ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Integración y calibración de movimientos captados mediante la cámara Kinect para el juego serio terapéutico "Phiby's Adventures v2"

AUTOR: Miguel Ángel Gil Gil

TITULACIÓN: Grado en Ingeniería de Imagen y Sonido

TUTOR: Martina Eckert

DEPARTAMENTO: Teoría de la Señal y Comunicaciones

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Fco. Javier Ramírez Ledesma

TUTOR: Martina Eckert

SECRETARIO: Enrique Rendón Angulo

Fecha de lectura: Julio 2019

Calificación:

El Secretario,

Agradecimientos

A los todos los profesores que, de un modo u otro, me han enseñado todos los conocimientos y valores necesarios para poder llegar hasta este punto determinado. En especial a mi tutora, Martina, por haber confiado en mí para la realización de este proyecto.

A mis amigos, por haber evolucionado conmigo durante todos estos años y haber compartido tanto los malos como los buenos momentos de esta carrera.

A mi familia, por haber creído en mí desde el principio y por haberme apoyado siempre.

Resumen

En la actualidad, los proyectos de investigación vinculados a la rama de la medicina cobran vital importancia ya que permiten ayudar en el tratamiento de los problemas de salud. Estos proyectos implementan la movilidad física del jugador dentro de un videojuego, para lograr fines terapéuticos mientras disfrutan jugando.

Las herramientas tecnológicas que se han empleado en este trabajo son, en primer lugar, herramientas software para el desarrollo del videojuego (*Unity*) y para la generación de animaciones (*Blender*). En segundo lugar, se emplea el hardware específico para capturar los movimientos del jugador (*Microsoft Kinect v2*)

Este documento describe el proyecto realizado, el videojuego "*Phiby's Adventures v2"*, basado en la creación de entornos completamente configurables y personalizables para la recreación de cada jugador. Los ejercicios de movilidad física son capturados mediante el *hardware Kinect v2* y actúan con fines terapéuticos en la rehabilitación del jugador. Además, un terapeuta puede realizar el seguimiento de las actividades realizadas por el jugador a través de una web médica. A su vez, se han desarrollado tres animaciones asociadas al personaje: reposo, andar y correr.

En el desarrollo de este proyecto se ha implementado el control del personaje *Phiby* mediante el movimiento del cuerpo del jugador y mediante teclado. También se han integrado una serie de ejercicios específicos que conllevan: movimiento de brazos y movimiento del tronco. De esta forma, el jugador puede talar un tronco, escalar un árbol, volar en ala delta o esquiar. Los movimientos de los brazos pueden ser amplificados, de forma que cualquier persona pueda jugar, independientemente de las capacidades físicas que posea.

Abstract

Nowadays, the importance of the research projects linked to medicine has been increased due to its help in the treatment of health problems. These projects implement the physical mobility of the player inside the videogame, to achieve therapeutic purposes while the players enjoy playing.

The technological tools that have been used in this project are, in the first place, software tools for game development (*Unity*) and for animations' generation (*Blender*). Secondly, specific hardware is used to capture the player's movement (*Microsoft Kinect v2).*

This document describes the carried out project, the videogame "*Phiby's Adventures v2"* based on the creation of completely configurable and customizable environments for each player. The physical mobility exercises are captured by the Kinect v2 hardware and they act for therapeutic purposes in the player's rehabilitation. Besides that, a threpist can track the activities performed by the player through a medical web. At the same time, three animations (idle, walk and run) assigned to the character have been developed.

In the development of this project, the Phiby's control with the player's body and keyboard has been implemented. It has also been integrated a series of specific exercises that involve: arms and body movements. In this way, the player can cut a log, fly in hang glider, climb a tree or ski. The arms' movements can be amplified, so that anyone can play regardless of their physic limitations.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

1.1 Introducción general

Hoy en día hay muchas posibilidades de aplicar los avances tecnológicos a temas de salud. En concreto, en el área de rehabilitación física, se está investigando mucho el uso en el uso de sistemas virtuales de captura de movimientos, que podrían aportar más ventajas tanto para el paciente como para el terapeuta. Hay enfoques que combinan los videojuegos con los movimientos, de tal manera que el paciente pueda usarlos en combinación con su terapia convencional, para mejorar su estado de salud.

La Universidad Politécnica de Madrid (UPM), y más concretamente, el grupo de investigación GAMMA [1] (Grupo de Aplicaciones Multimedia y Acústica) situado dentro del CITSEM [2] (Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad), está desarrollando proyectos dentro de este ámbito. Estos proyectos combinan la captura y el seguimiento de los movimientos del jugador con tareas de rehabilitación, con el objetivo de lograr fines terapéuticos.

Este documento describe una parte de desarrollo llevado a cabo dentro del proyecto "*Phiby's Adventures v2".* Se trata de un juego que pretende ser configurable y personalizable para cada jugador. Los movimientos que realiza un paciente para avanzar en el juego, en realidad son una serie de ejercicios especificados para su rehabilitación, que ejecuta sin darse cuenta. El objetivo es que, sobre todo, personas con diversidad funcional motora puedan realizar tareas y actividades específicas de rehabilitación, mientras se divierten y disfrutan jugando.

El entorno global del proyecto *"Phiby's Adventures v2",* en el cual se ubica este proyecto de fin de grado, es el proyecto *BLEXER* (*Blender Exergames*). El videojuego será sucesor de la primera versión *"Phiby's Adventures v1*" que se basa en una versión anterior de la cámara de captura de movimientos y un software de desarrollo más limitado. El juego se caracteriza por tener un mundo acotado y escenas repetitivas, cosas que se quieren mejorar en esta nueva versión, manteniendo el mismo personaje llamado *Phiby*.

Se pretende que el control de los movimientos sea ahora más preciso y fiable en comparación con la primera versión, usando la cámara *Microsoft Kinect V2* de la consola *XBOX One* para la captura de los movimientos. En la primera versión del videojuego, se utiliza la *Microsoft Kinect V1,* de la consola *XBOX 360* [3].

El objetivo principal de los videojuegos desarrollados dentro del entorno BLEXER es que debe ser perfectamente jugable por personas que tengan limitaciones en sus movimientos, así como discapacidad y, en base a ello, elaborar los ejercicios necesarios para su rehabilitación. Por lo tanto, los juegos se calibran para adaptarse ante las distintas capacidades de los jugadores, p.ej. a su rango de movimiento. Si el rango es limitado, los movimientos pueden amplificarse para que no suponga un sobreesfuerzo para el jugador llevar a cabo algunos de los ejercicios. Además de esta forma no se presentan diferencias entre jugadores que tengan más rango de movimiento que otros, ya que el personaje siempre completa el movimiento independientemente del ángulo adoptado por el jugador.

El entorno BLEXER prevé también que un terapeuta pueda seguir las actividades realizadas por el jugador y configurar la dificultad de los niveles a distancia. Para esto se personalizan múltiples parámetros en función de las capacidades y habilidades de cada uno.

El entorno de desarrollo software utilizado para la segunda generación de juegos (basados en la *Kinect V2*) es *Unity* [4]. La comunicación entre la *Kinect V2* y el juego se lleva a cabo a través del *middleware* "*K2UM"* desarrollado por César Luaces Vela, antiguo alumno de la Escuela y recogido en su trabajo de fin de grado [5].

Este desarrollo del videojuego *"Phiby's Adventures v2"* se está iniciando mediante el presente proyecto de fin de grado junto a otros tres compañeros: Laura Molero (PFG), José David Guerrero (prácticas externas) y Juan Alberto García (PFG). Entre los cuatro, se lleva a cabo el diseño del entorno, los objetivos y el funcionamiento del primer nivel y la comunicación con el *middleware* para realizar la personalización del juego. Se han distribuido las responsabilidades en esta primera fase de tal manera que Laura es la encargada del diseño de la isla en la que tiene lugar el juego y todos sus elementos, Juan Alberto se encarga de establecer la comunicación con la web donde el terapeuta sigue los resultados del jugador, así como de configurar cada ejercicio según las capacidades de cada jugador y José David aporta soporte técnico durante el desarrollo del videojuego. El objeto del proyecto presente sin embargo es la implementación y el ajuste de los movimientos del jugador para conseguir un control libre del personaje dentro de un mundo abierto e incluir diferentes escenas en las que se aplican los cuatro movimientos ya desarrollados y comprobados con anterioridad, que corresponden a los ejercicios de rehabilitación: remar, trepar, cortar troncos y bucear/volar. A continuación, se explican los objetivos más detallados de este proyecto de fin de grado.

1.2 Objetivos del proyecto

Mejora y animaciones del personaje *Phiby*.

Se trata del personaje principal del videojuego, un pequeño anfibio que representa al jugador. Este personaje se debe mover libremente en el entorno principal del videojuego (una isla), de tal manera que el jugador puede explorar el entorno. Cuando se encuentre con determinados objetos o llegue a determinados sitios de la isla, se ejecutan escenas específicas en las que el jugador realiza un ejercicio físico para conseguir puntos y avanzar en el juego. Para proporcionar una base de funcionamiento en este sentido, se deben solucionar las siguientes tareas en el entorno de este proyecto de fin de grado:

- Mejora del modelo 3D del personaje creado en Blender por la antigua alumna Cristina Esteban en su proyecto de fin de grado [6], quitándole complejidad que afectaría a la ejecución fluida del videojuego.
- Añadir las animaciones principales de las que dispone el personaje: reposo, andar y correr.

Ejercicios, manejo del personaje y control del juego.

El principal objetivo es que el jugador maneje todo el juego sin la necesidad de utilizar el teclado o ratón. Para conseguir esto, se deben distinguir diferentes tipos de movimientos del jugador durante el juego:

- Movimientos para **mover al personaje**: se trata de movimientos copiados directamente del jugador. Se utilizan para llevar a cabo los ejercicios y para controlar a *Phiby* dentro de la isla.
- Movimientos para **realizar los ejercicios**: Dado que no todas las personas que jueguen van a tener las mismas capacidades físicas y los mismos rangos de movimiento, es necesario, que algunos movimientos sean **amplificados** para que no supongan un esfuerzo adicional sobre el jugador. Así, se evita que este se frustre y que el personaje se mueva igual que cuando juegue una persona sin limitaciones físicas.
- Movimientos de control **ajenos al juego**, p.ej. para abrir el menú o ejecutar una animación del personaje que el jugador no puede simular dado que se encuentra frente a la *Kinect* (p.ej. andar y correr). Estos movimientos se deben definir como específicos que, cuando el juego los detecte, ejecuten un hecho determinado. Concretamente, se implementará que una inclinación del cuerpo del jugador provoque que el personaje avance hacia delante. También puede utilizarse para lanzar acciones adicionales en el juego, como, por ejemplo, activar el menú de pausa levantando una mano.

Control del personaje para el terapeuta o desarrolladores.

Se añade el manejo del personaje mediante el teclado como funcionalidad añadida para desarrolladores y terapeutas. Esto podría ser de utilidad, p.ej. para:

- Enseñar el entorno del juego en demostraciones o para introducir a un jugador nuevo.
- Probar nuevos entornos sin la *Kinect* conectada.
- Probar nuevas animaciones desarrolladas.
- Realizar tareas de mejora del personaje sin necesidad de conectar la *Kinect*.

Ejercicios físicos.

El objetivo principal es la ejecución de ejercicios físicos a realizar durante el juego que actúen con fines terapéuticos en el jugador. Hay cuatro movimientos que ya se probaron en trabajos anteriores [7] y que se implementaron en la versión 1 del videojuego [8], con lo cual han de mantenerse para fines comparativos. Estos movimientos son: movimiento del brazo derecho de arriba hacia abajo (hasta ahora implementado como golpeo y tala de troncos), movimiento de ambos brazos verticalmente de forma alterna (implementado como trepado de escalera o árbol), movimiento de ambos brazos hacia delante y hacia atrás (implementado como remar) y movimientos de control de tronco (implementado como volar en ala delta y bucear). Los objetivos concretos son:

- Crear escenas genéricas en las que el jugador haga estos movimientos. No tienen que ser necesariamente trepar, talar o remar, pero se deben aplicar estos mismos movimientos. El objetivo es que se puedan aplicar en diferentes sitios para tener más variedad (p.ej. se puede trepar árboles, barrancos, muros, casas etc. según el entorno del juego).

- Las escenas deben estar preparadas para poder ser configuradas mediante variables que determinan la dificultad del ejercicio, p.ej. la altura del árbol a trepar debe poder ajustarse (aunque este ajuste se realizará en el proyecto del compañero Juan Alberto García).
- Asegurar que el comportamiento de la *Kinect* es robusto y preciso en diferentes ambientes y con distintos elementos presentes en el entorno (muebles de fondo, personas o distinta iluminación de ambiente).

2. Antecedentes y entorno del proyecto

2.1.1 Aplicaciones comerciales

En la actualidad existen numerosos proyectos tanto en desarrollo como ya presentados, que emplean la captura de movimientos dentro de videojuegos para ejercer una serie de tareas que puedan sumarse a las actividades de rehabilitación de una persona. Estas tareas de rehabilitación pueden ir enfocadas tanto a ejercicios corporales como a ejercicios mentales.

Muchos de estos proyectos han alcanzado tasas de éxito muy elevadas, sirviendo como complementos adicionales a la rehabilitación regular. Todos emplean distintos elementos hardware para la captura del movimiento. Se destacan a continuación algunos ejemplos:

- REHABILITY

  Se trata de un proyecto que desarrolla videojuegos tanto físicos como cognitivos para llevar a cabo ejercicios de rehabilitación en sus jugadores [9]. Gracias a él, los pacientes pueden ejercitarse en el mismo centro en el que se encuentran hospitalizados o bien desde sus propios hogares. Los ejercicios se pueden desarrollar de forma autónoma, pero siempre bajo una constante supervisión médica.

  Se descompone en dos vertientes principales. La primera está especialmente enfocada a niños (*REHABILITY Kids*). La segunda, especialmente dirigida hacia la tercera edad (*REHABILITY Lite*).

  Finalmente, mediante *REHABILITY Home*, se establece una comunicación entre el hogar del paciente y la clínica. De esta forma, se añade el soporte que sea necesario, así como una continua evaluación de los procesos del paciente en cuestión.

- ADVANT

  Se trata de una plataforma destinada a la rehabilitación física y al entrenamiento cognitivo mediante la realización de ejercicios que implican movimientos por parte del propio usuario [10]. De forma similar a este proyecto, también se emplea la *Microsoft Kinect*.

  Esta plataforma permite generar ejercicios de manera sencilla mediante un configurador. Así, se pueden crear ejercicios específicos para cada jugador y se evitan que sean los mismos ejercicios repetitivos una y otra vez.

- VirtualRehab

  Se trata de una plataforma de rehabilitación pionera con mercado en la comunidad europea (CE) que ayuda a mejorar la terapia tradicional [11]. Se llevan a cabo tareas de

rehabilitación tanto en clínicas como en el hogar a través de tareas específicas simuladas en un videojuego. El catálogo de servicios ofrecido es muy amplio:

- *VirtualRehab Body*: Se trata de un módulo de terapia para la rehabilitación de extremidades superior e inferior.
- *VirtualRehab Hands:* Lleva a cabo ejercicios prácticos para la mejora de las habilidades motoras de las manos. Permite entrenar la destreza, la fuerza y el rango de movimientos.
- *Mindplay:* Aún no está disponible, pero se trata de un conjunto de ejercicios que permiten mejorar las y entrenar distintas funciones cognitivas.
- *Cognirehab:* Tampoco está disponible aún, pero se trata de una aplicación de realidad virtual que permite entrenar la función ejecutiva dentro de un entorno diseñado.

2.2 Plataforma Blexer-med

La plataforma Blexer-med [12] alberga diferentes videojuegos, siendo este proyecto uno más de ellos. Este proyecto supone una continuación a trabajos previos realizados por antiguos alumnos durante el desarrollo de sus respectivos proyectos. Por lo tanto, además de emplear el contenido del que ya se dispone, se ha creado contenido nuevo adicional. Se comienza durante las primeras fases del proyecto Blexer con la captura de movimientos empleando la cámara de *Microsoft Kinect V1*. El desarrollo durante esta primera versión está hecho en *Blender* [13]. La comunicación entre *Blender* y la *Kinect V1* emplea un *middleware* desarrollado por Ignacio Gómez-Martinho González denominado *"Chiro"* [14] y un *plugin* de recepción de datos implementado para *Blender*. Con ella, la cámara recoge todos los movimientos que realiza el jugador y envía la información al videojuego.

