ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: 'Blexer' – Desarrollo de videojuego terapéutico para personas con movilidad reducida

AUTOR: Jennie Yadira Peláez Castro

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

TUTOR: Martina Eckert

DEPARTAMENTO: Departamento de Teoría de la Señal y Comunicaciones

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Marisa Martín Ruiz

TUTOR: Martina Eckert

SECRETARIO: Enrique Rendón Angulo

Fecha de lectura:

Calificación:

El Secretario,

AGRADECIMIENTOS

A mis padres, por el esfuerzo que han hecho
para brindarme los estudios y la educación que tengo.

A mi tutora Martina, por la oportunidad
de participar en un proyecto tan enriquecedor.

A mis amigos, cada uno de ellos
han hecho de esta etapa un viaje asombroso.

Resumen

En este estudio se describe la primera fase de desarrollo del videojuego “Phiby’s Adventure”. Se trata de un videojuego serio de rehabilitación para personas con movilidad reducida. Estas personas deben asistir periódicamente a sesiones en las que realizan ejercicios para mantener o mejorar su movilidad. Sin embargo, a menudo estas sesiones son dolorosas y pesadas, lo que hace que el paciente se desmotive. Es por esto que se busca realizar un juego que les permita rehabilitarse sin que se centren en este hecho, sino en entretenerse y sentirse motivados mientras mejoren.

Este proyecto forma parte del proyecto “Blexer” (Blender *Exergames*), una de las líneas de investigación de interfaces naturales dirigida por la Doctora Martina Eckert en el Grupo de investigación GAMMA (Grupo de Aplicaciones MultiMedia y Acústicas) asociado al CITSEM (Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad).

El juego se ha desarrollado con la primera versión del sensor Kinect de Microsoft, el *software* gratuito Blender para modelado en 3D, el *middleware* Chiro y un *Add-on* para Blender, estos últimos desarrollados por Ignacio Gómez-Martinho [1]. Chiro comunica los datos recibidos desde la Kinect que capta al paciente con Blender, que es el entorno en el que se va a desarrollar el videojuego.

En concreto se han desarrollado dos de cuatro mini-juegos que permiten el control de un Avatar con forma de anfibio (Phiby) con el tronco y los brazos. Además del videojuego, se ha colaborado con Mónica Jiménez [2], que ha desarrollado una plataforma web médica (Blexer-Med), lo que permite la configuración de la dificultad de los mini-juegos por un terapeuta a distancia. También se han realizado pruebas con pacientes del centro ASEM (Asociación Madrileña de Enfermedades Neuromusculares) que han tenido una experiencia satisfactoria.

Para las personas con este tipo de problemas es difícil la utilización de mandos de consolas de videojuegos, ya que normalmente sus movimientos no son captados correctamente o sin la precisión necesaria. Una de las innovaciones de este proyecto es la posibilidad de ampliar los movimientos del paciente a partir del *Add-on* instalable en Blender. Permite asociar un objeto amplificador a alguna parte del cuerpo como las manos, el tronco o las piernas.

Abstract

This study describes the first phase of development of the serious game “Phiby’s Adventure” for people with reduced mobility. These people must assist periodically physiotherapy sessions to maintain or improve their physical condition. Nevertheless, often these sessions are tedious and exhausting. It could also happen that the patients feel demotivated and frustrated. For that reason, in this research, the main object is to develop a game that allows doing exercise while having fun and without noticing the same amount of effort.

This project belongs to the project “Blexer” (Blender Exergames), one of the research lines guided by Doctor Martina Eckert, within her research group GAMMA (*Grupo de Aplicaciones MultiMedia y Acústicas*) at CITSEM (*Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad*).

This game has been developed using the first version of Kinect Microsoft, the free and open-source 3D computer graphics software Blender for modelling, the middleware Chiro and the Blender’s Add-on developed by Ignacio Gómez-Martinho [1]. Chiro transmits the data received from Kinect to Blender. It is the environment where the game has been developed. The Kinect captures the patient’s skeleton that allows to know the movements of the user every moment.

Four mini-games have been created for this version. Here, the patient can control an amphibian like character (Phiby). It can be controlled by using the trunk and the arms. This project has been realized in collaboration with Mónica Jiménez [2]. She developed the web platform, “Blexer-Med”, which allows to configure the mini-games individually for each patient and to register as many patients and therapist as necessary, as well as storing the data in a database.

Tests have been performed with members of the association ASEM (*Asociación Madrileña de Enfermedades Neuromusculares*). They have had a satisfactory experience.

For persons with physical restrictions it is often difficult to use remote controls to play videogames. Sometimes, their movements are not correctly detected. One of the innovations of this study is the possibility to amplify the movements of the players with the Blender’s Add-on. It can be applied for those limbs that need this, like arms, trunk or legs, to achieve a more immersive experience in the game.

Índice

1. Introducción y objetivos

Uno de los aspectos más importantes tras el diagnóstico de una enfermedad neuromuscular es tratar de mantener las actividades de la vida cotidiana. Para ello es necesario realizar ejercicios de rehabilitación, tanto en un hospital como en el hogar. Muchas de las terapias están basadas en repeticiones para conseguir un rango de movimiento o control sobre un grupo muscular específico. Este proceso tiende a ser doloroso y poco motivante para el paciente, lo que provoca que la rehabilitación se vuelva lenta y frustrante. En ocasiones, es necesario complementar esto con ejercicios en casa prescritos por el terapeuta. Sin embargo, no es posible asegurar en estos casos la correcta ejecución de los mismos. Lo ideal sería proveer al paciente con la tecnología necesaria para tener un sistema de rehabilitación casero que ofreciera los mismos resultados que las sesiones en el hospital, pero sin la necesidad de desplazarse.

En la actualidad, existen diversas líneas de investigación que intentan integrar la tecnología en los procesos de rehabilitación. Es en el CITSEM (Centro de Investigación en Tecnología Software y Sistemas Multimedia), en el grupo GAMMA (Grupo de Aplicaciones Multimedia y Acústica) donde se ha desarrollado el proyecto "BLEXER" (Blender Exergames), que se ajusta a esta rama tecnológica.

Iniciado en 2015 por Ignacio Gómez-Martinho [1] y tutorizado por la Doctora Martina Eckert, se creó un *middleware* que permitía la conexión entre la Kinect de Microsoft y el *software* de animación Blender. Se buscaba entonces captar el movimiento de los pacientes sin necesidad de utilizar equipos que tuvieran que ser portados por el usuario. Durante el siguiente año colaboré en el proyecto en términos de prácticas junto con Pablo Parra [3] e Ignacio Gómez-Martinho. En este periodo se crearon cuatro mini-juegos que ponían a prueba el *middleware* mencionado y se añadieron funcionalidades como la ampliación de movimientos.

Tras algunas pruebas con personas con movilidad reducida, cuyos resultados se publicaron en [4], se obtuvo el *feedback* necesario para dar un paso más en la investigación. Es en este punto donde se inicia el actual proyecto junto con las alumnas Cristina Esteban [5] y Mónica Jiménez [2]. El objetivo global es la creación de un videojuego serio que permita la rehabilitación de forma dinámica, entretenida, motivante y que guarde un registro y control de la evolución de los pacientes.

"BLEXER" constará de tres módulos: el videojuego, el *middleware* y una plataforma web. Mónica Jiménez se encarga del desarrollo de una plataforma médica que permita la gestión de pacientes y terapeutas, así como de la configuración de los parámetros del juego. Por otra parte, junto a Cristina Esteban se modela y crea el entorno, los personajes del videojuego "Phiby's Adventure", y la adaptación de los mini-juegos desarrollados el año anterior en el nuevo contexto.

Los objetivos concretos para este proyecto son:

- Describir el entorno global y como se integran las funcionalidades de los mini-juegos existentes.
- Transformar dos mini-juegos realizados en el semestre anterior para que encajen en el nuevo entorno.

- Modelar un personaje común a todos los mini-juegos y el entorno del videojuego.
- Implementar mecanismos de juego como retos o puntuaciones, con el fin de motivar al usuario.
- Para la integración de la plataforma médica llevada a cabo por Mónica Jiménez, realizar la comunicación entre esta y el juego en Blender.
- Mejora del funcionamiento de los ejercicios que se realizan en los mini-juegos.

2. Marco tecnológico y antecedentes

2.1. Estado del arte

Para saber desde qué punto se parte en el proyecto, es necesario hacer una búsqueda de la situación actual en cuanto al uso y las aplicaciones de Kinect orientadas al campo que interesa, la rehabilitación.

