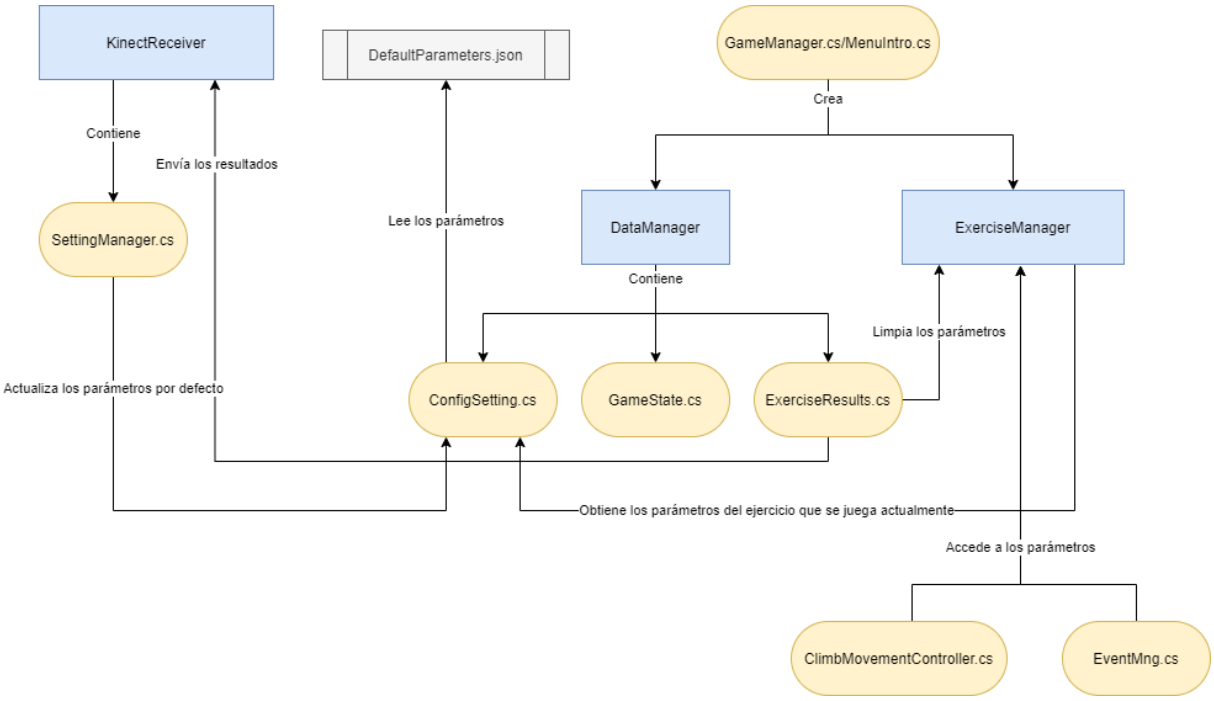

Figura 11. Diagrama de bloques parámetros.

1) El `GameManager.cs` o el `MenuIntro.cs` crean el DataManager y el ExerciseManager con los scripts que estos tienen asociados.
2) `ConfigSetting.cs` carga los parámetros por defecto del DefaultParameters.json
3) Si la Kinect está conectada, `SettingManager.cs` actualiza los parámetros del paciente y sustituye los por defecto por los recibidos por la Kinect.
4) Al entrar en un juego, el `exerciseManager.cs` carga los parámetros del ejercicio al que entra en su lista CurrentExercise.
5) `EventMng.cs` y `ClimbMovementController.cs` acceden a los parámetros del `exerciseManager.cs` para definir el ejercicio.
6) Cuando el ejercicio ha finalizado, `ExerciseResults.cs` envía los resultados al KinectReceiver y elimina los parámetros del `exerciseManager.cs`

5. Conclusiones

Durante la realización de la beca se han conseguido cumplir con los objetivos marcados de una manera satisfactoria. Esta beca ha sido gran oportunidad de empezar en el mundo laboral colaborando con diversos compañeros y realizando diferentes tareas. Los diferentes objetivos han sido muy diversos y completos, y la realización de estas tareas han permitido entender el completo funcionamiento del juego.

Como conclusión obtenida de esta beca, se puede determinar que a la hora de realizar un proyecto hay que organizarlo desde el principio con una serie de normas y bases y una buena documentación que permita que cuando nuevos programadores empiecen a trabajar en dicho proyecto el periodo de adaptación sea mucho más sencillo.

6. Bibliografía

[1] "Documentación para el diseño de Phiby's Adventure 3D"

[2] C. Luaces Vela, Diseño e implementación del entorno virtual de ejercicios físicos, basado en captura de movimiento, Proyecto fin de grado, Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, UPM, Madrid, Julio 2018.

[3] C. L. Vela, "Diseño e implementación de un entorno virtual de ejercicios físicos, basados en captura de movimiento.," UPM, ETSIST, Madrid, 2018

[4] D. I. Canelo "Informe Beca de Colaboración" UPM, ETSIST, Madrid,2019