Figura 1. Cámara Microsoft Kinect V2

"*Phiby's Adventures v1"* establecía una comunicación con una plataforma web médica [15] donde se almacenaban los progresos del jugador y el terapeuta podía realizar tareas de seguimiento sobre el mismo. Como renovación del *middleware "Chiro"* surge el *middleware "K2UM" (Kinect To Unity Middleware)*. El primer juego implementado en *Unity* y *Kinect* es "*Buscando a Totoro"* [16], a parte del proyecto [5].

La siguiente fase del proyecto continúa con *"Phiby's Adventures V2*", donde se emplea la Kinect V2 y el entorno de desarrollo *Unity*. Mediante la interfaz de usuario *K2UM* se establece la comunicación entre la *Kinect V2* y el videojuego, enviando los datos de los movimientos del jugador. Este *middleware* recibe datos a través del puerto 8050 y emite a través del puerto 8052. A su vez, la aplicación recibe por el puerto 8051 y emite por el puerto

8053. Gracias a esto es posible controlar y gestionar el estado de la aplicación. Se adjunta a continuación una captura correspondiente a la interfaz *K2UM* en la figura 2.

Figura 2. Interfaz K2UM

El hecho de emplear un *middleware* permite funciones adicionales como que pueda ser utilizado en cualquier otro videojuego desarrollado o permitir una comunicación simultánea con la *web*. Esta última funcionalidad se encuentra en paralelo con el desarrollo de este proyecto y por tanto no se dispone de una versión estable para utilizar.

Los movimientos del jugador quedan recogidos a través de la *Kinect*. Esta, a su vez, envía todos los datos recogidos a través del *middleware K2UM* al juego, dentro del entorno de desarrollo *Unity* para que sea capaz de interpretarlos y los aplique al personaje. Por último, la configuración de los parámetros dentro del videojuego, así como la personalización de este se llevan a cabo de forma remota a través de una web donde se comunica con el terapeuta. De esta forma, se puede seguir la evolución del paciente desde la página web y configurar las distintas sesiones o ejercicios según evolucione el jugador.

Se adjunta a continuación un diagrama en la figura 3 donde se recoge la estructura del proyecto descrito anteriormente.

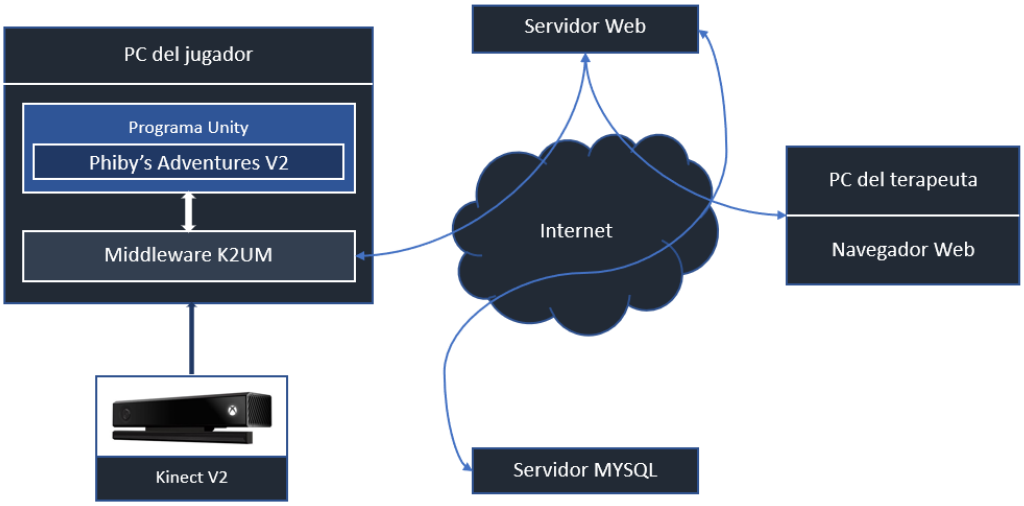

Figura 3. Diagrama estructural del proyecto

El *software* de desarrollo utilizado se muestra en la figura 4. Corresponde con una captura del entorno donde se llevan a cabo las tareas de edición e implementación del videojuego. Es un motor muy potente que combina la implementación de métodos propios y añadidos mediante código en los lenguajes de programación *C#* y *JavaScript*. Entre sus ventajas, destacan:

- Multiplataforma (compatibilidad tanto con *Windows* como con *Mac*).
- Dispone de una tienda donde encontrar una gran variedad de productos (algunos gratuitos) denominada "*Asset Store"*.
- Licencia de uso gratuita.
- Fácil manejo y control del entorno en una interfaz muy intuitiva.

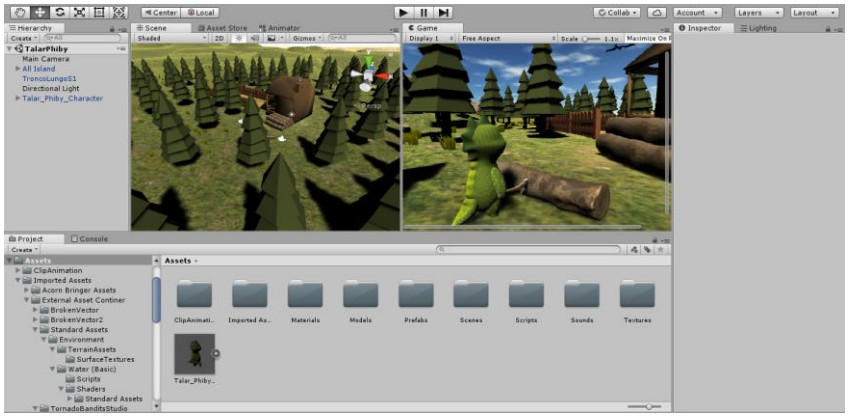

Figura 4. Entorno de desarrollo Unity

3. Personaje principal del videojuego "*Phiby*"

3.1 Descripción del personaje

El personaje que se utiliza durante todo el desarrollo del videojuego corresponde con un modelo, ya realizado previamente para la versión anterior, llamado "*Phiby*". Se adjunta una figura a continuación. Se trata del modelo representado con forma de anfibio, escogido y diseñado específicamente para un videojuego donde la gran mayoría de jugadores son niños.

Figura 5. Representación en Blender del personaje Phiby

3.2 Reducción de vértices, caras y polígonos del personaje

Inicialmente, el modelo con el que se presenta el inicio de este proyecto es ineficiente para cualquier software de desarrollo. Este primer modelo de *Phiby* tiene un gran número de vértices, caras y puntos, generados durante su modelado. Además, el tamaño del este no es el adecuado ya que no cumple con las dimensiones en XYZ iguales a 1 metro, siendo su tamaño actual de 9 metros. Como regla se establece que el tamaño de los modelos debe ser de un metro en todos sus ejes. Por otro lado, el objeto posee dos mapas UV (uno referido a la cara y otro para el resto del modelo) y esto no lo soporta *Unity*. Debido a esto pueden surgir problemas en cuanto a las texturas que se muestran en Unity y generar errores adicionales, como p.ej. que no se vea bien la cara del personaje. Para que sea correctamente interpretado por *Unity*, *Phiby* debe estar formado únicamente por un mapa UV.

Es necesario corregir los problemas que presenta el personaje antes de animarlo. Para evitar cálculos innecesarios durante el renderizado y disminuir su tamaño en memoria, se deben simplificar la malla del personaje y reducir las caras y polígonos de su malla. De esta forma se mantiene la estructura del personaje y su malla, pero sin tanto nivel de detalle, logrando un menor número de procesos y simplificación en los cálculos. Este proceso se debe realizar en *Blender* y después exportar el modelo corregido de nuevo a *Unity*.

Se comienza eliminando todos los vértices que hayan quedado duplicados dentro de la malla. La forma útil de llevarlo a cabo es la llamada *Remove Double Vertex* en *Blender*. Gracias a esta técnica, se eliminan todos los vértices que hayan quedado autoduplicados, los cuales han sido generados para aportar una mayor calidad al modelo.

A continuación, se emplea un modificador (*Modifier* en *Blender*) denominado *Decimate Modifier* [17]. Se adjunta a continuación la figura 6,una captura comparativa con los resultados de aplicar el modificador sobre la malla.

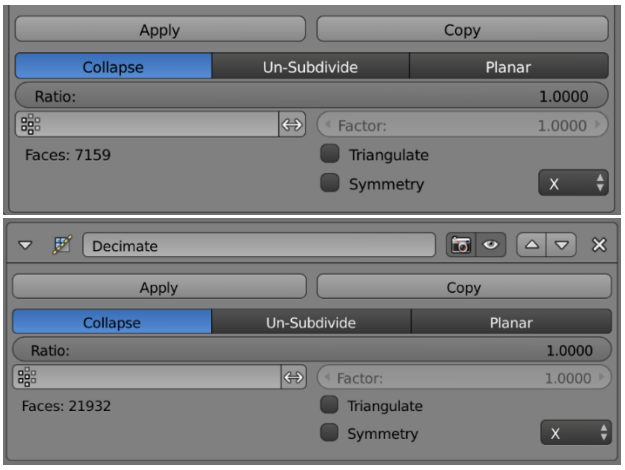

Figura 6. Parámetros del modificador Decimate antes y después de ser aplicado sobre la malla del personaje (Ratio y Factor)

La reducción del número de caras (*faces*) de la malla es muy elevado aplicando el procedimiento descrito previamente. Las 21932 caras iniciales quedan reducidas a un total de 7159. Gracias a este proceso se logra que el personaje no tenga posibles vértices duplicados, así como caras innecesarias que aumentan el trabajo que realiza el procesador durante la ejecución del videojuego. Esta reducción de caras no afecta a los movimientos que se produzcan durante las animaciones del personaje, que mantiene la forma de la malla correctamente sin deformarla. Por defecto, se aplican siempre los valores predeterminados de *Ratio* y *Factor* igual a uno en ambos casos. No se ha reducido más el número de caras ya que al pasar de las 7159 comienza a perderse mucho la forma de la malla del personaje y esto puede provocar comportamientos extraños durante el proceso de animación. Finalmente, se adjunta en la figura 7 el resultado final del modelo tras aplicar los procesos anteriores.

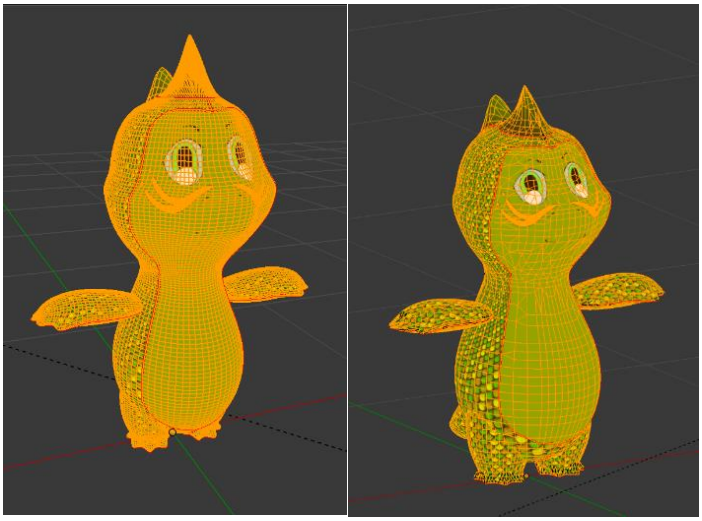

Figura 7. Representación del antes y después de la malla del personaje tras la reducción de caras y vértices

Como se observa en la figura 7, la reducción de las caras de la malla no afecta a la deformación del personaje, manteniendo así su estructura para dar paso al proceso de animación.

3.3 Animaciones desarrolladas

Para aportar sensación de realismo y mayor enriquecimiento a la estética del videojuego, se desarrollan tres animaciones diferentes para el personaje *Phiby*. Estas animaciones actúan en distintas situaciones dentro del videojuego, elevando su calidad visual. Estas tres animaciones, descritas en los siguientes subapartados del proyecto, son: animación en estado de reposo, animación mientras el personaje está andando y una última animación correspondiente a la acción de correr del personaje.

Para animar el modelo se recurre a *Blender*, entorno muy eficiente en cuanto al desarrollo de animaciones. En primer lugar, se crea un esqueleto que se asigna al modelo 3D del personaje, ya que inicialmente, este modelo carecía de esqueleto. Este es vital para realizar cualquier animación. Se adjunta a continuación en la figura 8 la asignación del esqueleto al modelo en *Blender*.

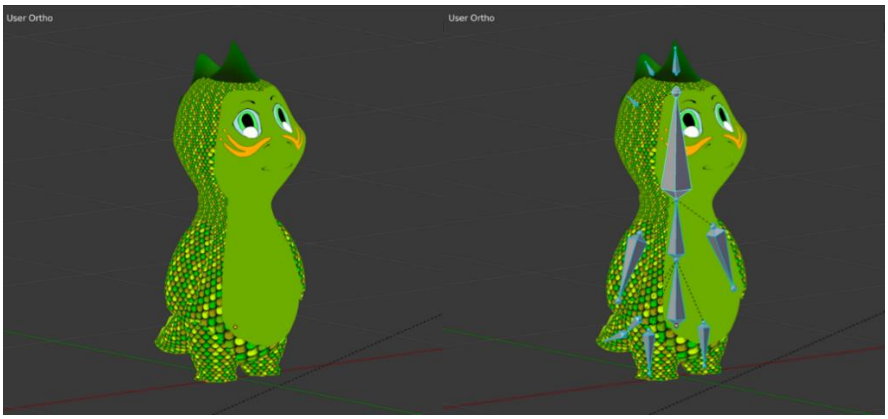

Figura 8. Modelo de Phiby sin y con esqueleto asignado

Como puede observarse de la figura 8, se han añadido distintos huesos (*Bones* [18] en *Blender*) en las extremidades del personaje (cabeza, brazos y piernas), partiendo de una columna principal subdividida. Se han añadido de la misma forma huesos de un tamaño menor a los cuernos de la cabeza y la cola del personaje, con el fin de generar posteriormente unas animaciones más completas y realistas con mayor sensación de dinamismo.

El proceso de animación en *Blender* [19] se basa en modificar, *frame* por *frame*, la disposición de cada uno de los huesos que constituyen el esqueleto asignado al modelo. De esta forma, al desplazar sus posiciones de forma mínima en cada frame, se logra mayor fluidez y resultados más realistas. Para las animaciones creadas para este modelo, se opta por modificar todos los *frames* hasta una posición intermedia, de posición máxima, para después copiar e invertir la posición de los *frames* ya modificados. Así se logra cerrar el ciclo

de la animación, obteniendo como resultado unas posiciones finales de huesos opuestas a las iniciales y logrando así simetría en el movimiento completo. A continuación, se muestra en la figura 9 una captura de *Blender* que corresponde con el proceso de animación.

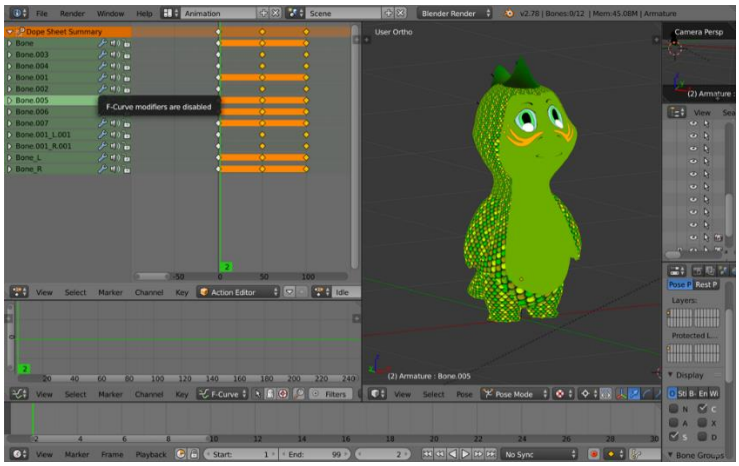

Figura 9. Proceso de animación en Blender (animación idle mostrada por defecto)

3.3.1 Animación reposo

Esta animación corresponde al estado de reposo del personaje cuando este se encuentra situado en la escena. Es muy importante en los videojuegos que los personajes dispongan de una animación por defecto en el estado de reposo o *idle*, ya que en numerosas situaciones el personaje va a permanecer quieto. La animación de reposo se constituye por un total de 100 *frames* que se repiten de forma cíclica para completar el estado de reposo del modelo.