A continuación, se enumeran algunos de los proyectos más interesantes que han aportado a la investigación actual.

- SeeMe [6]

  Es un sistema clínico de PC, para el control de ejercicios y diagnóstico. Ha sido diseñado como sistema auxiliar para el proceso de la rehabilitación y el seguimiento de los pacientes. Mejora la coordinación, el equilibrio, la fuerza muscular, el rango de movimientos, el tiempo de reacción y la memoria.

  La mayor de sus fortalezas es la posibilidad de adaptación a las necesidades individuales, objetivos e intereses científicos. Mediante la herramienta *SeeMe Manager* es posible generar informes del resultado de los tratamientos, ajustando los parámetros y toma de anotaciones mientras el paciente juega. Los ejercicios se practican a través de juegos, buscando que el paciente se sienta motivado a realizarlos. La aplicación tiene una librería de juegos independientes que son configurables y con distintos niveles. En la Figura 1 se muestran algunas de estas opciones.

Figura 1. Juegos del sistema SeeMe

- KineLabs Games [7]

  El Doctor Raymond Tong Kai-yu y la asociación de Profesores en el campo de la ingeniería Biomédica de la Universidad Politécnica de Hong Kong ha desarrollado el *software* gratuito “KineLabs” para facilitar la rehabilitación para personas de edad avanzada y pacientes que han sufrido un derrame cerebral. Usando la tecnología del sensor Kinect, el sistema puede capturar los datos de posición 3D de cada parte del cuerpo para su análisis. Puede ser utilizado tanto en centros

especializados, como en casa. Los juegos que componen el programa ayudan a mejorar la movilidad de los pacientes.

El Doctor Raymon Tong indica que una de las ventajas frente a otros sistemas existentes es que permite el entrenamiento de los movimientos en 3D, mientras que el resto suelen ser juegos en dos dimensiones. Esto permite estiramientos de los miembros superiores e inferiores, que es lo que los usuarios realmente necesitan.

Otro punto a remarcar es que el software tiene incorporada una función de calibración. El usuario realiza un test simple que establece el nivel apropiado para él. También permite que dos usuarios con diferentes niveles de movilidad jueguen simultáneamente.

Este software fue reconocido con el premio de plata en los premios ICT Hog Kong de 2012: Mejor Innovación e Investigación.

En Figura 2 se muestran algunos ejemplos de usuarios utilizando este *software*.

Figura 2. KineLabs testeado por pacientes reales

- Rehabtimals

En la publicación [8] se expone la investigación y creación del videojuego de rehabilitación "Rehabtimals". Se trata de un mundo virtual donde el paciente se recupera de sus lesiones mientras un avatar en forma de animal crece y sobrevive en el entorno en el que el juego se desarrolla. El proyecto se compone de un módulo servidor y un módulo para el terapeuta. En este segundo, se controla el tratamiento del paciente analizando las grabaciones de las reconstrucciones en 3D de los movimientos que realiza en las diferentes sesiones. En Figura 3 se pueden ver algunas de las pantallas del juego.

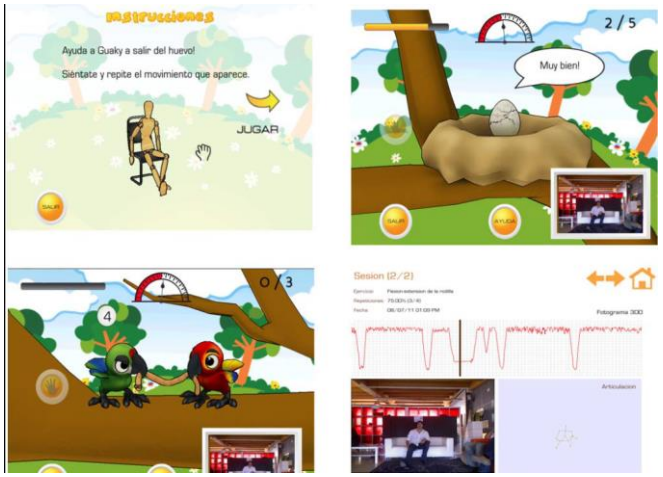

Figura 3. Captura de la aplicación "Rehabtimals". De arriba abajo, de izquierda a derecha: Pantalla de instrucciones. Pantalla de Nivel 1. Pantalla de nivel 2. Sesión de "Rehabtimals Pro" para el terapeuta.

- MIRA (*Medical Interactive Recovery Assistant*) [9]

  MIRA es una plataforma diseñada para enfocar la fisioterapia de forma que sea divertida y amena para la recuperación de los pacientes de cirugías o lesiones. El sistema transforma los ejercicios de fisioterapia en videojuegos. Usa el sensor de Kinect como control, seguimiento y evaluación del progreso de los usuarios a través de parámetros recopilados a través del mismo, como la velocidad en la que el paciente ejecuta los movimientos. Posee tutoriales para que los juegos sean fáciles de seguir. También permite que sea utilizado en casa. En Figura 4 se muestra una sesión con la plataforma. Actualmente este sistema se comercializa, y ha sido reconocido con tres premios*:* Imagine Cup 2011 - Final Round, New York - Top 6 Worldwide; Imagine Cup 2011 - Romania National Round, Bucharest - 1st place; y Imagine Cup 2011 - UBB Local Round, Cluj-Napoca - 2nd place.

Figura 4. Pruebas MIRA Rehab (imagen de su página de facebook)

2.2. Herramientas de trabajo

2.2.1. Kinect

Microsoft Kinect [10] fue lanzado al mercado en 2010 como un complemento de la consola XBOX 360. Permite al usuario interactuar con la XBOX 360 sin un control físico a través de una interfaz natural, usando gestos, la voz o imágenes. Posee una cámara RGB, un emisor infrarrojo, un sensor de profundidad y un micrófono de tipo *array*.

Los datos que interesan para el proyecto se transmiten por USB a través de flujos o *streams.* Se puede acceder a estos datos a través de funciones con ayuda del SDK (*Software Development Kit*) oficial de Microsoft. El más importante es el *stream* de esqueletos. En estos se obtiene la lista de los usuarios que detecta la Kinect. Puede haber hasta un máximo de seis, sin embargo, únicamente dos pueden tener la información de todo el esqueleto. En dichos *streams* se obtiene, en tiempo real, la posición de las articulaciones, hasta un máximo de veinte. También se aporta la localización del plano del suelo y la inclinación del sensor. Estas características han permitido que el uso de este dispositivo haya ido más allá que el de complementar una videoconsola [11].

Las 20 posiciones de articulaciones, mostradas en Figura 5, incluyen cabeza, el centro de los hombros, hombro derecho e izquierdo, codo derecho e izquierdo, muñeca derecha e izquierda, mano derecha e izquierda, tronco, centro de la cadera, cadera derecha e izquierda, rodilla derecha e izquierda, tobillo derecho e izquierdo y pie derecho e izquierdo.

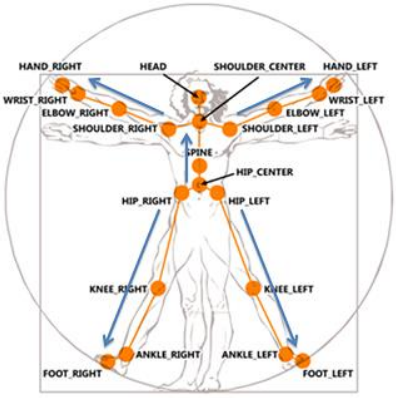

Figura 5. Articulaciones del Esqueleto Kinect [20]

En cuanto a la precisión de Kinect, [8] en su investigación compara la captura de movimiento entre Kinect y la herramienta *Optical Motion Capture Data*. Este último es un sistema óptico que consta de 4 a 32 cámaras y un ordenador que las controla. Normalmente, el sujeto tiene una serie de marcadores que son reflectantes (pasivos) o emisores (activos). Las cámaras en este sistema pueden capturar entre 30 y 2000 cuadros por segundo.

Al menos dos cámaras tienen que capturar un mismo marcador para poder realizar una representación 3D. Tras la sesión de captura, se quita el ruido y se recuperan los marcadores "perdidos". Este tipo de sistemas es bastante preciso, por lo que en dicho estudio se utiliza para testear la precisión de Kinect.

Durante el estudio se realizaron ejercicios utilizando las extremidades superiores e inferiores, como si de una sesión de rehabilitación se tratase, las cuales fueron grabadas por ambos sistemas.