Esta animación refleja un breve ladeo de la cabeza del personaje junto a un giro, también muy leve, de la cola. Los huesos de la columna también se han visto modificados, con el fin de que la caja torácica se expanda y se contraiga simulando la respiración del personaje. Se adjunta a continuación la figura 10 donde se muestra la animación reproducida en dos puntos diferentes.

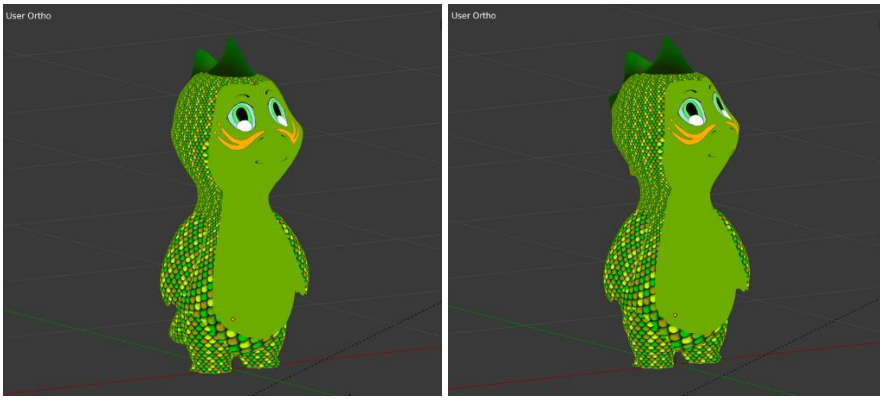

Figura 10. Izquierda: Primer frame de la animación con cabeza y cola del personaje en posición central, vista al frente. Derecha: Frame intermedio con cabeza y cola del personaje ladeadas

3.3.2 Animación de andar

Referencia a la acción de caminar por parte del personaje y se activa siempre que este comience a moverse en cualquier dirección. Se parte del esqueleto ya asignado previamente. En este caso, la animación de andar (*Walk animation*) se desarrolla en un total de 32 *frames*. Se aplica la misma técnica que en el apartado anterior para su generación.

Se colocan todos los huesos de cada frame adoptando la posición deseada. Con cada desplazamiento del hueso, se guardan los datos de su localización, rotación y escalado (*LocRotScale* [20] en *Blender*). Los primeros *frames* de la animación son utilizados para establecer el movimiento de las piernas y de los brazos. Una vez llegados a los *frames* intermedios (16-20) se copian todas las posiciones iniciales de los huesos (*frames* 1-15) y se pegan invertidos a continuación (*frames* 21-32). Estos *frames* intermedios, entre el *frame* 16 y el *frame* 20 sirven como punto medio de la animación, donde se produce el cambio de pierna para avanzar y el cambio de balanceo de los brazos. Se utilizan principalmente para pulir y ajustar el movimiento para que sea lo más preciso y natural posible. De esta forma, el último *frame* (32) coincide con el primero (1), completando el ciclo de la animación. Al reproducirlo de forma cíclica se obtiene como resultado una animación correcta sin cambios extraños en las extremidades del modelo.

El resultado de este proceso es un balanceo tanto de brazos como de piernas hacia delante y hacia atrás, junto a pequeños ladeos de la cola en función de cada paso efectuado por el modelo. A su vez, la cabeza y los cuernos ubicados en ella oscilan ligeramente. Se muestran en la figura 11 dos *frames* diferentes de la animación generada.

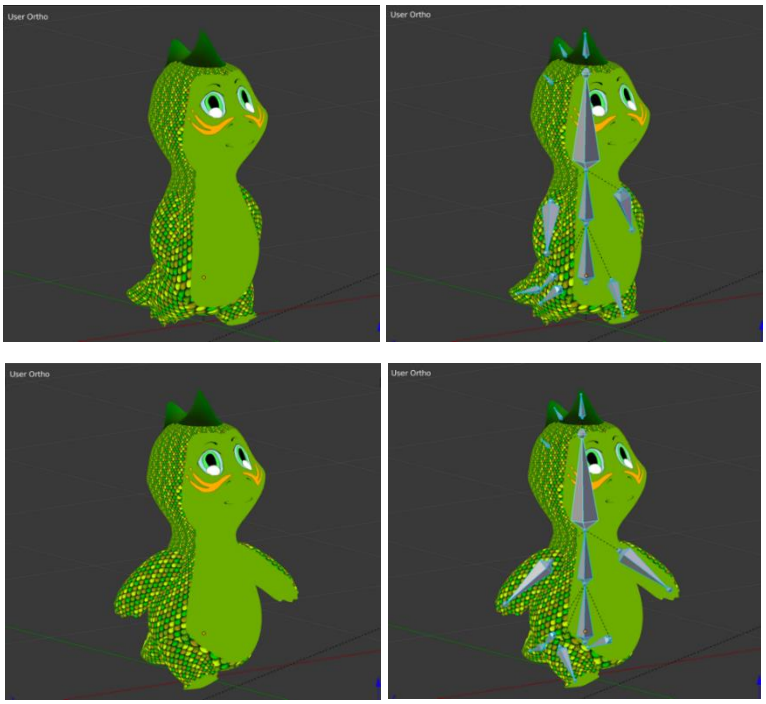

Figura 11. Posición inicial (parte superior) e intermedia (parte inferior) del modelo durante la reproducción de la animación

3.3.3 Animación de correr

A diferencia de la anterior, esta última animación de la que dispone el personaje provoca que las extremidades de este alcancen mayor amplitud en su desplazamiento y se alargue su balanceo. Las piernas y brazos se mueven a mayor velocidad, acompañados por un balanceo mayor de la cabeza y oscilaciones más rápidas de los cuernos dispuestos sobre ella.

A diferencia de la animación anterior, para esta han sido empleados únicamente 12 *frames*. El número de *frames* empleado es menor ya que la animación va a reproducirse a una velocidad mayor, y no es necesario que haya un elevado nivel de detalle en el balanceo de las extremidades. Siguiendo el proceso descrito previamente, se comienza por el primer *frame* donde se sitúan brazos y piernas en su rango de desplazamiento mayor. Esto corresponde a elevar los brazos casi a la altura de la cabeza, situando uno delante y otro detrás de la misma.

A partir de los *frames* intermedios, se pegan invertidos todos los *frames* iniciales de la animación para cerrar, de nuevo, el ciclo de reproducción. Se muestra en la figura 12 el resultado.

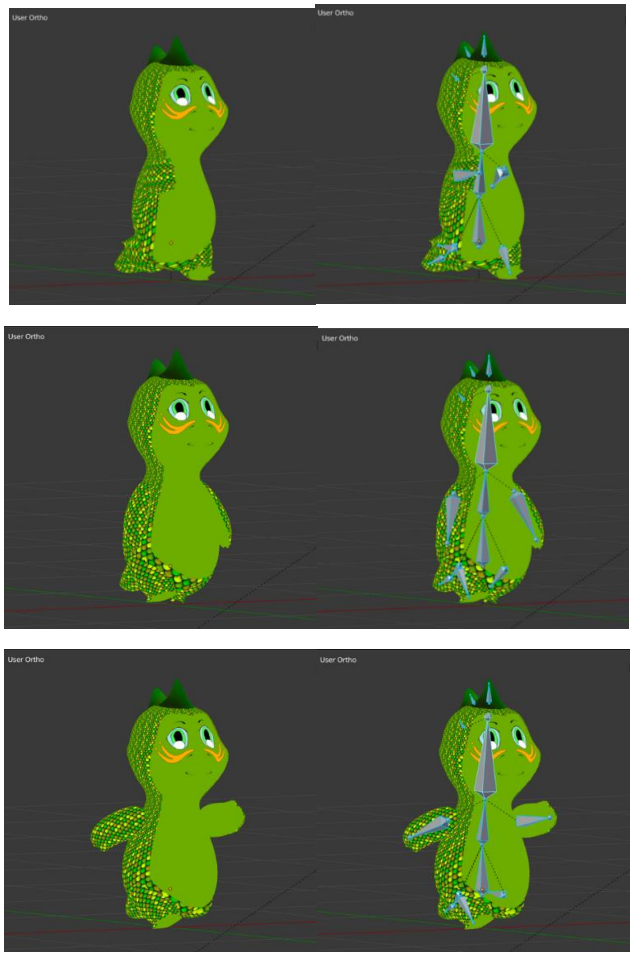

Figura 12. Ejecución de la animación en diferentes intervalos (frames) junto al esqueleto del modelo

3.4 Desarrollo de controles

Una vez creadas las animaciones que posee el personaje, es necesario realizar correctamente su exportación desde *Blender* para poder incorporarlo a *Unity*. Es de vital importancia que la exportación se efectúe de manera correcta, ya que pueden surgir problemas en *Unity* en caso contrario. Para ello, es necesario exportar el modelo del personaje *Phiby* como un tipo de archivo con extensión *.fbx* o *.blend* [21] desde *Blender*. A la hora de realizar la exportación, en necesario cumplir una serie de requisitos para evitar problemas futuros en *Unity*, como situar el personaje en el origen de coordenadas, tamaño de un metro en sus tres ejes y postura del tronco recta con los brazos en cruz.

De esta forma, cualquier modificación que se ejecute sobre el modelo desde *Blender* se actualiza al momento en *Unity*. P.ej., si cambia alguna textura del personaje en *Blender*, al guardar los cambios se actualiza en *Unity* cambiando su textura también. Así, *Unity* es capaz de importar: todos los nodos (posición, rotación y escala), puntos de pivote, meshes con vértices y polígonos juntos a los UVs, huesos, meshes asignadas a los huesos y las animaciones.

Dispuesto el modelo importado correctamente en el entorno de *Unity*, se procede a la generación de los controles tanto por teclado como por *Kinect*.

3.4.1 Control por teclado

Este proyecto está enfocado al desarrollo de un control para el personaje mediante la *Kinect* y a ejercicios en los que se emplea el cuerpo como medio de juego. Sin embargo, se completa con un control por teclado simple e intuitivo.

El control por teclado del personaje es muy similar al que se puede encontrar en el mercado de los videojuegos para ordenador. Se adjunta en la tabla 1 un resumen de estos controles junto a una breve explicación de la acción que provocan.

Tabla 1. Control del personaje mediante teclado

Tecla/Botón	Acción
W / ↑	Avanzar
S / ↓	Retroceder
A / ←	Girar a la izquierda
D / →	Girar a la derecha
SHIFT (mantener)	Correr
SPACE	Saltar

Para que el movimiento del personaje dentro de la escena sea óptimo, es necesario aplicar una serie de procedimientos para que pueda desplazarse e interactuar con el entorno. Como línea general, cualquier personaje dentro de un videojuego debe verse afectado por la gravedad, la forma del terreno y superficies, así como otros elementos físicos del escenario. Es decir, si hay una pared u objeto en la línea de desplazamiento que dibuja el personaje, este debería ser incapaz de poder atravesarla. Otro ejemplo de lo anterior queda reflejado con las diferencias de terreno dentro del juego. Si en la escena se encuentra una colina, el personaje al desplazarse sobre ella debe seguir la forma del terreno y ascender o descender según se distribuya.

Para que todos los requisitos anteriores se cumplan, existen una serie de componentes [22] que se pueden añadir al personaje dentro de *Unity*. Un componente permite dotar de nuevas propiedades o características a un objeto dentro del juego. La nomenclatura que *Unity* utiliza para referirse a cualquiera de estos objetos en el videojuego es *GameObjects* [23].

Una vez introducido el modelo de *Phiby* con sus respectivas animaciones (basta con arrastrar el fichero correspondiente "*PhibyCharacter.blend"* dentro del editor de *Unity*) se procede a agregar todos los componentes necesarios que garanticen un correcto control del personaje. A continuación, se adjunta en la figura 13 el conjunto global de estos componentes asociados.

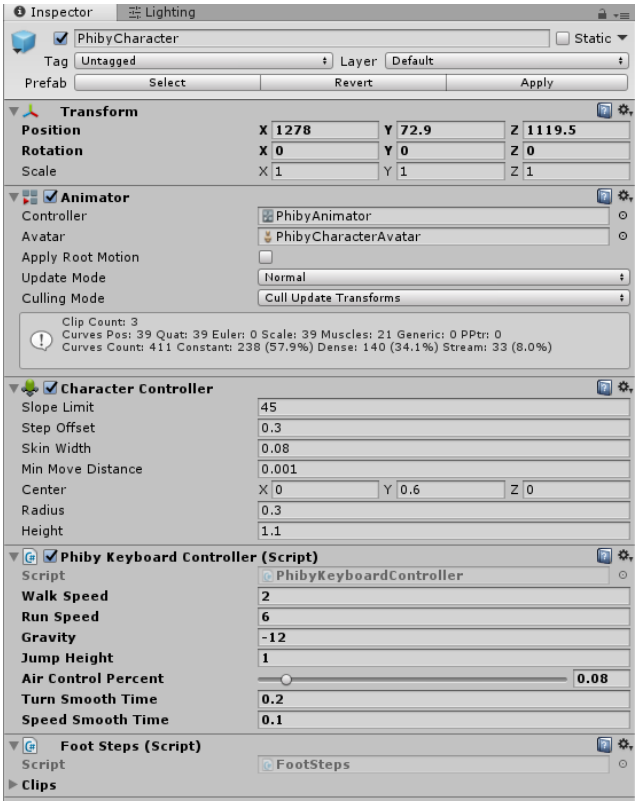

Figura 13. Pestaña Inspector de Unity de los componentes asociados al GameObject del personaje PhibyCharacter

En primer lugar, el componente *Transform* [24] determina la posición, rotación y escala del objeto situado en la escena. Es muy importante señalar que debe tener los valores de posición y rotación en sus tres ejes a cero, para situarlo en el centro de la escena sin rotaciones adicionales. Este proceso debe realizarse previamente en *Blender.* Además, la escala correcta para trabajar es igual a uno en sus tres ejes.

El componente *Animator* [25] se emplea para asignar una animación a un *GameObject* presente en la escena. Se hablará de él en detalle en el subapartado 3.5 Transición entre animaciones. Este se encarga de gestionar las animaciones del personaje y permitir la transición de una a otra.

El componente *Character Controller* [26] es utilizado para los controles del jugador en primera o en tercera persona. También se profundizará en este componente en el apartado 3.6 Físicas del personaje.

Finalmente, los últimos dos componentes son *Scripts* [27] de código (*"PhibyKeyboardController"* y *"FootSteps")* desarrollados en C#. El primero es el encargado de permitir el movimiento y control del personaje. El siguiente se trata de un *script* que permite lanzar efectos de sonido cada vez que *Phiby* camine o corra en la escena. Se profundizará más sobre este componente en el apartado "*3.4.4. Sonidos y efectos producidos al andar y correr"*. Se adjunta en la tabla 2 el resumen de contenidos de los parámetros que aparecen reflejados en el *script "PhibyKeyboardController"* junto a su significado.

Tabla 2. Parámetros de control del personaje definidos en el script "PhibyKeyboardController"

Propiedad	Función
Walk Speed	Velocidad a la que se desplaza el personaje caminando
Run Speed	Velocidad a la que se desplaza el personaje corriendo
Gravity	Gravedad que afecta al personaje
Jump Height	Altura máxima alcanzada con un salto
Air Control Percent	Control del personaje mientras está en el aire tras haber saltado
Turn Smooth Time	Velocidad de rotación del personaje sobre sí mismo
Speed Smooth Time	Velocidad de cambio de transición del personaje al alternar entre reposo, andar y caminar

Todos los parámetros que se muestran en la tabla 2 son de carácter público. Esto significa que, sea cual sea la necesidad (realización de pruebas, calibración, nuevos desarrollos), se pueden modificar sus valores directamente desde el editor.

3.4.2 Control y manejo de la cámara

El control de la cámara queda asignado al ratón. La función es indicar el punto hacia el que el personaje está mirando, obedeciendo los movimientos dibujados con el ratón y situando la cámara en el punto que se haya seleccionado. Al ser un videojuego de mundo abierto donde el jugador debe explorar todo el entorno, el disponer de un control sencillo y efectivo de la cámara es muy importante.