Las conclusiones de este son, como era previsible, que la precisión de Kinect es menor que el sistema de captura óptico. Sin embargo, tiene otras ventajas: económicas, de portabilidad y de intrusión para el paciente, ya que no es necesario utilizar marcadores. Además, el grado de precisión obtenido para las articulaciones principales es suficiente para permitir que Kinect se utilice como tecnología en los tratamientos de rehabilitación.

2.2.2. Blender, Python y C#

Como se ha indicado en la introducción, para satisfacer los diversos objetivos del proyecto, se ha optado por trabajar con Blender, un software gratuito para la creación de videojuegos en 3D. Con este se pueden realizar todas las fases que suponen la creación de un juego de este estilo, como son modelado, *rigging*, animación, simulación, *rendering,* composición y rastreo de movimientos. En definitiva, permite el libre modelado de objetos, que posteriormente es integrado en los escenarios junto con sistemas de luces, partículas, código ejecutable en Python, etc.

Si se desea añadir funcionalidades a Blender para que sea compatible con la Kinect, la opción más evidente a priori es escribir un código, que desde los ejecutables de los juegos, acceda a la Kinect y obtenga los datos sobre las posiciones del jugador. Sin embargo, no existe esa posibilidad, ya que

Blender solo permite ejecutar código en el lenguaje de programación Python, y el kit de desarrollo oficial de Kinect es una biblioteca que sólo puede ser utilizada en los lenguajes C#, C++ y Visual Basic.

Esto hace necesario la codificación de un software intermediario (*middleware*), en uno de los lenguajes compatibles, que controle el funcionamiento del sensor, obtenga sus datos y se los retransmita a Blender. La mayoría de los recursos online sobre el desarrollo con el SDK 1.8 de Kinect contemplan sólo el desarrollo con C#, por lo que ese es uno de los lenguajes que han sido utilizados.

2.3. Antecedentes

En relación a lo anterior indicado, era necesario desarrollar dos herramientas de software. Por un lado, un *middleware* que se ocupara de leer la posición del usuario y transmitirla; y por otro, un código instalable en Blender (*Add-on*) que implementase el sistema de control para videojuegos utilizando los movimientos del jugador.

El *middleware* Chiro, denominado así por su autor Ignacio Gómez-Martinho González, se diseñó con un enfoque modular, permitiendo añadir funcionalidades. Este es capaz de controlar *smartphones*, cascos de realidad virtual y la cámara Kinect.

Cada dispositivo se comunica mediante USB, y su comunicación con Blender se hace mediante distintos *Add-ons*. En este proyecto, el utilizado es el que recibe los datos desde la Kinect mediante Chiro.

La primera versión de este *add-on* permitía recibir las coordenadas de las articulaciones proporcionadas por Kinect y a partir de ellas crear varios objetos: esqueleto estático, de posición, de rotación y de rotación y posición; cuerda; y acelerómetro. El aspecto de esta versión es el indicado en Figura 6.

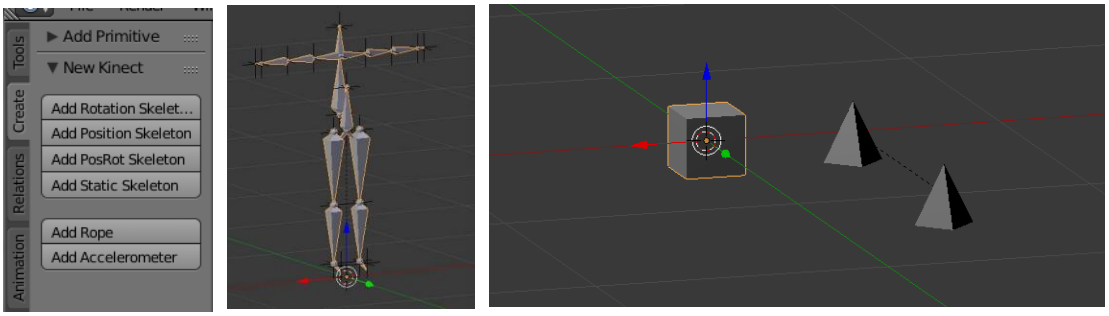

Figura 6. De izq a derc: Interfaz del addón; Esqueleto combinado; Cuerda y acelerómetro.

Para la realización del *add-on* se utilizó la librería *PythonOSC* [12], que permite leer las cadenas de texto y las variables *float* recibidas desde el puerto configurado.

En una segunda versión de este *add-on*, tras seleccionar el esqueleto estático, aparece la opción de añadir un objeto amplificador como se muestra en la imagen de la izquierda de la Figura 7. Su función es detectar movimientos ligeros de una extremidad del jugador (mano, pie, cabeza o torso) y escalarlo, para que traducido en el avatar se asemeje al movimiento que realizaría una persona sin movilidad reducida. La amplificación se establece midiendo la distancia que se ha desplazado una extremidad desde su punto de reposo, y señalando el punto del espacio que equivale a ese desplazamiento

multiplicado por un factor. Dicho factor tendrá tres componentes, ya que se trata de una amplificación 3D [1].

En la imagen derecha de la Figura 7, se puede apreciar el resultado de dicho amplificador. Se trata de la articulación de la muñeca en el esqueleto estático. La esfera roja representa la posición de reposo; la esfera verde muestra la posición real del paciente al levantar la mano; y la esfera amarilla la posición del movimiento amplificado.

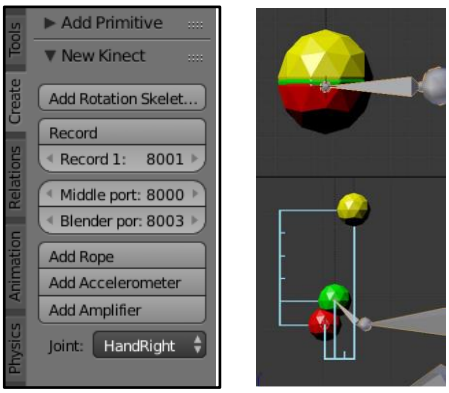

Figura 7. Izq:Interfaz del Add-on con el amplificador de movimientos. Derch: Movimiento amplificado [1].

Una vez se implementó el sistema, se procedió a esquematizar el que sería el juego serio de rehabilitación.

Mediante una lluvia de ideas se realizó un listado de los posibles ejercicios que se podrían incluir en el proyecto, acordes a las funcionalidades ya descritas. De todos ellos se implementaron cuatro mini-juegos que pondrían a prueba el sistema.

Los mini juegos son:

- **Trepar**
  En este mini-juego se copia mediante Kinect y el *add-on* los movimientos del usuario. El ejercicio se centra en el movimiento de los brazos. El objetivo es que el personaje suba una escalera, controlado por el movimiento alternativo de los miembros superiores (Figura 8 arriba izq.).
- **Remar**
  El ejercicio de rehabilitación en el que se quiere profundizar es el movimiento de los brazos en sentido horizontal. Movimientos de delante hacia detrás, como si el personaje estuviera remando (Figura 8 arriba der.).
- **Golpear**
  En este ejercicio el personaje copia los movimientos que el usuario realiza. El movimiento consiste en apuntar con el brazo a una dirección donde aparece un topo, levantarlo y dejarlo caer (Figura 8 abajo izq.).

- **Volar**

  En este caso el control del personaje se realiza con el tronco. El usuario tiene que esquivar y atravesar una serie de obstáculos, como si estuviera volando (Figura 8 abajo der.).

Figura 8. De arriba a abjo, de izquierda a derecha: Trepar; Remar; Golpear; Volar.

En paralelo a la implementación de los ejercicios, se buscó un personaje principal y un entorno o mapa donde los movimientos claves para la rehabilitación concordasen con acciones entretenidas y de superación. En Figura 9 se muestran los bocetos realizados para esta tarea.

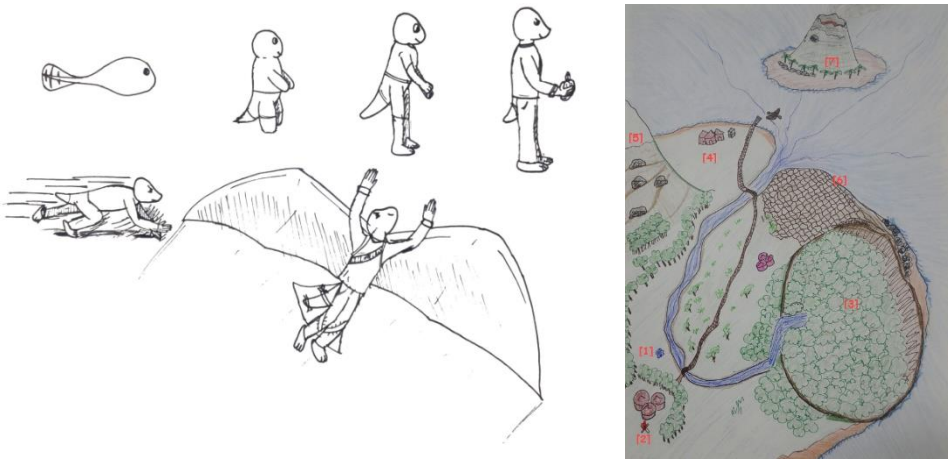

Figura 9. Izq: Boceto de la evolución del personaje principal. Derech: Boceto del mapa del mundo virtual.

La sinopsis del videojuego, redactado junto a Pablo Parras e Ignacio Gómez-Martinho, es la siguiente:

“Un mundo poblado por animales inteligentes, que viven en pequeños poblados en una isla. El protagonista despierta en un lago, convertido en un diminuto renacuajo, sin recuerdos de quién es ni de cómo llegó allí. Con la ayuda de los habitantes del lugar, consigue salir del agua y desarrollar unas pequeñas extremidades. Investigando lo ocurrido, descubre que un misterioso villano está utilizando magia para robar los poderes evolutivos de todos los animales: la capacidad de salto de las ranas, la velocidad de los leopardos, el vuelo de los pájaros, etc. El héroe debe emprender una aventura para rescatar a todos los animales en apuros, conseguir nuevas habilidades con las que evolucionar y derrotar a quien esté detrás de todo.”

En torno a mayo de 2016 se concertó una cita con la fisioterapeuta Maite Manzano, que dirige el Servicio de fisioterapia de la asociación ASEM, la cual analizó los ejercicios implementados dando *feedback* al proyecto sobre las carencias que tenían los mini-juegos en dicho momento.

Tras un semestre de trabajo se obtuvieron 4 mini-juegos, los indicados anteriormente, más un ejecutable de configuración (*SetAmp*) que servía para inicializar el esqueleto que se usaría para reproducir los movimientos del paciente. En esta inicialización se evalúa el máximo rango de movimiento que tiene el jugador y se configuran los amplificadores del juego en función de sus necesidades.

A finales del semestre de primavera de 2016, colaboré en las pruebas de los mini-juegos con algunos voluntarios del centro ASEM (Asociación de Enfermedades Neuromusculares Madrid) y con un grupo de control sin ningún tipo de limitación de movimiento [4]. La mayoría de los afectados eran niños entre 5 y 12 años en silla de ruedas.

De este encuentro se pretendía evaluar el interés que los usuarios potenciales mostraban hacia los mini-juegos, así como analizar las mejoras necesarias con los amplificadores y la lógica de los juegos.

En las pruebas en las que los sujetos se encontraban en silla de ruedas y eran de pequeño tamaño en comparación con la misma, la Kinect tenía problemas de reconocimiento del esqueleto y reproducía erróneamente los movimientos en el juego.

En cuanto a impresiones, los participantes tuvieron un grado alto de satisfacción con el proyecto.

3. Solución propuesta

Tras las pruebas con pacientes, comenzó una nueva etapa del proyecto Blexer, dentro de la cual se ubica el actual PFG.

Como se ha indicado en la introducción, se busca crear un juego que sea desafiante, motivador y permita la realización de ejercicios sin tener la sensación de que sean obligatorios. Sin embargo, hasta el momento se tenían cuatro mini-juegos inconexos. Es por ello que antes de implementar el videojuego, era necesario definir una historia y su guion, para poder continuar el proceso de creación.

Como personaje principal, que puede ser atractivo para un rango amplio de edades (p.ej. 6 a 26 años), se inventó a "Phiby", una especie de anfibio extraterrestre. La historia sería que se despierta sumergido en un lago sin saber quién es ni de dónde viene. Inicialmente no tiene extremidades inferiores, solo una cola y brazos diminutos que le permiten nadar rápidamente, como un renacuajo. Para sobrevivir se debe alimentar del plancton que encuentra a su alrededor, eso también le permite crecer. Cuando se desarrolla obtiene unas patas que le permiten salir fuera del agua y explorar el nuevo mundo que le rodea. Así, el paciente controla a Phiby en su aventura a través de varios mini-juegos con una temática en común y está motivado a hacerle crecer y avanzar en el juego. Se encuentra la historia completa redactada por Cristina Esteban en el Anexo 1 de su memoria [4].

A partir de la historia y la temática del juego, se han de adaptar los cuatro mini-juegos existentes al nuevo entorno:

- **Trepar**
  Este mini-juego se centra en el uso de los brazos, levantándolos lo máximo que se pueda para aparentemente sujetarse a la escalera e ir subiendo peldaño a peldaño. El movimiento de las piernas en este ejercicio es animado.
  Se barajan distintas opciones para incluir este ejercicio en el videojuego, como por ejemplo escenas en las que el personaje evolucionado tenga que trepar a un árbol para coger frutos o agarrarse de una liana teniendo que levantar los brazos.
- **Remar**
  La funcionalidad de este ejercicio se quiere mantener en el videojuego. Se puede incluir en alguna escena en la que el personaje evolucionado se monte en una barca y reme en un lago o de una isla a otra.
- **Golpear**
  Se quiere integrar este ejercicio en el juego en una escena en la que el paciente tenga que levantar el brazo y "golpear" seguidamente otro objeto. Por cuestiones éticas (trato con menores) no se debe incluir violencia de ningún tipo en el juego, pudiéndose sustituir los topos por madera que tenga que cortar el personaje.
- **Volar**
  En las pruebas realizadas, resultaba muy cansado mantener los brazos en cruz, y no todos los pacientes eran capaces de realizar dicho movimiento. Debido a ello y a que se quiere modelar dos estados del personaje (renacuajo y adulto), se sustituye el mini-juego de volar por bucear. Se puede

incluir este ejercicio en la primera escena en la que Phiby se encuentra en un lago, en el que el personaje es manejado mediante el movimiento del tronco del paciente.

Se quiere implementar una serie de escenarios que muestren el mundo donde se va a desarrollar el juego, el personaje principal con dos de sus evoluciones, mejorar la programación de los mini-juegos ya creados y testear su funcionalidad realizando pruebas con usuarios potenciales. Este desarrollo formará parte del proyecto piloto del videojuego.

Se decide adaptar los mini-juegos a 4 escenas:

- *Dive and Eat* (Bucear y comer, antes "Volar")
- *Chop the Wood* (Cortar la madera, antes "Golpear")
- *Climb the tree* (Trepar el árbol)
- *Row the boat* (Remar en la barca)

Una vez establecidas las escenas y los mini-juegos a adaptar se realiza el siguiente reparto de tareas:

Cristina Esteban:

- Guion del juego completo.
- Estudio de las técnicas de gamificación [13].
- Modelar el personaje en su segundo estado y objetos (casa, árboles)
- Transformar dos de los mini-juegos: *Chop the wood* y *Row the boat*
- Integrar el Avatar en los nuevos mini-juegos y mejorar su manejo

Yadira Peláez:

- Modelar el personaje en su primer estado (renacuajo)
- Modelar el entorno global e integrar las funcionalidades de los mini-juegos existentes.
- Transformar dos de los mini-juegos: *Dive and Eat* y *Climb the tree*
- Integrar el Avatar en los nuevos mini-juegos y mejorar su manejo
- Para la integración de la plataforma médica, desarrollar un *script* que permita guardar los resultados del juego y cargar los valores de configuración establecidos por el terapeuta

3.1. Diseño, modelado y animación

El *software* a utilizar para el diseño, modelado y animación es Blender. Antes de entrar en el detalle del desarrollo de los mini-juegos, en el Anexo 1 se adjunta un glosario de algunas características que posee el software, y en el Anexo 2 una explicación de los pasos a seguir para el desarrollo de un videojuego en Blender, nociones necesarias para poder seguir la implementación de los mini-juegos.

La primera fase del proyecto se dedica al modelaje de objetos comunes, las dos fases del personaje principal y el entorno.

A partir de la idea desarrollada se realizan dos bocetos. Uno de la primera etapa del personaje principal en forma de renacuajo (Figura 10) y otro de sus diferentes fases, el cuál fue realizado por Cristina Esteban [5].