Unity ya cuenta con una cámara principal, denominada *Main Camera* [28]. Esta queda situada siempre en cualquier escena [29] (*Scene*) por defecto. Esta es la cámara que sigue a *Phiby*. Siguiendo la metodología del apartado anterior, al seleccionar la *Main Camera* desde el editor de *Unity* obtenemos el siguiente resultado, recogido en la figura 14.

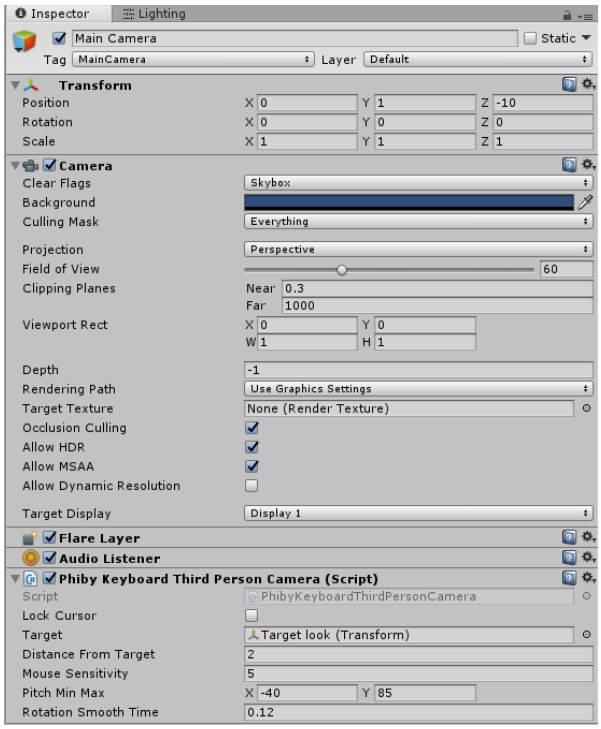

Figura 14. Pestaña Inspector asociada a Main Camera

El único componente que se va a describir es el *script* de control de la cámara *"PhibyKeyboardThirdPersonCamera"*. Permite girar la cámara alrededor del personaje para observar el entorno. El resto de componentes que se muestran en la figura 14 vienen preestablecidos con el *GameObject*. Los parámetros del *script* se describen en la tabla 3 a continuación.

Tabla 3. Parámetros de control de la cámara definidos en el script "PhibyKeyboardThirdPersonCamera"

Propiedad	Función
Lock Cursor	Permite ocultar el icono del cursor durante el juego
Target	Elemento/personaje que la cámara sigue por la escena
Distance Form Target	Distancia que separa la cámara del personaje
Mouse Sensitivity	Sensibilidad del ratón ante los movimientos
Pitch Min Máx	Ángulos mínimo y máximo de rotación de la cámara sobre el personaje
Rotation Smooth Time	Parámetro que configura la fluidez en el giro de la cámara al desplazar el ratón

3.4.3 Configuración del *prefab* y asignación de la cámara

La definición que *Unity* otorga al elemento *prefab* [30] es la siguiente: "*El prefab es un elemento que permite almacenar un objeto GameObject con todas sus componentes y propiedades. El*

prefab actúa como una plantilla a partir de la cual se pueden crear nuevas instancias del objeto que hay en la escena. Cualquier edición hecha a un prefab será inmediatamente reflejada en todas las instancias producidas por él, pero también se pueden anular componentes y ajustes para cada instancia individualmente".

En resumen, un *prefab* en *Unity* permite almacenar un *GameObject* ya configurado para después poder utilizarlo de manera ilimitada. En este proyecto se ha creado un *prefab* del personaje. Para el control por teclado, existe un *prefab* que contiene tanto el modelo y las animaciones como los componentes asignados. Esto simplifica enormemente la forma de trabajar.

De esta forma, si un miembro del equipo de trabajo quiere continuar con el desarrollo del proyecto o bien realizar pruebas, no tendrá más que arrastrar el *prefab* de *Phiby* controlado por teclado a la escena. Así se evita que cada vez que se quiera probar un nuevo desarrollo se tengan que añadir nuevos componentes o generar las animaciones desde cero. Es una vía óptima de simplificar las tareas y de reducir tiempo. Se adjunta a continuación en la figura 15 una captura del editor de *Unity* reflejando las propiedades del *prefab* de *Phiby*.

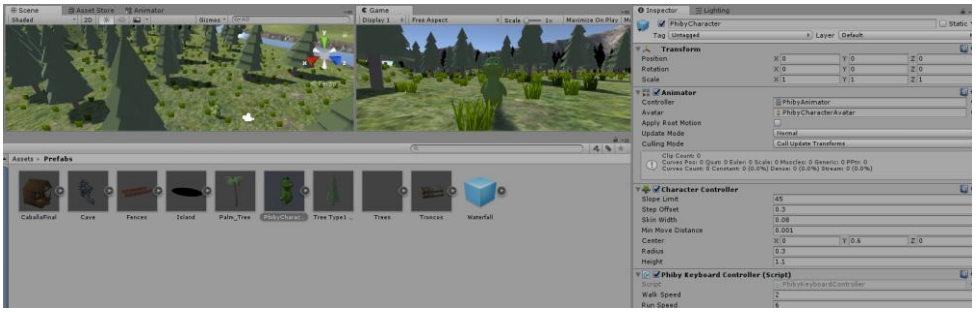

Figura 15. Captura del editor de Unity

Como se observa en la figura 15, al seleccionar el *prefab* de *Phiby,* se despliega en la zona de la derecha el *Inspector*. En él aparecen los mismos componentes y propiedades del *Phiby* que fue generado en un inicio. Al tratarse ahora de un *prefab*, puede ser utilizado en cualquier momento arrastrando únicamente el *GameObject* "*PhibyCharacter*" a la escena. Se describe a continuación el procedimiento para poder disponer de este *prefab* ("*PhibyCharacter.prefab"*) generado en cualquier proyecto que se quiera desarrollar en *Unity*.

1. Importar el *unityPackage* [31] denominado "*PV2.unitypackage*" al proyecto.
2. Dentro de la carpeta *Phiby*, se selecciona la subcarpeta *Prefab.*
3. Una vez abierta, se arrastra el prefab *PhibyCharacter* a la escena.
4. Dentro de la carpeta *Scripts,* se selecciona el script llamado "*PhibyThirdPersonCamera"* y se arrastra hasta la cámara principal (*Main Camera*) de la escena.
5. Finalmente, se expanden los desplegables del personaje (se adjunta una captura en la figura 16) y se escoge *target look* para arrastrarlo hasta el parámetro público "*target"* del *script* asociado previamente a la cámara.

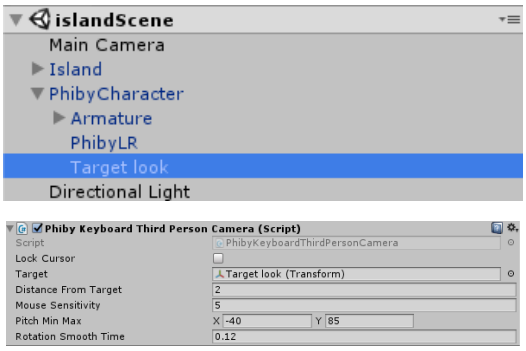

Figura 16. Parte superior: GameObject de Phiby desplegado. Parte inferior: Script asociado a la cámara junto al objeto target look asociado

3.4.4 Sonidos y efectos producidos al andar y correr

Para aportar más realismo al personaje, se le ha dotado de efectos de sonido asociados a cada paso que este efectúa. Así, cada vez que una de sus patas choque con el suelo va a producirse un efecto de sonido simulando pasos. Mediante el *script "FootSteps"* asociado al *prefab* de *Phiby* se pueden seleccionar los sonidos deseados que se van a reproducir. Se dispone de un array, a rellenar de forma personalizada, en el que se insertan todos los clips de audio que se desee.

Su diseño se basa en la reproducción de un sonido distinto cada vez que se efectúa una pisada, escogiéndose dicho efecto de sonido del *array* antes mencionado. De esta forma, si se dispone de un número considerable de *clips* de audio, se pueden producir pisadas con efectos diferentes, enriqueciéndose la experiencia de juego. Se adjunta a continuación en la figura 17 una captura correspondiente al *script "FootSteps"* donde únicamente se han asociado dos *clips* de audio. La selección aleatoria de los mismos se produce a nivel de código dentro del *script*.

Figura 17. Script "FootSteps" con clips de audio asignados

Una forma muy sencilla de poder enlazar el *script* anterior junto a la pisada de *Phiby* se basa en utilizar funciones de evento (*Event Functions*) [32] de *Unity*. De esta forma, se establece un evento cada vez que el pie de *Phiby* toque el suelo. En ese mismo instante, se produce una llamada a la función encargada de gestionar los *clips* de audio y se efectúa el sonido. Así, cada vez que cualquier pie de *Phiby* entre en contacto con la superficie, el evento se lanza llamando a la función deseada. A continuación, se adjunta la figura 18 donde se visualiza el proceso anterior. Esta sección se encuentra dentro de la ruta *Assets/Phiby* y seleccionando el *GameObject PhibyCharacter*.

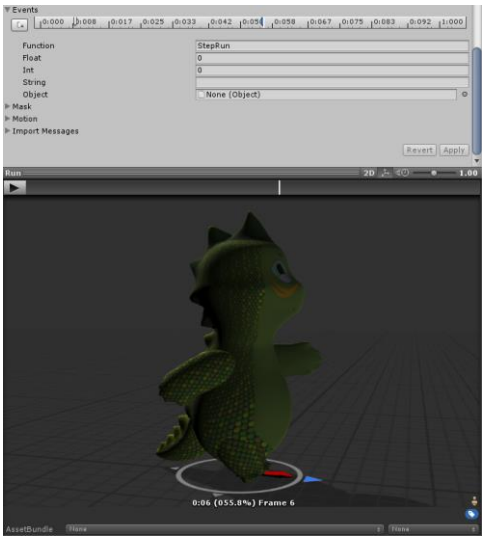

Figura 18.Evento (situado en la parte superior de la figura, en color azul) lanzado cada vez que el pie de Phiby choca con el suelo

3.4.5 Parámetros configurables: control por teclado

En la tabla 3 ubicada en el subapartado "*3.4.2. Control y manejo de la cámara"* se exponen todos los parámetros públicos que pueden ser directamente modificados desde el editor de *Unity*. Estos parámetros permiten variar el comportamiento de *Phiby* dentro de la escena. Los parámetros permiten, entre otras funciones, que camine más rápido o más lento, así como la altura que se puede alcanzar en cada salto.

Todos los parámetros son completamente personalizables. La finalidad de que estos se traten como parámetros públicos no es otra que la adaptación o personalización de cada jugador según su modo de juego o según sus preferencias. Al tratarse de los parámetros configurables asignados al control por teclado, su finalidad no es otra que la realización de pruebas dentro del entorno en desarrollo. Cobran mayor importancia los parámetros configurables que afectan al control mediante *Kinect*, como se ve en apartados posteriores en este proyecto.

3.4.6 Control mediante *Kinect* del personaje

La realización de este proyecto se basa en el desarrollo de un videojuego con fines terapéuticos que es controlado mediante la cámara de captura de movimientos *Kinect*. El medio de control de *Phiby* es únicamente el cuerpo del jugador. El jugador es capaz de controlar el desplazamiento del personaje y la cámara que tiene asociada mediante el tronco. Además, en la ejecución de los ejercicios se llevan a cabo movimientos específicos de las extremidades del jugador, como los brazos. De esta forma, se cubre un abanico de movimientos pensados para la apoyar las tareas de rehabilitación del jugador.

Es de vital importancia que se establezcan unos movimientos para su control que sean de sencilla ejecución y que no impliquen la adopción de posturas incómodas ni generen sobreesfuerzo. Se describe a continuación el procedimiento seguido para que el personaje *Phiby* sea capaz de copiar los movimientos del jugador.

En primer lugar, se asocia el *asset* desarrollado por César Luaces Vela en su trabajo de fin de grado [5] denominado "*KinectAsset"*. Este *asset*, de forma resumida, se encarga de copiar los movimientos del jugador ejecutados frente a la *Kinect* y realizar el envío de datos a *Unity*. Está compuesto por dos elementos: "*KinectReceiver"* y *"AmplifyManager".* El primero se trata de un esqueleto formado por las distintas articulaciones que pueden ser copiadas. El segundo es el encargado de gestionar la amplificación de cada movimiento, permitiendo completar el movimiento del personaje sin necesidad de que el jugador tenga que completar el mismo rango de movimiento. Se profundiza más en este último en apartados posteriores.

Para que el modelo *Phiby* copie los movimientos que ejecuta el jugador es necesario que se asigne correctamente el esqueleto del *KinectReceiver*. De forma resumida, consiste en asociar las extremidades y articulaciones del esqueleto con las de *Phiby*. Este procedimiento se describe con mayor detalle en el ANEXO II [5] del proyecto de fin de grado de César Luaces Vela. Se adjunta un resumen sobre como realizar el proceso en el ANEXO *I*.

El *KinectReceiver* es el encargado de copiar los movimientos del jugador capturados por la *Kinect* y traducirlos al modelo asignado, en este caso, *Phiby*. Por ejemplo, permite a *Phiby* levantar uno de sus brazos cuando el jugador ejecuta la misma acción. Los brazos y patas de *Phiby* son tratados en este proyecto como huesos completos. Es decir, no están subdivididos en muñeca codo y hombro, sino que se interpretan como una única extremidad del jugador. Se emplea el mismo esqueleto que utiliza Pablo López Miedes en su trabajo de fin de grado [16] y su *script* de control del personaje. Sin embargo, este *script* de control se amplía en el desarrollo de este proyecto para mejorar el control de *Phiby* y hacerlo más fiable y preciso.

El control de *Phiby* dentro de la escena funciona de la siguiente manera. Para avanzar, el jugador inclina ligeramente el tronco hacia delante. Para que se detenga, debe volver a la posición inicial. No se trata de un ángulo de inclinación forzado, ya que la posición que se adopte debe ser lo más natural posible y que no produzca fatiga o molestia. Si el jugador se inclina un poquito más fuerte hacia delante, *Phiby* pasa de caminar y a correr. El ángulo de inclinación para que el jugador debe adoptar para que *Phiby* camine es de 0.25 grados. Es un valor bajo para que no se fuerce al jugador y este no se fatigue. Para correr, debe inclinarse 4 grados. Así, se mantiene una postura natural sin forzar al jugador. Para que se detenga por completo, basta con que retroceda 5 grados inclinándose hacia atrás. Esto se traduce como volver a la posición inicial. Se adjunta la figura 19 a continuación que recoge esto ángulos de inclinación a adoptar.

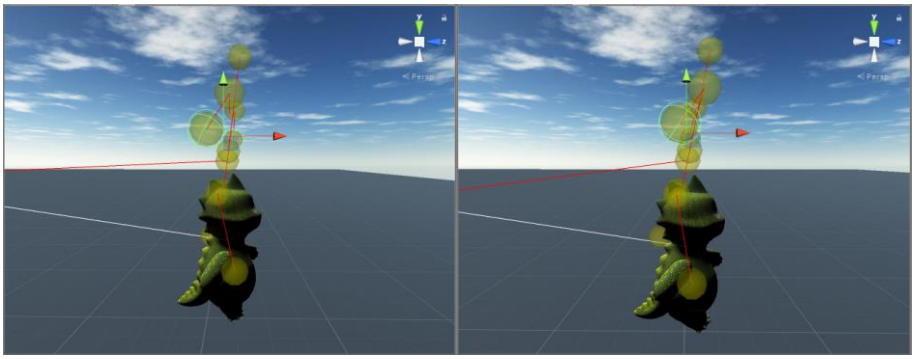

Figura 19. Ángulos de inclinación adoptados por el jugador para caminar/correr

Como se observa en la figura 19, el esqueleto no corresponde con las articulaciones del personaje como se describe previamente. Esto es debido a que únicamente se quieren detectar las inclinaciones del tronco para aplicarlas al personaje y que este avance, sin copiar el movimiento de ninguna extremidad. En el caso de los ejercicios sí es necesario que las articulaciones coincidan con gran exactitud, como se describe más adelante.

Una vez establecido el procedimiento para que *Phiby* avance dentro de la escena se completa el control del personaje mediante un sistema de giro sobre el propio eje. De esta forma, si para avanzar es necesario que el jugador se incline ligeramente hacia delante, para girar es necesario que se mueva el tronco en la dirección de giro deseada. Los desplazamientos y rotaciones se realizan sobre el propio eje del jugador. Así, si el jugador quiere que *Phiby* gire a la derecha, basta con que rote sobre sí mismo hacia esa dirección, copiando el personaje el mismo giro. Esta acción se puede llevar a cabo también mientras el personaje camina o corre. Los giros se desarrollan en función del movimiento de los hombros del jugador. Se adjunta a continuación la figura 20 donde se recogen ambos casos de giro.

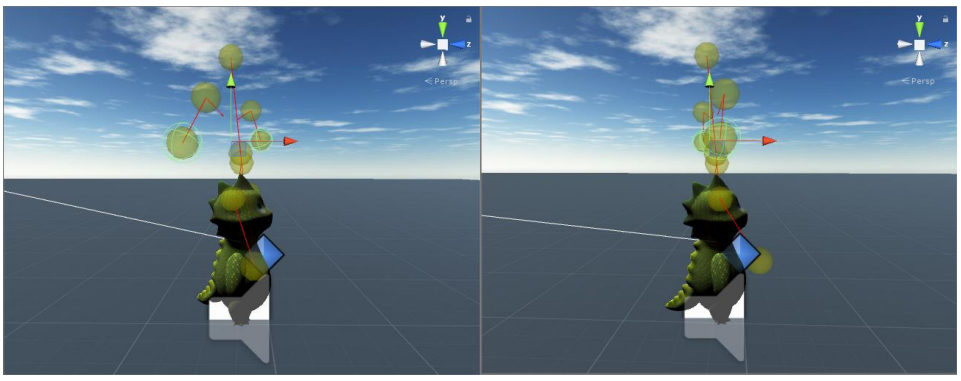

Figura 20. Ángulos de rotación del tronco sobre el eje Y para que el personaje gire

3.4.7 Control de la cámara mediante *Kinect*

Al no disponer del ratón para poder controlar la cámara, el control de esta se reduce únicamente al cuerpo del jugador. Para poder mirar alrededor, basta con que el jugador gire el tronco hacia cada lado, de la misma forma que se controlaba el giro de *Phiby* en el apartado anterior. Esto se debe a que la rotación de la cámara describe el mismo giro que se representa rotando el tronco del jugador.

De esta forma, la cámara gira a medida que *Phiby* gira. Es un control natural e intuitivo que no genera una fatiga adicional al jugador. Por lo tanto, si se desea que *Phiby* mire hacia la derecha, sólo es necesario rotar el tronco siguiendo esa misma dirección. A continuación, se adjuntan las figuras 21 y 22 donde se recoge la posición de la cámara con respecto al personaje y la vista desde el videojuego.

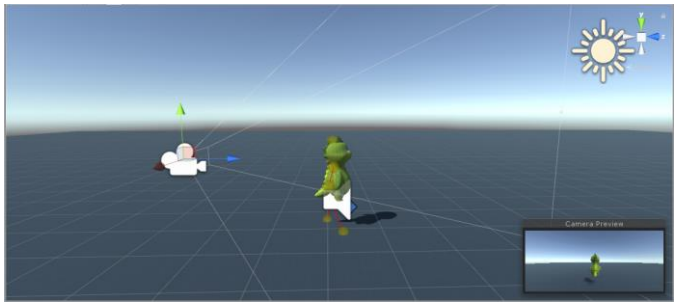

Figura 21. Localización del GameObject Main Camera con respecto a Phiby

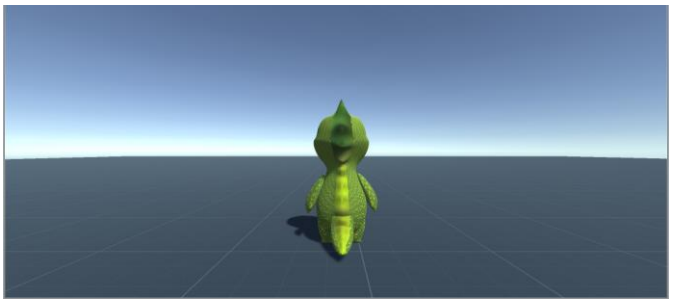

Figura 22. Vista del videojuego desde el GameObject Main Camera

3.4.8 Control, configuración y manejo del *prefab*

Dado que la configuración del *GameObject* que permite controlar a *Phiby* es más tediosa, se opta de nuevo por la generación de un *prefab* que pueda ser utilizado por cualquier persona ajena a su desarrollo. Por lo tanto, siguiendo la misma estructura que la mostrada en el apartado 3.4.3. Configuración del *prefab* y asignación de la cámara, se listan los pasos a continuación para su configuración.

1. Importar el paquete "*PhibyV3.unitypackage*" al proyecto en *Unity*.
2. Dentro de la ruta *Assets/Phiby/prefab* se encuentra el *prefab* controlado por *Kinect* de *Phiby*. Está constituido por el *GameObject* del personaje junto al *KinectReceiver*. Una vez seleccionado se arrastra a la escena para continuar con su configuración.
3. Para la configuración de la cámara basta con asociar el *script* "*CamControl"* ubicado en la ruta *Assets/Scripts/Camera* al *GameObject Main Camera* de la escena.
4. Al parámetro "*target"* del *script* se le asocia el *GameObject* "*target look"* ubicado dentro del *GameObject "PhibyCharacter".* Se expone el resultado de este apartado en la figura 23. El *GameObject* "*target look"* se encuentra ubicado en la figura 24.
5. El *script* "*Controller"* se encarga de gestionar el control de *Phiby*. Se encuentra en la ruta *Phiby/KinectAsset/KinectReceiver/SkeletonReceiver/Armature/Ground/HipCenterUp*. El motivo de que este *script* esté asociado a "*HipCenterUp"* es porque corresponde con toda la zona superior del tronco. De esta forma es como se controla el giro mediante los hombros y el avance con la inclinación del tronco del jugador. Se muestra en la figura 25 el contenido del *script*.

6. Finalmente, para que la cámara siga al personaje mientras se desplaza, se configuran los parámetros "*Mirar Camara"* y "*Armature"* dentro del script "*Controller"*. Al primero, se le asocia el *GameObject* "*target look"* contenido dentro de *PhibyCharacter.* Al segundo, se le asocia el *GameObject* "*Armature"*, también contenido dentro del *GameObject PhibyCharacter*.

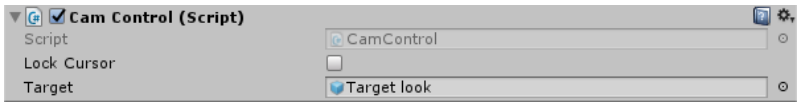

Figura 23. Asociación del parámetro target dentro del script de control de la cámara

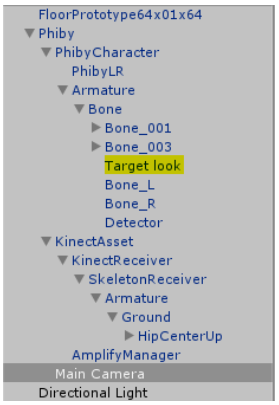

Figura 24. Ubicación del GameObject "target look"

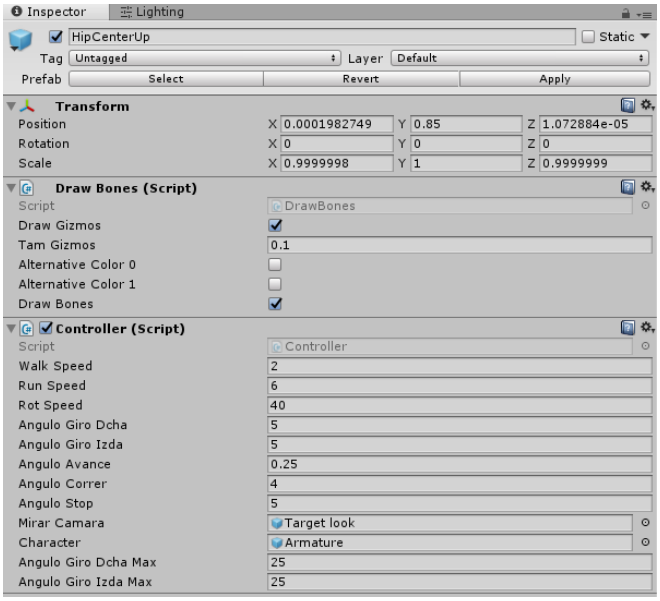

Figura 25. Script "Controller" asociado al GameObject "HipCenterUp"

El *GameObject* "*KinectAsset"* se descompone como se muestra en la figura 24. Es el esqueleto que copia los movimientos del jugador y son asignados al modelo que se haya asociado. El hecho de asociar al *GameObject* "*HipCenterUp"* el *script* de control de *Phiby* radica en que hace referencia a todo el conjunto de huesos del tronco superior. Es decir, todo el control del personaje se gestiona mediante el tronco superior (hombros, inclinación del tronco). Por lo tanto, no tiene sentido que el control del personaje se asigne al conjunto de huesos que

pertenecen a las extremidades inferiores (“*HipCenterLeft”, “HipCenterRight”)* ya que ningún movimiento o ejercicio se controla mediante ellas.

3.4.9 Sonidos y efectos producidos al andar y correr

De forma análoga al apartado “*3.4.4. Sonidos y efectos producidos al andar y correr”*, se sigue el mismo procedimiento para generar efectos de sonido cada vez que *Phiby* camine o corra. En este caso, el *prefab* dispone de un *array* de 18 *clips* de audio. De esta forma, al andar y correr se obtiene un conjunto de sonidos de pisadas completamente diferentes entre sí. El *script* “*FootSteps”* se encarga de seleccionar de manera aleatoria un *clip* para cada pisada. Al disponer de 18 *clips* de audio, la aleatoriedad en la selección de un *clip* es mayor en comparación con *array* de 2 *clips* de audio. Se adjunta la figura 26 a continuación donde se recogen todos los *clips* de audio empleados.

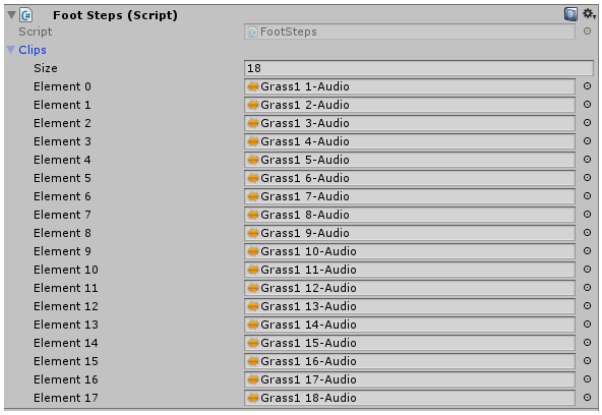

Figura 26. Clips de audio asociados al script "FootSteps"

Como puede observarse en la figura 26, todos los *clips* de audio llevan el mismo prefijo “*Grass”* en su nombre. Esto quiere decir que el sonido que produzca cada paso al caminar simulará el efecto de caminar sobre hierba. Se establecen por defecto ya que el escenario donde tiene lugar la acción del videojuego está cubierto principalmente por hierba.

3.4.10 Parámetros configurables: control mediante *Kinect*

Como se muestra en el apartado “*3.4.6. Parámetros configurables: Control por teclado*”, existen una serie de parámetros que pueden ser modificados según distintas necesidades. Dado que este proyecto se basa en la utilización de la cámara *Kinect* como medio de control de juego y ejercicio del jugador, cobran mayor interés todos los parámetros que puedan ser alterados en este campo.

Todos estos parámetros se encuentran ubicados en el *script* “*Controller”*, encargado del control de *Phiby*. Se muestra a continuación la figura 27 donde se recogen. Se encuentra ubicado en la pestaña *Inspector* de *Unity* cuando se selecciona el *script*.

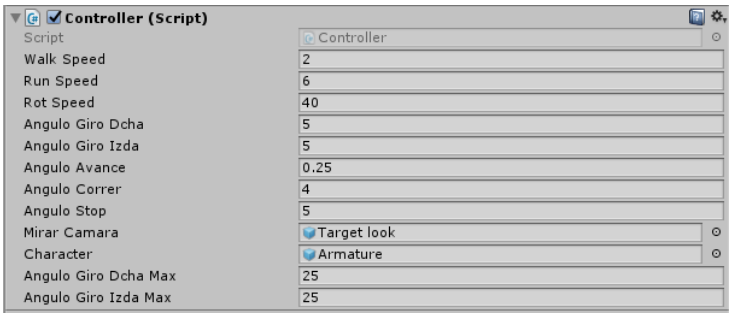

Figura 27. Parámetros configurables disponibles en el script "Controller" asociado a Phiby

El hecho de que los parámetros de control sean de carácter público se basa en la búsqueda de los valores adecuados de los mismos para cada jugador. Al tratarse de un videojuego enfocado a un público heterogéneo en cuanto a capacidades motoras, cobra mucha importancia que cada jugador tenga unos controles específicos y adecuados a sus capacidades. Con ello se pretende evitar, a toda costa, que el jugador alcance posturas que le generen incomodidad o que le produzcan fatiga. Es muy importante señalar que el control de *Phiby*, así como los ejercicios que se llevan a cabo, pueden ser controlados con éxito tanto de pie como en silla de ruedas.

Lograr alcanzar los valores que permitan a cada jugador divertirse a la vez que se ejercitan y completan las tareas de rehabilitación es vital, ya que se puede generar frustración en caso de que el personaje no responda bien a los movimientos dibujados por el jugador. Como se comenta en el apartado "*2.2 Plataforma Blexer-med*", se puede establecer una comunicación con la web. A través de esta web, el terapeuta puede acceder al control de todos estos parámetros y configurarlos según el estado y evolución de cada uno de los jugadores. De esta forma, en función de los avances de cada jugador, el terapeuta puede, por ejemplo, incrementar el número de troncos que se pueden talar o bien disminuir el ángulo de inclinación que debe alcanzar el jugador para caminar. Se recogen todos los parámetros de control de *Phiby* en la tabla 4.

Tabla 4. Descripción de los parámetros públicos contenidos en el script "Controller"

Propiedad	Función
Walk Speed	Velocidad a la que anda el personaje
Run Speed	Velocidad a la que corre el personaje
Rot Speed	Velocidad de rotación/giro de la cámara
Ángulo Giro Dcha.	Ángulo de rotación necesario del tronco para que el personaje gire en ese ángulo
Ángulo Giro Izda.	Ángulo de rotación necesario del tronco para que el personaje gire en ese ángulo
Ángulo avance	Ángulo que tiene que inclinarse el jugador hacia delante para que el personaje avance
Ángulo Correr	Ángulo que tiene que inclinarse el jugador hacia delante para que el personaje corra
Ángulo Stop	Ángulo que debe retroceder el jugador para que el personaje se detenga
Ángulo Giro Dcha. Máx	Ángulo de giro que una vez sobrepasado por el jugador provocará que el personaje gire más rápido
Ángulo Giro Izda. Máx	Ángulo de giro que una vez sobrepasado por el jugador provocará que el personaje gire más rápido

3.5 Transición entre animaciones

Para que el cambio entre animaciones se produzca de forma suave y exponencial se utiliza la técnica del árbol de mezclas o *Blend Tree* [33] de *Unity* desde la pestaña *Animator* [34]. El *Animator* se encarga de juntar todas las animaciones disponibles del personaje y poder establecer control sobre las mismas en función de distintas circunstancias. Con esto se logra fusionar con éxito las animaciones de reposo, andar y correr del personaje.

La definición de *Blend Tree* según el manual de *Unity* es la siguiente: *"Los blend trees son utilizados para permitir a múltiples animaciones ser mezcladas de manera suave. La cantidad que cada movimiento aporta a la animación final es controlada utilizando un parámetro de mezcla o bleeding parameter, que es sólo uno de los animation parameters asociados con el Animation Controller. Con el fin de que el resultado mezclado tenga sentido y sea natural, los movimientos mezclados deben ser de naturaleza y tiempo similares"*.