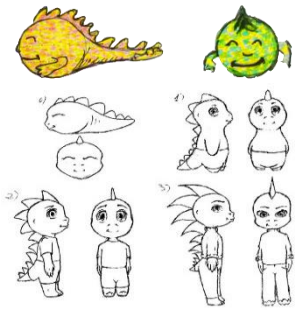

Figura 10.Bocetos del personaje principal. Arriba: Renacuajo. Abajo: Evolución del personaje

Todos los objetos realizados han seguido las siguientes fases:

- Realización de un boceto
- Malla base
- *Sculpt Mode*
- Materiales texturizados

Los objetos realizados en Blender son los siguientes:

- Primera versión del renacuajo

  A partir del boceto de Figura 10, se modeló la primera versión del renacuajo mostrado en Figura 11. Para ello se ha editado una esfera con un modificador *Mirror* y un modificador de *Subsurf* junto con la técnica de *Sculpt Mode*. Se ha utilizado una textura modificada con el *software* Photoshop, superponiendo el archivo PNG que es una de las posibles caras del avatar realizadas por Cristina Esteban [5].

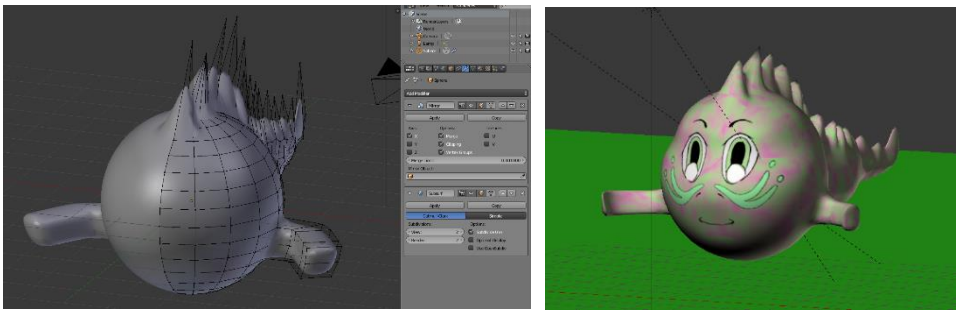

Figura 11. Primera versión del renacuajo

- Primera versión del personaje adulto

  De la misma forma que en el objeto “renacuajo”, se ha realizado el objeto “personaje” de Figura 12, utilizando las técnicas de *Mirror*, *Subsurf* y *Sculpt Mode*. “Renacuajo” y “personaje” han servido de base para que mi compañera Cristina Esteban realice la versión final de los mismos.

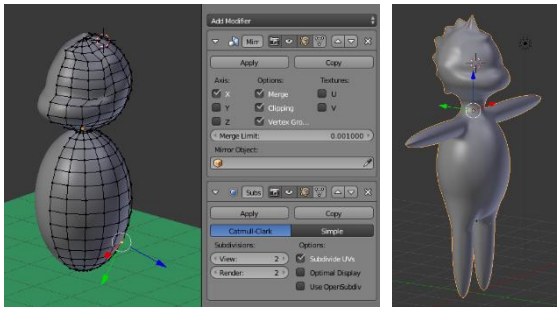

Figura 12. Primera versión del personaje adulto

- Rocas, algas y Plánctones

  Para completar los diversos escenarios es necesario realizar algunos objetos de decoración como rocas, árboles, Plancton, alguna casa, madera, suelo, lago, etc. Con las técnicas ya mencionadas se han obtenido los objetos de Figura 13.

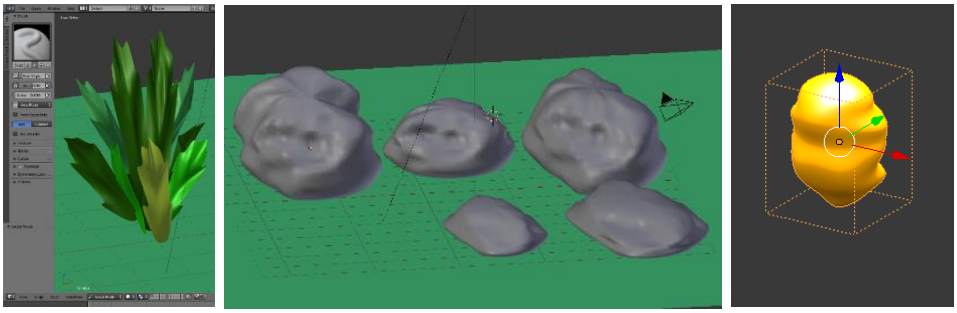

Figura 13. De derecha a izquierda: Algas, rocas y Plancton

- Diferentes versiones del escenario global

  Como se indica en el Anexo 1, cuando se quiere realizar el modelado de un objeto o conjunto de objetos de grandes dimensiones lo ideal es realizar una malla base, que tenga la forma simple de lo que se quiere realizar, y a partir de esta ir tallándola hasta conseguir la forma deseada. En Figura 14 se aprecia la primera versión o malla base del escenario del lago o suelo que se va a utilizar en el resto de escenarios.

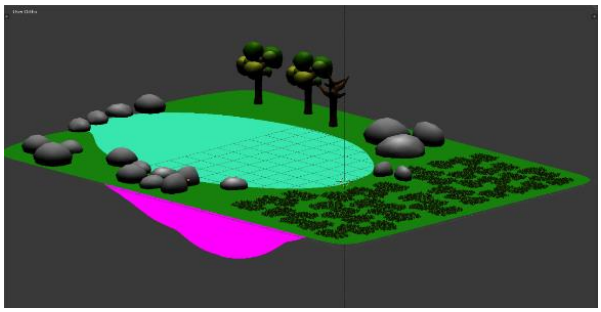

Figura 14. Primera versión del lago

En este "suelo" se incluyeron algunos objetos realizados por Cristina Esteban, como los árboles, las rocas antes mencionadas y el objeto "hierba" de prueba, que finalmente ha sido desechado.

A partir de esta base, utilizando un plano con un material de color marrón y la técnica *Sculpt Mode* y *Subsurf*, se obtiene el siguiente suelo (Figura 15) para la parte del lago.

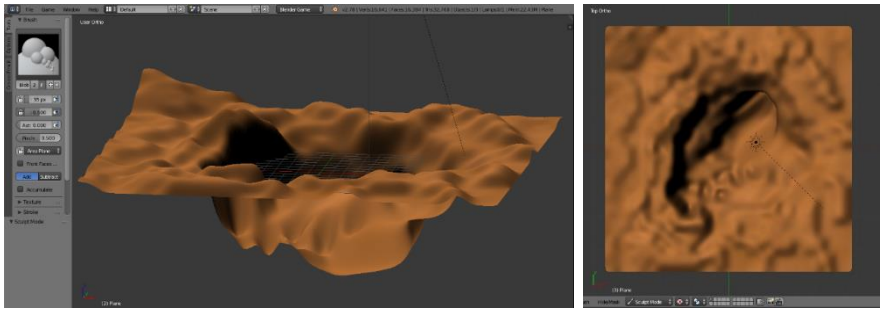

Figura 15. Suelo con Sculpt Mode *en vista de alzado y lateral*

El siguiente paso es añadir una textura para que el suelo sea más elaborado y que tenga aspecto terroso. Sin embargo, las texturas disponibles en el motor *Blender Render* y *Blender Game* son diferentes. En el primero existen diversas opciones de configuración sobre el material del objeto, en cambio en el segundo solo es posible añadir texturas en formato de imagen. Como se va a trabajar con el motor *Blender Game*, primero se tiene que elegir un material con textura en el modo *Blender Render,* como se muestra en Figura 16.

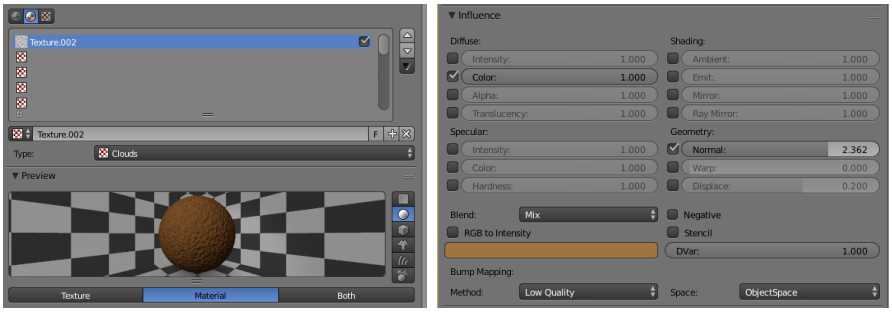

Figura 16. Parámetros para añadir una textura a un objeto en el motor Blender Render

En este caso se ha elegido el Tipo *Cloud*, que es un efecto granulado. Modificando el factor "Normal" de las opciones de Geometría, se puede influir en la apariencia de la sensación de profundidad de dicho efecto. Para el suelo se ha elegido un factor 2,362. Cuando se ha *renderizado* se ha generado una textura como la que aparece en Figura 17.

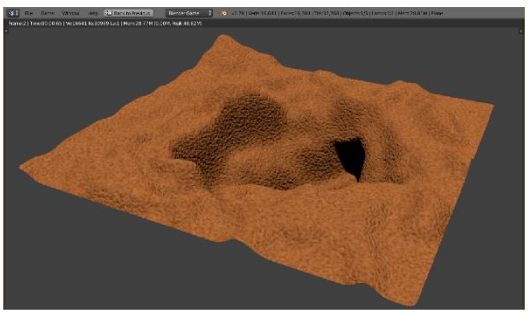

Figura 17. Suelo renderizado con textura

Una vez conseguida la textura deseada, utilizando la técnica *Render Baking* [14], se crea un mapa de la malla en una imagen 2D de la superficie renderizada en la vista que se desee. En este caso, para que la imagen no sea deformada se *renderiza* desde la vista de planta.

En Figura 18 (parte inferior), se muestra el resultado del material texturizado guardado en una imagen en formato png.

Para dar un aspecto más realista sin necesidad de modelar, en *Blender Game Engine* es posible añadir texturas y asociar partes de una imagen en 2D a las partes de un objeto en 3D. Gracias a esta posibilidad, teniendo la nueva textura del suelo, se puede asociar esta al objeto "suelo" en el videojuego. El editor que permite esto es el *UV Editing* para texturas.

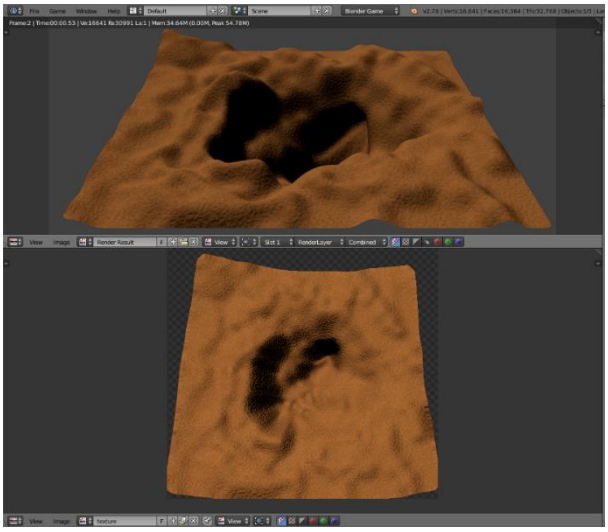

Figura 18. Baking del suelo

Siguiendo la misma técnica, se decide ampliar el terreno en la escena. Sin embargo, para esta segunda fase del modelaje del escenario se añade una textura de hierba.

En Figura 19 se puede apreciar la etapa de *UV Editing*, una vez se ha *renderizado* y realizado el *baking* de la nueva textura.

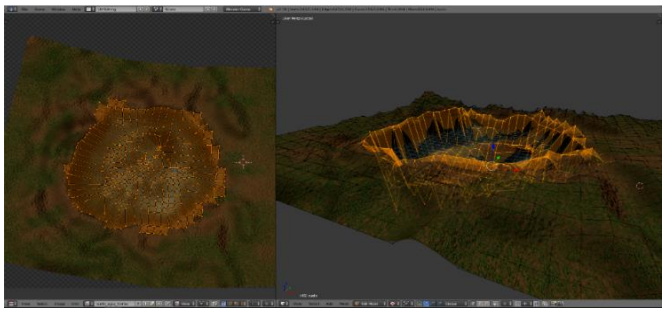

Figura 19.UV Editing para el fondo del lago.

Para completar el escenario base del juego se unifican los objetos modelados por Cristina Esteban [5] como los árboles con hojas, los árboles secos, los troncos y la cabaña.

También se le da textura al fondo del lago con un aspecto azulado para que parezca que el renacuajo se encuentra debajo del agua. Se añade también un plano que cubra el lago que tenga aspecto de agua. A este material se le añaden características para que sea traslúcido.

En Figura 20 se muestra el escenario desde diferentes puntos de vista. Para que la carga computacional de los mini-juegos sea menor se decide posteriormente dividir en cuatro partes para que correspondan a las distintas escenas.

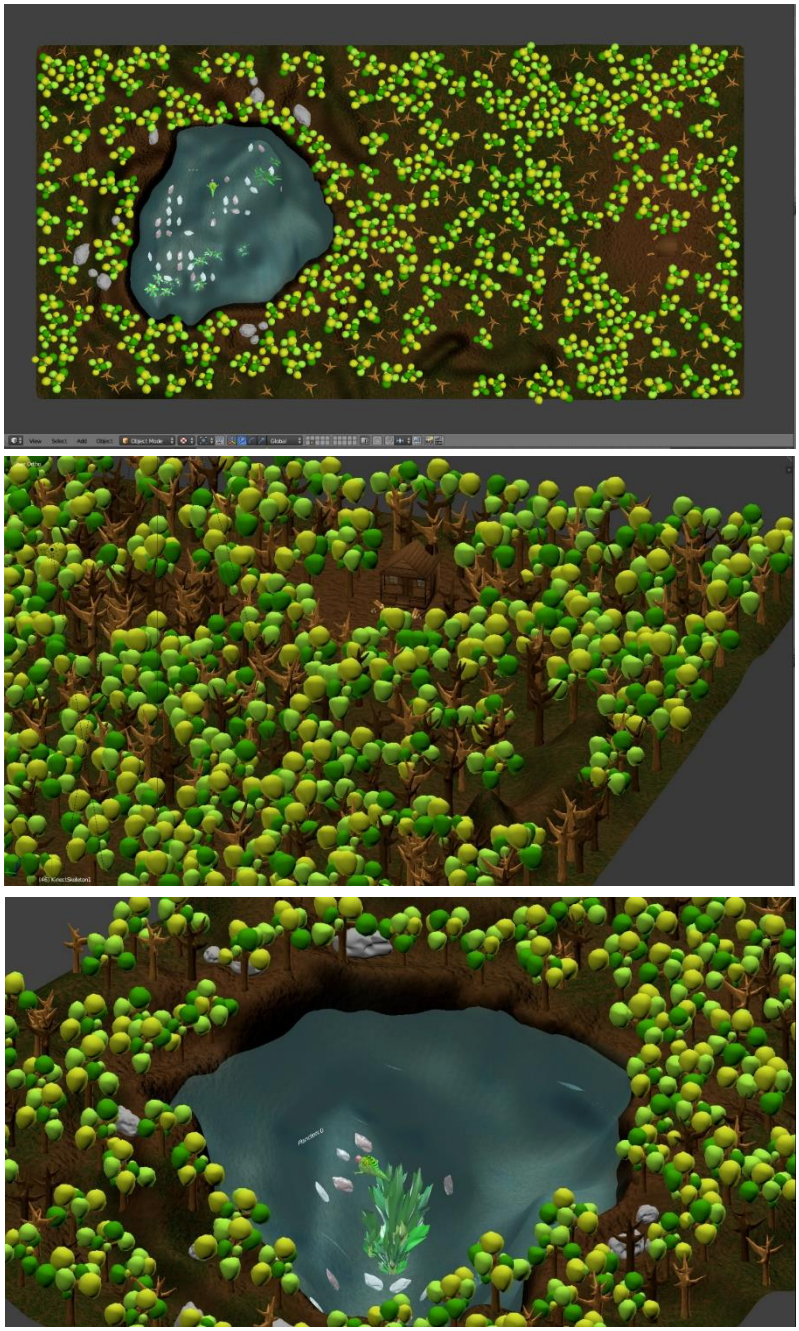

Figura 20. Diferentes vistas del escenario.

3.2. Phibys Adventures

En este apartado se explicará la adaptación e implementación de los mini-juegos: *Dive and Eat* y *Climb the Tree.*

En primer lugar, para ambos se ha instalado el Add-on de Ignacio Gómez-Martinho, el cual recibe los datos de posición del usuario gracias a Kinect y lo reproduce en 3D mediante esqueletos en el escenario deseado (Figura 21).