Se toma como ejemplo el cambio de la animación de andar a correr. Para que el cambio de una a otra se produzca de forma natural se debe controlar mediante un parámetro. Además, los movimientos entre animaciones deben dar lugar a movimientos que sean muy parecidos, ya que, de lo contrario, los resultados pueden no ser los adecuados. Se adjunta en la figura 28 el *blend tree* que *Phiby* tiene asociado, así como la pestaña *Animator*. Para llegar a la misma basta con seleccionar a *Phiby* dentro de la escena, comprobar la ventana del *Inspector* y abrir la ventana *Animator*.

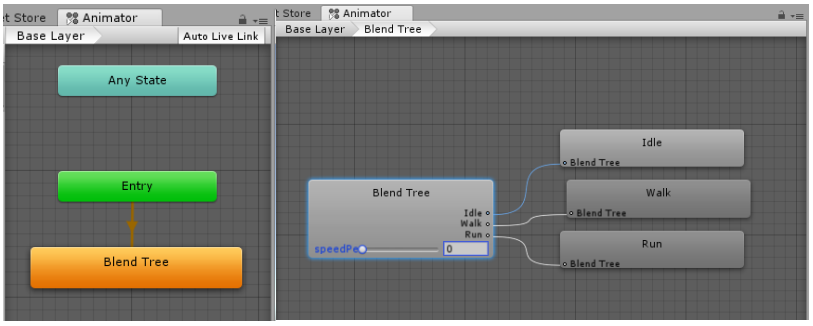

Figura 28. Izquierda: Jerarquía del Animator asociado a Phiby. Derecha: Desglose del blend tree junto a las animaciones de las que dispone el personaje

En la figura 28 aparece el parámetro encargado de controlar la mezcla de las animaciones. Se trata del parámetro "*speedPercent*", cuyo valor inicial es igual a cero. Este se encuentra dentro del contenedor del *Blend Tree*. Este parámetro se encarga de gestionar la mezcla y transición entre las animaciones. Su valor oscila entre cero y uno, siendo ambos valores el mínimo y máximo, respectivamente. Dependiendo de su valor, se reproduce una u otra animación del personaje. Se controla mediante un parámetro público con el valor de "*speedPercent*" dentro del *script* de control de *Phiby "Controller"*.

Las tres animaciones que posee el personaje aparecen ubicadas en la zona derecha de la figura 28. Se denominan "*Idle", "Walk"* y *"Run"*. Todas ellas se enlazan con el *Blend Tree* ya que se encarga de alternar entre ellas dependiendo del valor de "*speedPercent*". La ventana del *Inspector* se recoge en la figura 29.

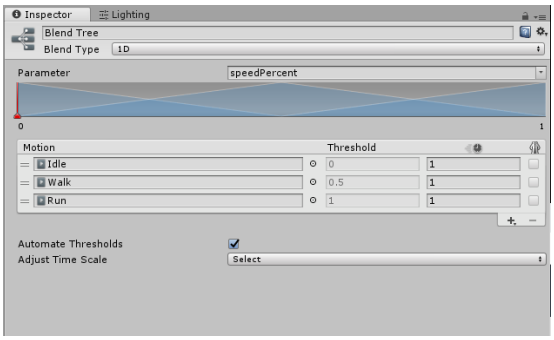

Figura 29. Ventana Inspector asociada al Blend Tree

Se aporta a continuación en la figura 30 una configuración del *Blend Tree*, junto a un valor aleatorio del parámetro "*speedPercent*" para ilustrar lo descrito previamente.

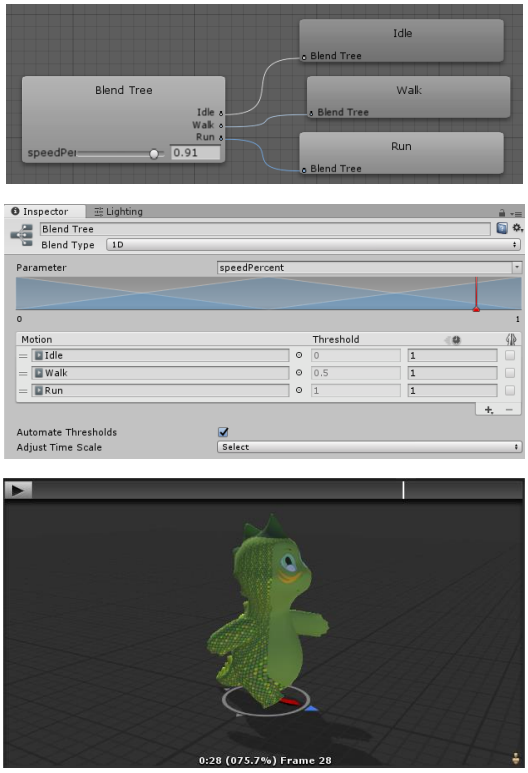

Figura 30. Fila superior: Blend tree con valor 0.91 en "speedPercent". Parte central: Ventana del Inspector asociada. Parte inferior: Representación de la animación ejecutada para ese valor

En la figura 30 se aporta un valor de "*speedPercent*" igual a 0.91. Este valor significa que se van a reproducir dos animaciones, como se muestra en la parte central de la figura 30. Cada uno de los picos mostrados en azul representa el valor máximo de la animación que se reproduce. Esto quiere decir que, si el indicador rojo se sitúa en uno de esos tres picos, se está reproduciendo única y exclusivamente una sola animación. El primer pico, junto al descenso de este, corresponde con la animación de reposo. Los dos siguientes corresponden a las animaciones restantes.

Como en la figura 30 el indicador rojo se encuentra a punto de alcanzar el pico máximo de la animación de correr, esto quiere decir que se está reproduciendo casi por completo esta animación, aunque queden partes mínimas de la animación de andar. Así se fusionan las animaciones durante esa transición sin producir efectos extraños en el personaje. Se adjunta a continuación la tabla 5. Esta recoge las animaciones que se reproducen en función del valor que adopte el parámetro "*speedPercent*".

Tabla 5. Animaciones reproducidas en función del valor del parámetro "speedPercent"

Valor de "*speedPercent*"	Animación reproducida
0	Reposo
Entre 0 y 0.5	Reposo/Andar
0.5	Andar
Entre 0.5 y 1	Andar/Correr
1	Correr

El valor de "*speedPercent*" depende del *script* "*Controller",* ya que es el encargado de gestionar el movimiento del personaje. Debido a esto, su valor varía dependiendo del tipo de control empleado para *Phiby*. Para el caso del control por teclado, su valor depende de si se pulsa la tecla asociada a andar o la pulsada a correr. Sin embargo, como *Phiby* se controla principalmente mediante la *Kinect*, depende del ángulo de inclinación del jugador.

3.6 Físicas del personaje

Como en la gran mayoría de videojuegos, existe una serie de físicas que afectan tanto al comportamiento del personaje como al entorno en el que se encuentra. En primer lugar, la gravedad actúa como factor predominante. Esto va a evitar que *Phiby* levite o que otros elementos del juego floten en la escena. Además, si *Phiby* salta lo correcto es que finalizado el salto vuelva a su posición de origen. De la misma forma, si al desplazarse se topa con otro objeto dentro de la escena, en normal que este se detenga.

Como en este proyecto se desarrollan dos tipos de controles para el personaje (control por teclado y control por *Kinect*) la forma en que la física actúa sobre cada uno de ellos es diferente. Su explicación queda cubierta en los dos siguientes subapartados, donde, además, se exponen las diferencias entre ambos y el procedimiento que se sigue para su correcto control en el videojuego.

3.6.1 Físicas aplicadas al personaje (control por teclado)

En el apartado "*3.4.1. Control por teclado*" se comenta que para el control del personaje se emplea un componente denominado "*Character Controller*". Este componente se utiliza principalmente para definir los controles de un personaje tanto en primera como en tercera persona en *Unity*. Se puede añadir a cualquier *GameObject* de la escena para que pueda interpretar controles. Se adjunta en la figura 31 el *Character Controller* asociado a *Phiby*. Todas las propiedades que este componente incorpora se adjuntan a su vez en la tabla 6.

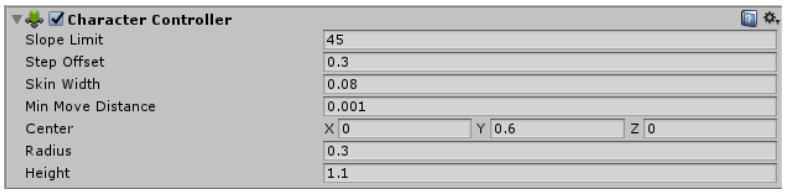

Figura 31. Character Controller ubicado en la ventana Inspector que está asociado a Phiby

Tabla 6. Propiedades asociadas al componente Character Controller asignadas a Phiby

Propiedad	Función
Slope Limit	Limita el *Collider* para que solo suba pendientes que sean menos empinadas que el valor asignado en grados
Step Offset	Establece la altura del escalón que puede subir el personaje cuando se encuentre cerca de ella
Skin Width	Permite a dos *colliders* introducirse el uno en el otro tanto como el valor de este parámetro. Se recomienda que su valor sea el 10% del radio (*Radius*)
Min Move Distance	Se utiliza para reducir el Jitter. Si el personaje intenta moverse por debajo del valor indicado, este no se moverá.
Center	Sitúa el *Capsule Collider* en la escena
Radius	Establece el radio del *Capsule Collider*
Height	Configura la altura del *Capsule Collider*

Este componente incorpora un elemento denominado "*Capsule Collider*" [33]. Se trata de una figura compuesta por dos semiesferas unidas por un cilindro, que deben contener dentro al personaje que va a ser objeto de control en el videojuego. Permite interpretar los choques con otros elementos (*GameObjects*) y su posición en la escena. La forma de este componente debe adaptarse con la mayor precisión posible al personaje, ya que de esta forma aumenta el realismo de los movimientos en el mundo y las interacciones en la escena.

Dado que este componente permite interpretar la posición de *Phiby* en la escena (saltando o en contacto con el suelo) no hace uso de otro componente denominado "*RigidBody*" [34]. De forma resumida, este componente permite a los *GameObjects* actuar bajo el control de la física. En este caso no es necesario utilizarlo ya que la posición del personaje y sus físicas se controlan mediante el *Character Controller*. Se profundiza más en este componente en el siguiente apartado.

Desde el *script* de control del personaje por teclado "*PhibyKeyboardController*" se accede fácilmente al *Character Controller* para gestionar el control de *Phiby*. Se adjunta a continuación en la figura 32 un fragmento de código donde se recoge su captura y en la figura 33 el *Character Controller* que tiene asociado *Phiby*.

```
Animator animator;
Transform cameraT;
CharacterController controller; // Character Controller encargado de controlar el movimiento del personaje

void Start () {
    animator = GetComponent<Animator> ();
    cameraT = Camera.main.transform;
    controller = GetComponent<CharacterController> (); // Obtención del componente asignado
}
```

Figura 32. Obtención del componente Character Controller en el script "PhibyKeyboardController"

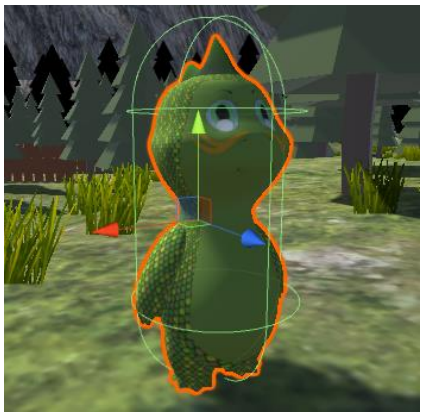

Figura 33. Character Controller asociado a Phiby

3.6.2 Físicas aplicadas al personaje (control por *Kinect*)

En este apartado se trata la explicación de los procesos seguidos para lograr que *Phiby* se desplace correctamente en la escena respetando el contorno de las superficies y viéndose afectado por otros elementos físicos. En el subapartado anterior el personaje se controla mediante el componente *Character Controller*, ya que su manejo queda reducido al teclado y al ratón. En este caso, el control del personaje queda derivado exclusivamente al movimiento del cuerpo del jugador que es capturado por la *Kinect*. Por lo tanto, es necesario que *Phiby* quede afectado por la gravedad e interactúe con otros elementos del juego (*GameObjects*).

Para el control por teclado y ratón, el componente *Character Controller* incorpora por defecto el componente *Capsule Collider*, encargado de gestionar las interacciones entre el objeto que contiene y el entorno. Como *Phiby* ahora se controla mediante movimientos corporales, es necesario añadir el *Capsule Collider*. En primer lugar, se asigna al elemento "*Armature"* de *Phiby* este componente. El motivo por el cual se asigna a *Armature* es debido a que este *GameObject* contiene todos los huesos y elementos del modelo. De esta forma se controla que cualquier interacción entre el personaje y el entorno quede recogida en este componente. Se adjunta a continuación la figura 34.

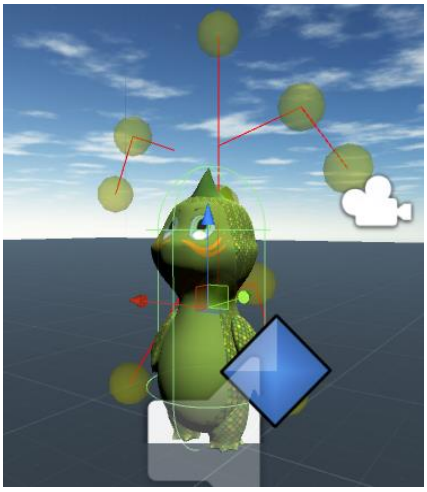

Figura 34. Capsule Collider asignado a Phiby

Como se puede observar en la figura 34, además del *Capsule Collider* aparece mostrado el esqueleto de la *Kinect* (conjunto de esferas de colores unidas por el segmento rojo). Este representa el cuerpo del jugador. No es necesario que coincida con el modelo, ya que en este caso no se copian movimientos de articulaciones si no que se utilizan los diferentes ángulos y giros del tronco para permitir el movimiento del personaje.

En segundo lugar, es necesario añadir un componente que permita al personaje verse afectado por las leyes de la física. Este componente es el *RigidBody*, mencionado previamente. Es necesario para que *Phiby* obedezca las leyes de la física dentro del mundo. Se adjunta a continuación la figura 35 que recoge ambos componentes, asociados al modelo y mostrados en la pestaña del *Inspector* en *Unity*.

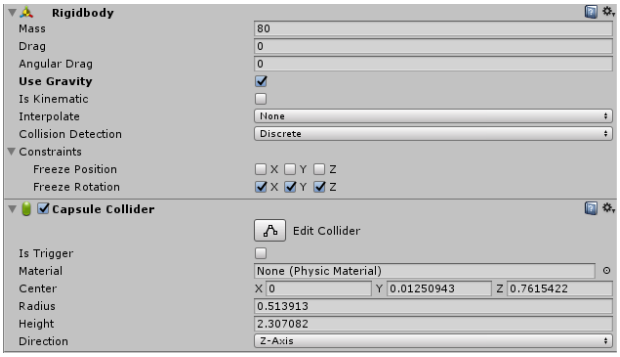

Figura 35. Componentes "Rigidbody" y "Capsule Collider" asociados a Phiby

Los componentes mostrados en la figura 35 se añaden al personaje para permitir que este se desplace dentro de la escena, que se vea afectado por la gravedad dentro del videojuego, que sea capaz de responder ante las variaciones del terreno y que se vea afectado por los choques con otros elementos en su camino. Para que *Phiby* quede afectado por la variación del suelo, el componente *Rigidbody* juega un papel fundamental.

Como se aprecia en la figura 35, está activado "*Use Gravity*". Esto, junto a un valor de masa inicial ("*Mass")* en este caso de 80, permite que *Phiby* por su propio peso descienda laderas o las pueda subir. Si no se activa el parámetro "*Use Gravity", Phiby* se desplaza siguiendo la misma trayectoria en el eje X, atravesando entornos y sin verse afectado por pendientes o descensos. Dentro del mismo componente, se despliega el parámetro "*Constraints*". Este se encarga de indicar sobre qué ejes (posición y rotación) el movimiento del personaje puede quedar afectado.