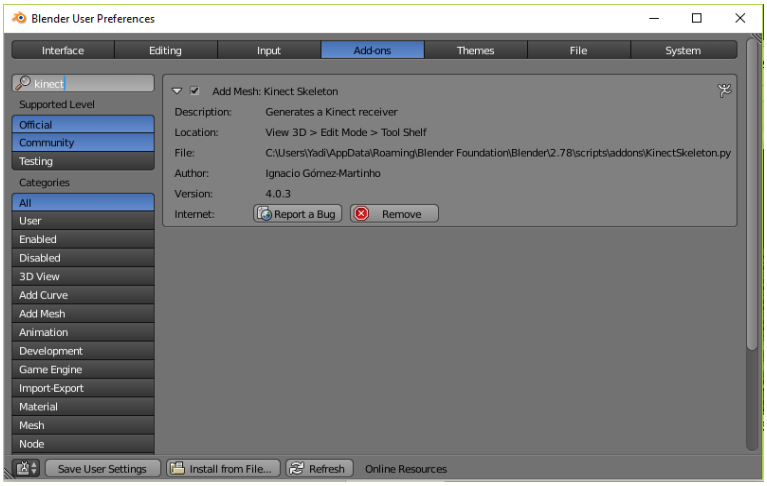

Figura 21.Add-on Kinect Skeleton

Siguiendo con el trabajo antecesor al proyecto, es necesario ejecutar el archivo *SaveUser* realizado por Ignacio Gomez-Martinho [1]. Cuando este se lanza aparecen en pantalla las esferas que se pueden ver en Figura 22, representando las posiciones de las articulaciones del usuario. El paciente ha de colocarse frente a la Kinect, mientras una cuenta atrás sucede, y ha de realizar todos los movimientos que pueda, llegando a su máximo rango. El fin de esto es conocer las restricciones motoras que posee y adecuar el juego a sus capacidades. Estos datos se guardan en un fichero con extensión ".pkl" en el directorio BodyFiles, que ha de estar en la raíz del juego.

Una vez se ha obtenido este fichero, se guarda para su posterior utilización en la configuración del mini-juego. En ambos juegos es necesario añadir desde las opciones del Add-on el esqueleto *Static*, que muestra los movimientos del usuario en todo momento.

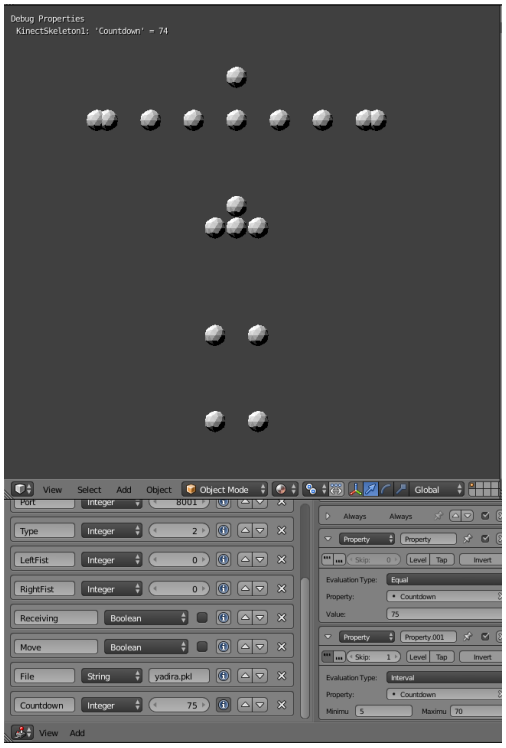

Figura 22. Ejecutable SaveUser

En la Figura 23 se muestra la pantalla de configuración del esqueleto *Static*. En este ejemplo, en el panel de propiedades se le asigna el nombre del fichero de configuración "yadira.pkl". Al iniciar el juego, los objetos amplificadores ajustan su posición de reposo y sus factores de multiplicación automáticamente [1].

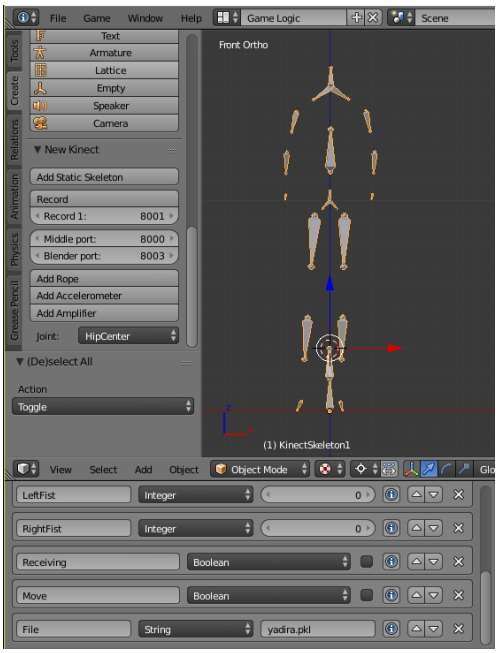

Figura 23.Interfaz del Add-on desde la que se añade el Esqueleto static

Para todos los ejercicios, se ha de hacer el *rigging* del avatar, asociar amplificadores a las articulaciones que se desee y crear esqueletos intermedios que generen los movimientos amplificados. Una vez realizada esta tarea, se prosigue con la lógica del juego.

Cristina Esteban [5] se ha encargado de implementar los mini-juegos *Chop the Wood* y *Row the Boat*. Para poder realizar estos ejercicios era necesario realizar el *rigging* del personaje evolucionado. Como indica el Anexo I, este acto consiste en añadir un esqueleto que permita deformar la malla en función de los movimientos realizados. En Figura 24 se puede ver el desarrollo de este punto.

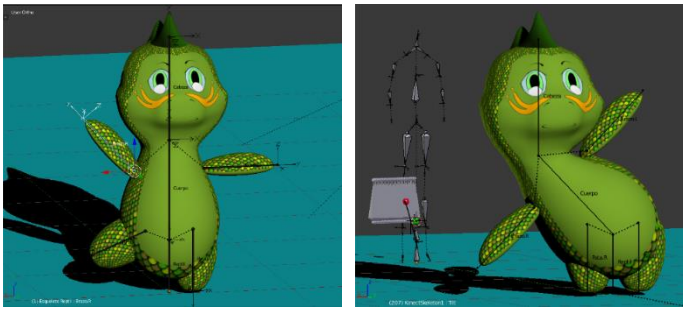

Figura 24. Rigging y amplificador del esqueleto del personaje evolucionado [5].

3.2.1. Dive and Eat

El escenario utilizado, mostrado en Figura 25, es la versión reducida del escenario global de Figura 20. Se escoge solo esta parte para ahorrar recursos, ya que el usuario únicamente verá el fondo del lago.

Figura 25. Escena principal del lago

Para este mini-juego se desea que el control del avatar se realice mediante el movimiento del tronco, por lo tanto, es necesario añadir un amplificador en esa parte del esqueleto.

En Figura 26, a la izquierda se aprecia el Esqueleto *Static*, en el que el hueso que representa el tronco corresponde con *HipCenterRot*. Este será el control del renacuajo. Para poder configurar su movimiento es necesario añadir un amplificador a este hueso. Este consiste en una armadura que se compone de 24 huesos superpuestos, los cuales se unen a una articulación o al tronco. Los huesos van copiando el movimiento real del Esqueleto *Static*, que es a su vez es el movimiento del usuario, pero con un *offset* que permite obtener una posición de referencia amplificada.

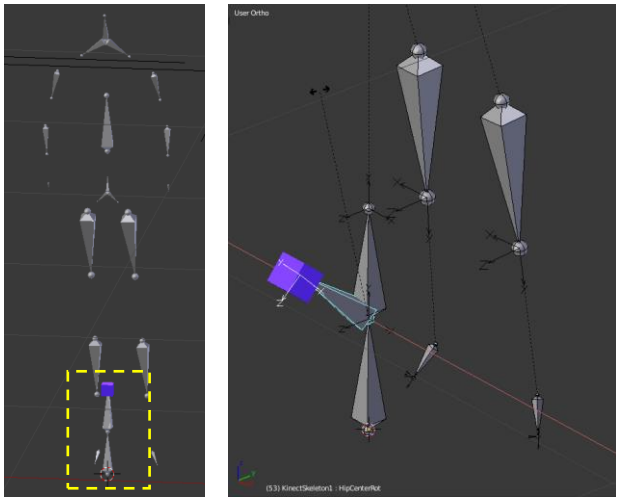

Figura 26. Hueso HipCenterRot en el esqueleto Static

Tomando como punto de referencia la cabeza del hueso *HipCenterRot* (señalado como un cubo morado en Figura 26), se añade un amplificador en dicho punto. Como éste solo da la posición, se requiere de la creación de un esqueleto intermedio que dé el movimiento amplificado. En Figura 27 se refleja el movimiento real y el amplificado del tronco (esfera roja). En la parte inferior se muestra el panel de *Bone Constraint*. En él se puede configurar el factor de los Amplificadores.