Los ejes que se muestran en "*Freeze Position"* están desactivados. Esto se debe principalmente a que si se activa uno de ellos, significa que el personaje congela su desplazamiento en esa dirección y se mantiene fija. A modo de ejemplo, si se activa el eje Y, *Phiby* no obedece a los cambios de pendiente dentro de la escena. Por otro lado, los ejes ubicados en "*Freeze Rotation"* están activados, evitando así que el personaje rote en cualquier dirección. Esto se debe a que si ninguno se activa, *Phiby* comienza a rotar como si se aplicaran fuerzas externas sobre el personaje. Una vez activados, *Phiby* queda configurado de forma que las leyes de la física le afectan de la manera prevista.

4. Ejercicios

El desarrollo de este proyecto se basa tanto en el control de *Phiby* mediante movimientos corporales como en la realización de ejercicios terapéuticos que copien el movimiento de las extremidades del jugador. Cada uno de estos ejercicios está integrado dentro de la historia del videojuego, convirtiéndose así en un mismo entorno donde el jugador no contemple cada ejercicio como una actividad independiente sino que todo forme parte de la misma experiencia. Todos ellos tienen como finalidad que se realicen una serie de movimientos muy específicos y que a su vez permitan tareas de rehabilitación mientras el jugador se divierte.

La versión inicial de estos ejercicios se descompone en cuatro minijuegos diferentes. Estos son: trepar, remar, matar topos y volar en ala delta. Su mejora se incorpora en "*Phiby V1"* donde evolucionan hasta convertirse en trepar, remar, cortar troncos, bucear y remar. En la versión actual "*Phiby's Adventures V2*" desarrollada en este proyecto se integran en el mismo escenario de mundo abierto y se convierten en tareas de trepar árboles, cortar troncos de la isla, así como volar en ala delta o bien hacer esquí sobre la montaña principal de la isla.

Todas estas escenas han sido ya implementadas en la versión Demo realizada por César Luaces Vela en su proyecto [5], salvo el ejercicio de esquiar. Por lo tanto, se adaptan las escenas ya realizadas previamente a esta nueva versión, manteniendo sus movimientos iniciales. Estos ejercicios mantienen la misma actividad que tiene que ejecutar el jugador, moviendo los brazos o mantener el equilibrio con el tronco. Además, los mismos movimientos se aplicarán a diferentes situaciones dentro del juego, como escalar una ladera, donde el jugador continúa con el movimiento de brazos para su rehabilitación sin caer en la monotonía de ejercitarse siempre realizando el mismo ejercicio.

4.1 Amplificación

Este proyecto se enfoca hacia jugadores que no tienen las mismas capacidades motoras entre ellos. Algunos pueden realizar el movimiento en un rango menor que otros, o bien les cueste mayor esfuerzo llegar a completar el mismo para ejecutar una acción dentro del juego. Debido a esto se hace necesaria la amplificación de los movimientos efectuados. El prefab "*AmplifyManager"* [5] contenido dentro del *asset* "*KinectAsset"* se encarga de esta tarea.

Esta amplificación consiste fundamentalmente en aplicar una rotación amplificada a la misma articulación que tenga asociada el personaje, en este caso, *Phiby.* De esta forma, aunque el jugador eleve ligeramente el brazo para iniciar una acción, la amplificación se encarga de completar el movimiento que se aplica sobre el personaje, finalizando su recorrido y ejecución en el juego. El hecho de emplear la amplificación en los ejercicios permite que cualquier jugador, sean cuales sean sus limitaciones físicas, pueda llevarlo a cabo y no sentirse frustrado porque no pueda completar el movimiento. De esta forma, todos los jugadores son tratados de la misma forma y permite que sigan divirtiéndose, realizando los movimientos que sean capaces de completar. Por ejemplo, si un jugador en concreto es incapaz de levantar el brazo más de 45 grados para talar un tronco, la amplificación se encarga de tomar este último valor como referencia para el desplazamiento máximo. Es muy importante realizar el proceso de calibración de forma precisa. Este proceso se describe en el ANEXO II del proyecto [5]. Ya calibrado, el

GameObject "*AmplifyManager"* se encarga de completar el movimiento hasta los 90 grados y aplicarlo al movimiento de *Phiby*.

La amplificación se aplica fundamentalmente a los brazos. El control de *Phiby* empleando movimientos del tronco no requiere amplificación, mientras que completar los ejercicios de talar un tronco o bien escalar un árbol requieren únicamente el movimiento de brazos del jugador. Se aplica a ambos ejercicios, donde se profundiza en los siguientes subapartados.

A continuación se explica el proceso seguido para aplicar correctamente la amplificación sobre cualquiera de los dos brazos que se quiera amplificar. Se toma como ejemplo el ejercicio de escalar un árbol, descrito en profundidad en apartados posteriores. Siguiendo el ANEXO II del proyecto [5] se calibra correctamente la amplificación en el brazo derecho. Sin embargo, surgen problemas cuando se aplica la calibración sobre el brazo izquierdo. Se adjunta en la figura 36 el resultado de este proceso.

Figura 36. Proceso de amplificación finalizado sobre el brazo izquierdo

Como se observa en la figura 36, el brazo izquierdo de *Phiby* se encuentra rotado. En un principio, si se sigue al pie de la letra toda la configuración mostrada en el ANEXO II [5] no debería surgir ningún tipo de problema. Esto ha ocurrido en el caso del brazo derecho, pero en el brazo izquierdo no. Por lo tanto, para solucionar este problema y para poder cubrir desarrollos futuros de nuevos ejercicios donde ocurran problemas parecidos al mostrado, se aporta la solución a continuación. En el *asset "AmplifyManager"* asociado a cada uno de los brazos, hay un parámetro que puede activarse/desactivarse. Este parámetro se denomina "*Init Rotation"* y se ubica en el *script* "*IKManager"* . En la figura 37 se adjunta la captura correspondiente.

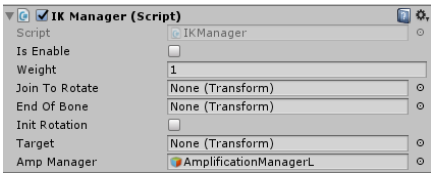

Figura 37. Parámetro Init Rotation desactivado en el script "IKManager"

Si se activa ese parámetro, por defecto aplica una rotación de -90 grados sobre la articulación que tenga asignada. Es decir, mediante este parámetro se puede configurar una rotación inicial que se aplica sobre la articulación en cuestión para poder ajustarla al cuerpo del personaje en caso de que no coincidan (como es el caso de este ejemplo mostrado sobre el brazo izquierdo).

Por lo tanto, para solucionar este caso en concreto, se edita el *script "IKManager"* y se aplica entonces una rotación inicial de -180 grados, haciendo así coincidir el brazo con el cuerpo del personaje. Se aporta el resultado final obtenido en la figura 38.

Figura 38. Brazo izquierdo rotado correctamente con el parámetro Init Rotation activado

4.2 Ejercicios de brazos

Los ejercicios de brazos que se integran en este proyecto y que forman parte de la aventura en "*Phiby's Adventures V2"* son dos: Talar un tronco y escalar un árbol. Estos ejercicios ya fueron desarrollados por César en su versión Demo [5], pero han sido integrados en esta nueva versión del videojuego manteniendo los mismos movimientos, insertándolos en puntos clave de la isla como p.ej. la zona de talar los troncos cercana a la cabaña de la isla.

De esta forma, el jugador no interpreta cada uno de ellos como si se tratasen de minijuegos o juegos independientes, sino que forman parte de las tareas que tiene que llevar a cabo *Phiby* dentro de la isla para poder alcanzar los objetivos marcados. Así, el jugador puede cortar los troncos cerca de la cabaña en la isla o bien trepar árboles específicos que se encuentran repartidos dentro del mundo.

4.2.1 Talar un tronco

Este ejercicio consiste en talar un tronco ubicado en frente del personaje mediante el movimiento ascendente del brazo del jugador. Como se comenta en el apartado anterior, se utiliza la amplificación del brazo en el movimiento para así aplicarla al personaje y completar el ejercicio. Se adjunta en la figura 39 la representación de este ejercicio dentro del mundo. El número de troncos a talar es un parámetro que podrá ser configurado por el terapeuta en función de las tareas de rehabilitación que esté llevando a cabo el jugador.

Figura 39. Ejercicio de talar un tronco con el brazo derecho completamente levantado

4.2.2 Escalar un árbol

Este ejercicio consiste en levantar los brazos realizando movimientos ascendentes para lograr que *Phiby* escale un árbol dentro de la isla. Mientras está escalando, el jugador debe ser cuidadoso y evitar que *Phiby* sea alcanzado por unos cohetes durante su ascenso. Análogamente, se aplica la amplificación en ambos brazos para que se pueda completar el movimiento de *Phiby* y se finalice el movimiento. La altura del tronco podrá ser configurada por el terapeuta en función de la rehabilitación seguida por el jugador y su evolución. Se adjunta a continuación en la figura 40 la representación de este ejercicio.

Figura 40. Ejercicio de trepar un árbol

4.3 Ejercicios de tronco

Los ejercicios de tronco que se presentan en el desarrollo de este proyecto también son dos: Volar en ala delta y Esquiar. El ejercicio de volar en ala delta fue también desarrollado por César en su versión Demo [5] y se han mantenido los mismos movimientos de control que fueron implementados.

Ambos consisten en mantener el equilibrio con el tronco mientras se controla el personaje. En este caso, los ejercicios de tronco no cuentan con la amplificación, ya que no se trata de una extremidad específica que necesite ser amplificada. Además, los movimientos del tronco son suaves y no requieren ningún tipo de sobreesfuerzo por parte del jugador para poder completarlos.

4.3.1 Volar en ala delta

Para la realización de este ejercicio, el jugador debe controlar la dirección en la que vuela *Phiby* mediante el tronco. Dado que el ala delta avanza por defecto, se deben controlar los giros, ascensos y descensos. El jugador debe alcanzar el máximo número de globos posibles dentro de ejercicio. Estos aparecen representados por colores junto a una flecha encima del ala delta que indica la dirección del próximo. El número de globos a conseguir es un parámetro configurable que podrá ser establecido por el terapeuta. Se adjunta en la figura 41 la representación del ejercicio.

Figura 41. Phiby a bordo del ala delta durante la ejecución del ejercicio

4.3.2 Esquiar

Una vez realizada la integración de los ejercicios iniciales desarrollados por César, es sencilla la generación de nuevos ejercicios que se basen en los movimientos implementados previamente. Es el caso del ejercicio de Esquiar, aquí descrito, donde se utilizan los movimientos de control de tronco empleados para controlar el ala delta.

En este ejercicio, el jugador debe controlar mediante el tronco el descenso de *Phiby* mientras esquía una montaña. Para avanzar, basta con que se incline ligeramente hacia delante y *Phiby*

iniciará el descenso. Para girar, basta con rotar levemente el tronco hacia los lados. En este proyecto no se ha definido el objetivo de la ejecución de este ejercicio, pero algunas ideas son: alcanzar la meta en un tiempo determinado, recoger el mayor número de objetos dispuestos en la montaña o bien esquivar el mayor número de objetos que se presenten en el camino de *Phiby*. Se adjunta en la figura 42 su representación.

Figura 42. Phiby en el ejercicio de esquiar durante un salto

4.4 Generación de escenas

Todos los ejercicios descritos en los apartados anteriores, salvo el ejercicio de esquiar, han sido integrados de la primera versión del juego [5]. Por lo tanto, en caso de que se quiera generar un ejercicio nuevo, basándose en los movimientos que ya están disponibles, se puede hacer uso de *prefabs*. Se han guardado los cuatro *prefabs* de *Phiby* correspondientes a los ejercicios de esquiar, talar, volar y escalar. Se adjunta en la figura 43 una captura correspondiente de la carpeta ubicada en la ruta *Assets/Prefabs/Character Prefabs*.

Figura 43. Prefabs de Phiby de cada uno de los ejercicios

De esta forma, si se quiere generar un ejercicio que implique, p.ej., el movimiento del brazo derecho de arriba hacia abajo se puede utilizar el *prefab "Talar_Phiby"* y arrastrarlo a la nueva escena donde se quiera implementar. Por otro lado, si se quiere generar un nuevo ejercicio en el que intervenga el movimiento del tronco del jugador, se pueden utilizar los *prefabs "Ski_Phiby"* y *"Volar_Phiby"*. Al disponer de los cuatro *prefabs* configurados y preparados para ser utilizados nada más arrastrarse en la escena, se simplifica en mayor medida el tiempo de desarrollo de una nueva escena.

En caso contrario, si se quieren desarrollar nuevos movimientos que formen parte de nuevos ejercicios, se pueden utilizar como base cualquiera de los cuatro *prefabs* citados previamente,

o bien seguir el ANEXO II del proyecto [5] y teniendo en cuenta la configuración del parámetro "*Init Rotation"* a la hora de aplicar la amplificación sobre una articulación del personaje. Se adjunta a continuación la figura 44 correspondiente a la generación de una nueva escena donde ha sido arrastrado el *prefab* del ejercicio *"talar"*.

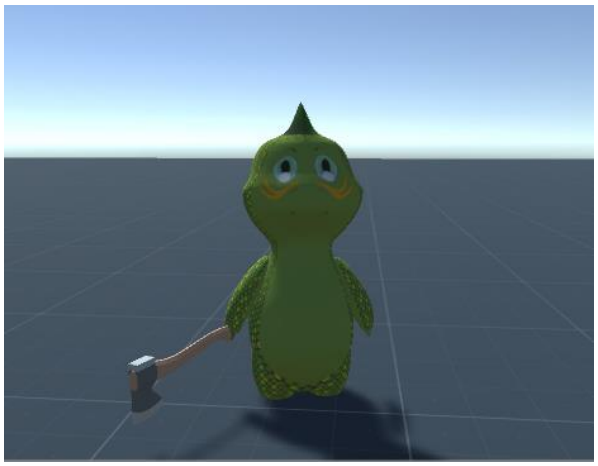

Figura 44. Prefab de Phiby en una nueva escena generada

5. Resultados

Dado que este proyecto está dirigido a personas que tengan capacidades motoras reducidas o dificultades a la hora de ejecutar ciertos movimientos, se han llevado a cabo una serie de pruebas funcionales sobre una persona de 82 años con dificultades de movimiento tanto en piernas como en brazos.

El primer juego de pruebas realizado engloba el control del personaje *Phiby* dentro de la isla. Se ha probado tanto con la persona de 82 años de pie como sentada en una silla frente a la *Kinect.* Ambas pruebas han aportado resultados muy satisfactorios, ya que el control del personaje resulta ser muy intuitivo y cómodo. En ningún momento se produce sobrecarga debido a una postura forzada o fatiga en su ejecución. Los controles de *Phiby,* a su vez, responden satisfactoriamente a los movimientos ejecutados. Por otro lado, al realizar las mismas pruebas de control del personaje con la persona

El segundo juego de pruebas engloba a los ejercicios de "*Volar en ala delta*" y "*Esquiar*". Ambos ejercicios se basan en movimientos del tronco y equilibrio para controlar al personaje. Los resultados de estas pruebas también han sido exitosos, ya que en ningún momento la persona que realiza las pruebas se ha visto forzada o bien ha resultado fatigada tras su ejecución. Se trata de movimientos simples y de fácil ejecución que responden correctamente aplicándose sobre el personaje en cada ejercicio.

A continuación se ha realizado la prueba correspondiente al ejercicio "*Talar un tronco"*. En este caso, el jugador debe levantar el brazo hasta una determinada altura para talar el tronco que se encuentra situado frente a *Phiby*. Los resultados lanzados tras su ejecución son también satisfactorios. La persona encargada de ejecutar el ejercicio no se ha visto

forzada en ningún momento a adoptar una postura forzada, estando cómoda durante su desarrollo. La amplificación, a su vez, responde correctamente a los movimientos. Se han probado distintas alturas a la hora de levantar el brazo para comprobar que la amplificación se ejecuta correctamente. El número de tronco talados inicialmente ha sido de 7 troncos. Dado que este ejercicio no tiene un límite establecido de troncos, se ha decidido cortar la prueba en ese punto ya que estaba respondiendo correctamente.

La última prueba realizada se ha basado en la ejecución del ejercicio "*Escalar"*. En este caso, se emplea el movimiento de los dos brazos del jugador. Cada uno de ellos está amplificado. Los resultados obtenidos son también exitosos, ya que el personaje responde correctamente a los movimientos llevados a cabo por el jugador y amplificándolos. Además, la persona encargada de realizar la prueba no se ha sentido fatigada durante la ejecución, ni se ha visto sometida a adoptar una serie de posturas que puedan haberle generado sensación de molestia.

Los juegos de pruebas han sido realizados en condiciones de distinta luminosidad y con numerosos elementos en la escena capturada, para poder comprobar el comportamiento robusto y fiable de la *Kinect*. La realización total de la prueba ha demorado un total de 30 minutos aproximadamente, donde se han repetido los movimientos en ciertas ocasiones para comprobar el correcto funcionamiento de estos.

Finalizado el primer juego de pruebas, se ha realizado una prueba más con una persona usuaria de silla de ruedas. La prueba realizada se centró en el control del personaje *Phiby* dentro de la isla. Los movimientos de giro del personaje respondieron correctamente, tanto hacia la izquierda como hacia la derecha. Sin embargo, los movimientos realizados para avanzar funcionaron bien, ya que el jugado tuvo que inclinarse mucho, casi una postura muy forzada, para poder lograr que el personaje avanzara.

6. Conclusiones

Al comienzo de este documento se establecen una serie de objetivos a alcanzar tras el desarrollo del proyecto. Todos ellos necesarios e importantes para dar mayor valor al resultado final. El primer objetivo consiste en mejorar y animar al personaje *Phiby*. Este objetivo se ha cumplido dado que el personaje cuenta con las tres animaciones principales (reposo, andar y correr) además de la reducción de caras y vértices del personaje.

El siguiente objetivo consiste en el control del personaje mediante dos entradas: teclado y movimientos corporales capturados del jugador a través de la *Kinect*. Dado que se ha desarrollado un control por teclado del personaje para la realización de pruebas y desarrollo, así como el control de *Phiby* mediante la *Kinect* gracias a los movimientos del jugador, este objetivo también se cumple.

El objetivo descrito a continuación trata la integración de diferentes tipos de movimientos del jugador en el juego para lograr que funcione sin teclado o ratón. Se ha cumplido la integración de estos movimientos del jugador (movimientos para mover al personaje y movimientos para realizar los ejercicios) salvo los movimientos de control ajenos al juego

(como abrir un menú). Se debe principalmente a la falta de tiempo y a lograr que tanto los movimientos sin amplificar como amplificados funcionaran correctamente.

Otro de los objetivos que se han presentado es el mantenimiento de los movimientos y ejercicios con fines terapéuticos. Los ejercicios de "*talar el tronco"*, "*escalar un árbol"*, "*esquiar"* y *"volar en ala delta"* mantienen los movimientos que se desarrollaron en la versión v1 del videojuego. El único ejercicio que no se ha podido integrar, debido principalmente a la falta de tiempo y a su mayor complejidad en el desarrollo (basada principalmente en numerosos *GameObjects* asociados a los puntos de unión entre los brazos y los remos), ha sido el ejercicio de "*Remar"*, donde se recogen movimientos de ambos brazos del jugador. Este ejercicio sí estaba disponible en la v1. Por lo tanto, se ha cumplido mayoritariamente este objetivo.

El último objetivo que se presenta en el documento es asegurar el comportamiento robusto de la *Kinect* en distintos entornos, así como variaciones de luminosidad y con elementos presentes en la escena. Este objetivo se cumple ya que durante todo el desarrollo del proyecto las pruebas se han realizado en ambientes poco luminosos y con muchos elementos presentes en la escena a capturar.

7. Futuras líneas de trabajo

El desarrollo de este proyecto constituye la primera versión del videojuego "*Phiby's Adventures V2"*. En ella, se integra el personaje *Phiby* dentro de la isla junto con los primeros ejercicios que pueden ejecutarse. Las posibilidades de cara a seguir ampliando el proyecto son casi infinitas. Dado que aún queda bastante camino por delante, las posibles futuras líneas de trabajo son las siguientes.

- Generación de nuevos ejercicios empleando los movimientos ya citados en este proyecto. Añadir, a los ejercicios de tronco y brazo ya existentes, nuevos ejercicios que empleen los mismos movimientos que ya se utilizan. Algunos ejemplos de ellos pueden ser: Controlar un vehículo dentro de la isla, arrancar hierbajos, encender antorchas dentro de la cueva, entre otros.
- Generación de nuevos ejercicios empleando nuevos movimientos de rehabilitación para ello. Recopilar información sobre posibles nuevos movimientos que tengan fines terapéuticos para añadirlos al videojuego. De esta forma, el abanico de ejercicios que un jugador pueda realizar será mayor.
- Creación de distintos niveles de juego. Al tratarse de la primera versión del videojuego, aún no se han podido establecer más niveles que el mostrado dentro de la isla con los ejercicios recogidos.
- Cinemáticas dentro del videojuego. Generar pequeños fragmentos animados en distintas partes del juego, así como por ejemplo, para cambiar de ejercicio o bien al iniciarse la aventura.
- Añadir algún efecto de sonido adicional o voz al personaje, para aumentar el realismo y enriquecer aún más la experiencia de juego.

- Capacidad para poder seleccionar entre varios personajes, así como su personalización. Al establecerse una variedad de personajes iniciales y la posibilidad de poder personalizarlos al gusto del jugador incrementa aún más el interés por el juego.
- Comunicación entre el videojuego y la web del terapeuta para seguir la rehabilitación de cada jugador. Actualmente, este apartado se encuentra en desarrollo paralelo con respecto a este proyecto.
- Posibilidad de incluir más dispositivos *hardware* de realidad aumentada, como gafas de realidad virtual. De esta forma, el jugador no tendría por qué girar el tronco para cambiar la vista del personaje si no que esta dependería exclusivamente del movimiento de la cabeza del jugador.
- Añadir un acompañante o amigo a *Phiby* durante la aventura que actúe a modo de guía y pueda aportar soporte o ayuda en caso de necesitarlo.
- Provocar un aumento en la velocidad de desplazamiento del personaje a medida que el jugado se incline hacia delante. Es decir, que se produzca un aumento exponencial de la velocidad a medida que el jugador aumenta su inclinación hacia delante en el juego.

8. Referencias

[**1**] UPM «Grupo de Aplicaciones Multimedia y Acústica (GAMMA)». *Fecha de última consulta 10/05/2019*

https://www.citsem.upm.es/index.php/es/personal/grupos/personal-gamma

[**2**] UPM «Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad (CITSEM)». *Fecha de última consulta 10/05/2019*

https://www.citsem.upm.es/index.php/es/personal/grupos/personal-gamma

[**3**] Microsoft «Kinect - Desarrollo de aplicaciones» *Fecha de última consulta 18/06/2019*

https://developer.microsoft.com/es-es/windows/kinect

[**4**] Unity Technologies. *Fecha de última consulta: 04/06/19*

https://unity3d.com/es/

[**5**] César Luaces Vela, «Diseño e implementación de un entorno virtual de ejercicios físicos, basados en la captura de movimiento» Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, UPM, Madrid, Julio 2018.

[**6**] Cristina Esteban González «Diseño e implementación del entorno de videojuego serio de rehabilitación del proyecto "Blexer"» Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, UPM, Madrid, Julio 2017.

[**7**] Martina Eckert, Ignacio Gómez-Martinho, Juan Meneses y José-Fernán Martínez, "New Approaches to Exciting Exergames-Experiences for People with Motor Functions Impairments", Sensors 17, 354, 2017.

[**8**] M. Eckert, I. Gómez-Martinho, C. Esteban, Y. Peláez, M. Jiménez, M.L. Martín-Ruiz, M. Manzano, A. Aglio, V. Osma, J. Meneses and L. Salgado, "The Blexer system – Adaptative full play therapeutic Exergames with web-based supervisión for people with motor dysfunctionalities", EAI Endorsed Transactions on Serious Games, 2018.

[**9**] REHABILITY «Rehabilitation games co-designed with patients» *Fecha de última consulta 1/06/2019*

http://www.rehability.me/#gallery

[**10**] ADVANT «Advanced Therapeutics» *Fecha de última consulta 1/06/2019*

http://advant.iter.es/

[**11**] Virtual Ware Group «VirtualRehab» *Fecha de última consulta 1/06/2019*

http://virtualwaregroup.com/es/productos/virtualrehab

[**12**] Eckert, M., Gómez-Martinho, I., Esteban, C., Peláez, Y., Meneses, J., and Salgado, L. (2017) Blexer – Full Play Therapeutic Blender Exergames for People with Physical Impaiments. 3rd EAI International Conference on Smart Objects and Technologies for Social Good, Pisa, Italy, Nov 29-30.

[**13**] Blender Project «Blender Documentation» *Fecha de última consulta 15/06/2019*

https://www.blender.org/

[**14**] Ignacio Gómez-Martinho González «Desarrollo e implementación de middleware entre Blender, Kinect y otros dispositivos» Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, UPM, Madrid, Julio 2016.

[**15**] Mónica Jiménez Ramos «Plataforma médica para el entorno del videojuego terapéutico BLEXER» Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, UPM, Madrid, Julio 2017

[**16**] Pablo López Miedes «Diseño e Implementación de un videojuego de ejercicio físico para personas con discapacidad» Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, UPM, Madrid, Julio 2017.

[**17**] Blender «Decimate Modifier» *Fecha de última consulta 15/06/2019*

https://docs.blender.org/manual/en/dev/modeling/modifiers/generate/decimate.html

[**18**] Blender «Bones Manual» *Fecha de última consulta 15/06/2019*

https://docs.blender.org/manual/en/latest/rigging/armatures/bones/index.html

[**19**] Blender «Blender Animation» *Fecha de última consulta 15/06/2019*

https://www.blender.org/features/animation/

[**20**] Blender «Blender – Editing Keyframes» *Fecha de última consulta 15/06/2019*

https://docs.blender.org/manual/en/dev/animation/keyframes/editing.html

[**21**] Blender «Blender – Importar objetos desde Blender» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/530/Manual/HOWTO-ImportObjectBlender.html

[**22**] Unity «Unity – Introducción a los componentes» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/Components.html

[**23**] Unity «Unity – GameObjects Manual» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/GameObjects.html

[**24**] Unity «Unity – Transform» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/class-Transform.html

[**25**] Unity «Unity - Animator» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/class-Animator.html

[**26**] Unity «Unity – Character Controller» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/class-CharacterController.html

[**27**] Unity «Unity – Creating and Using Scripts» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

[**28**] Unity «Unity – Main Camera» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/ScriptReference/Camera-main.html

[**29**] Unity «Unity - Scenes Manual» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/Manual/CreatingScenes.html

[**30**] Unity «Unity – Prefabs Unity Manual» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/Prefabs.html

[**31**] Unity «Unity – Asset Packages» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/Manual/AssetPackages.html

[**32**] Unity «Unity – Asset Workflow» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/Manual/AssetWorkflow.html

[**33**] Unity «Unity – Blend Tree Manual» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/Manual/class-BlendTree.html

[**34**] Unity «Unity – Animator Documentation» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/Animator.html

[**35**] Unity «Unity – Capsule Collider» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/class-CapsuleCollider.html

[**36**] Unity «Unity - RigidBody» *Fecha de última consulta 15/06/2019*

https://docs.unity3d.com/es/current/Manual/class-Rigidbody.html

ANEXO I: Asignación del personaje *Phiby* al *KinectAsset*

A continuación se presenta el procedimiento seguido durante la realización del proyecto para asignar las extremidades y cuerpo de *Phiby* al *asset "KinectAsset"* para el control del personaje mediante *Kinect*. Siguiendo el proceso estipulado en [5], el *KinectAsset* desglosado se muestra a continuación.

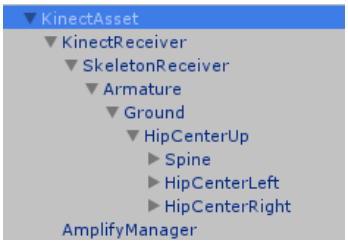

Figura 45. KinectAsset desglosado asociado a Phiby

Al seleccionar el elemento *"KinectReceiver"* mostrado en la figura 45, aparece en la ventana *Inspector* el *script "Launcher"* . Este se encarga de asociar las distintas articulaciones del esqueleto del *KinectAsset* con las articulaciones del modelo 3D empleado (*Phiby* en este caso). Se adjunta en la figura 46 una captura del contenido del *script*.

Figura 46. Script "Launcher"

Dado que en este caso el *KinectAsset* se emplea para el control del personaje dentro de la isla, se activa el parámetro "*Is Spine Base"* para asociar el tronco del jugador y que se

interpreten los movimientos de este. De esta forma, cualquier movimiento que el jugador ejecute con el tronco superior queda recogido e interpretado. Como se observa en la figura 46, el parámetro *"Is Spine Base"* tiene asignado un *GameObject* denominado "*Detector"*. Este se encuentra dentro del personaje *Phiby*. Como se quiere que el personaje se desplace por la escena, pero que no simule los movimientos del jugador porque en este caso en concreto no debe copiarlos, "*Detector"* se define como un objeto vacío.

De esta forma, cada vez que el jugador ejecute algún movimiento de rotación o de inclinación con el tronco, se aplicará al personaje para desplazarlo sin copiar el movimiento ejecutado del jugador sobre el personaje. En caso de que sí se quieran copiar los movimientos del jugador, se deben activar los parámetros del *KinectAsset* que correspondan con estas articulaciones. Para el caso del brazo izquierdo, p.ej., durante el ejercicio de talar, se adjunta la figura 47.

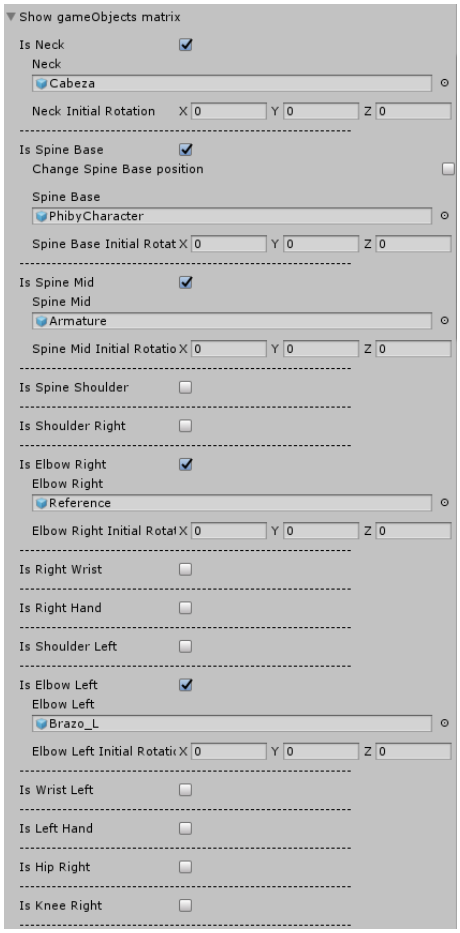

Figura 47. Script "Launcher" en la ventana Inspector asociado a Phiby

Como se observa en la figura 47, para que se copie el movimiento del brazo izquierdo está activado el parámetro "*Is Elbow Left"*. Asociado al mismo se encuentra el brazo izquierdo del personaje *Phiby*, mostrado en la figura 48 a continuación.

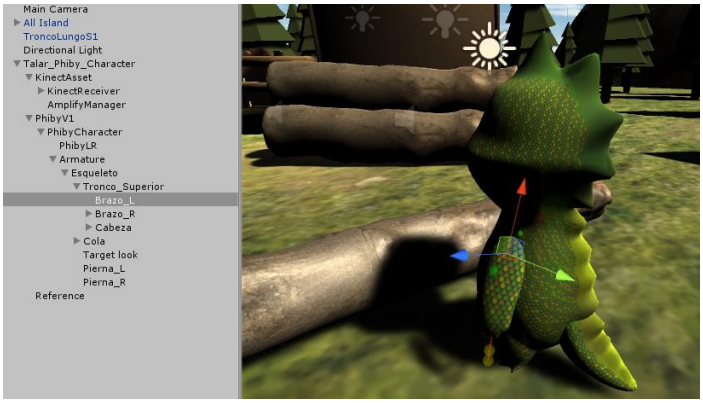

Figura 48. Brazo izquierdo (Brazo_L) de Phiby asociado al KinectAsset

De esta forma, se pueden asociar todas las articulaciones necesarias que intervengan en un ejercicio, al modelo 3D que se esté empleando.