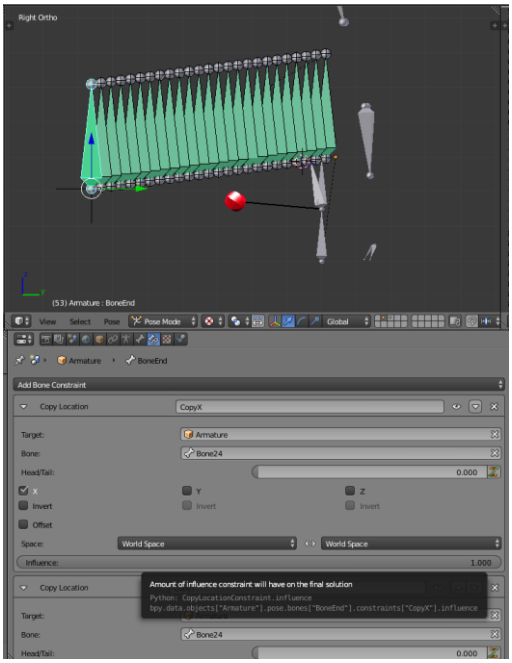

Figura 27. Amplificador del tronco.

Una vez disponible un esqueleto que reproduzca el movimiento que se quiere captar amplificado, se integra este movimiento en la lógica del juego.

Para detectar en qué dirección se mueve el paciente, se añaden cuatro objetos, dos esferas y dos cilindros. Los últimos tienen forma alargada para detectar el movimiento del tronco en un rango mayor. A cada objeto se le asocia una propiedad: esfera magenta “delante”, esfera azul “detrás”, objeto amarillo “izquierda” y objeto verde “derecha” (Figura 28).

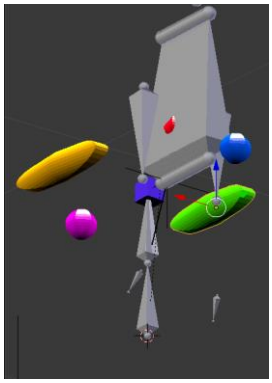

Figura 28. Objetos que detectan la dirección del tronco

Por otra parte, se tiene el personaje renacuajo, al que se le superpone un objeto que lo rodea con una malla adicional, denominado “Avatar” (Figura 29). Es el objeto que se usa para la detección de los movimientos y por lo tanto donde se aplica la mayoría de la lógica.

Avatar posee las propiedades “control” de tipo *Boolean*, “avatar” de tipo *Float*, “cantidadPlancton” de tipo *Integer,* “numTotal” de tipo *Integer* y “tiempo” de tipo *Timer*.

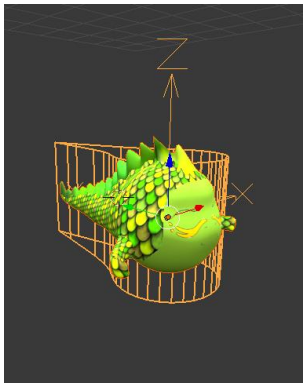

Figura 29. Avatar y la malla del renacuajo.

Otro objeto clave en este mini-juego es “Plancton”, el cual posee la propiedad “plancton”. Este es el objeto que tiene que buscar el usuario para avanzar en este nivel, como si de alimento se tratase.

Seleccionando “Avatar”, en el Editor Lógico se asignan varios Sensores, Controladores y Actuadores:

- Sensor *Always*

  Este sensor da una señal de salida en un intervalo regular. En este caso solo se desea que se lance al principio del ejecutable. Se encuentra conectado a un Controlador tipo “And”, que a su vez se conecta a un actuador “Motion”, “Sound” y “Scene” que superpone la escena “Interfaz”, como se ve en Figura 30. El primero genera un movimiento en el sentido negativo del eje Y local del Avatar. El segundo reproduce un sonido de ambiente en bucle durante todo el juego.

Figura 30.Escena Iterfaz sobre escena principal Scene

Cada vez que el mini-juego es ejecutado, Avatar se desplaza de forma constante hacia delante, ya que en este proyecto no se ha implementado ningún mecanismo para que el jugador lo haga avanzar.

- Sensor *Delay*

  Este sensor corresponde a una función retardo que se ha configurado para 10 niveles lógicos, donde 60 equivalen a 1 segundo. En este caso sirve para que se cargue la configuración inicial mediante un controlador tipo Python que lanza el módulo "nadar.cargar" del *script* Python "nadar.py".

- Sensor *Collision*

  Este sensor sirve para detectar, como su nombre indica, la colisión con otros objetos que posean la propiedad que indica su campo *Property*. Cada vez que "Avatar" colisiona con el objeto "Plancton" se lanza el controlador "nadar.actualiza".

- Sensor *Near*

  Este tipo de sensor se usa para detectar la cercanía del objeto a una propiedad que pertenece a otro objeto. Las propiedades que se detectan son "delante", "detras", "izquierda" y "derecha". Se utilizan 4 sensores de este tipo. Todos están conectados a controladores tipo "And", que a su vez están conectados a actuadores de tipo *Motion*. Cuando el usuario se incline hacia delante, el sensor *Near* detecta la propiedad "delante" y se activa un actuador que rota el Avatar en el eje local -X por 1 grado. Esto se traduce en un movimiento hacia adelante y hacia abajo del personaje. De forma contraria, si el usuario echa el tronco hacia detrás se activa otro sensor que rota el Avatar en el eje +X por 1 grado, moviéndose por lo tanto hacia detrás. Lo mismo sucede cuando se quiere ir a la derecha y a la izquierda, el Avatar rotará en el eje ±Z.

- Controlador Python ejecución *Module*

  Este controlador ejecuta *scripts* en Python cuando se activa un sensor conectado a él. El fichero del programa ha de ser añadido al Editor de Texto. El controlador del tipo *Module* ejecuta únicamente las funciones del *script* indicadas en su campo *Module name*.

  En este caso, existen dos controladores de Python que utilizan el *script* "nadar.py". El primero se lanza cuando comienza el juego con el Sensor *Delay* (nadar.cargar). Esta función comprueba si los valores iniciales ya están disponibles. En primer lugar, lee de un fichero con formato JSON el número total de planctons que ha de coger el usuario y lo guarda en la propiedad "numTotal".

El segundo corresponde a la detección de colisiones con los objetos Plancton (nadar.actualiza) que actualiza el valor del contador.

El valor de la propiedad “numTotal” se utiliza para colocar objetos Plancton de manera aleatoria en el lago para que cuando el paciente juegue no encuentre siempre los objetos en la misma posición, lo cual haría que perdiese el interés en el juego. Para ello se utiliza la biblioteca “random” de Python. Las posiciones ya vienen predefinidas por objetos Empty, numeradas del 1 al 16. En caso de que no se encontrase el fichero de configuración se inicializa el número total de Plánctones a encontrar a un valor por defecto. En este caso se ha asociado un valor de 8.

Por otra parte, cuando Avatar y Plancton colisionan se lanza la función “nadar.actualiza” y se reproduce un sonido que indica lo sucedido. En la función se incrementa la propiedad “cantidadPlancton” que hace de contador.

Si “cantidadPlancton” se igualase a la propiedad “numTotal” finalizaría el juego, reemplazando la escena principal por la escena “fin” (Figura 31).

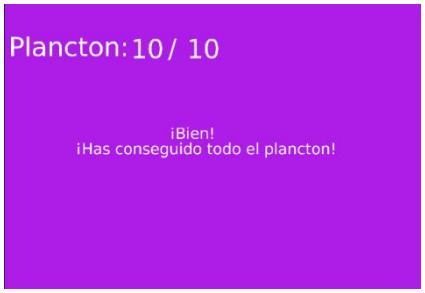

Figura 31. Escena fin

Cuando el juego termina se lanza la función “salvar()”. Esta función guarda en un fichero “.txt” con formato JSON la fecha, el tiempo que ha tardado el usuario en realizar el ejercicio, cuántos objetos Plancton ha recogido y el nombre del juego (ejercicio), como el ejemplo en Figura 32.


```
{
   "fecha":"2018-04-01 19:19:31",
   "duracion":5.333347797393799,
   "cantidadPlancton":2,
   "ejercicio":"Dive"
}
```


Figura 32. Fichero de salida de la función "salvar()"

A continuación, en Figura 33, se puede ver cómo quedan todas las conexiones citadas. A la izquierda los Sensores; Controladores en el medio; y Actuadores a la derecha.

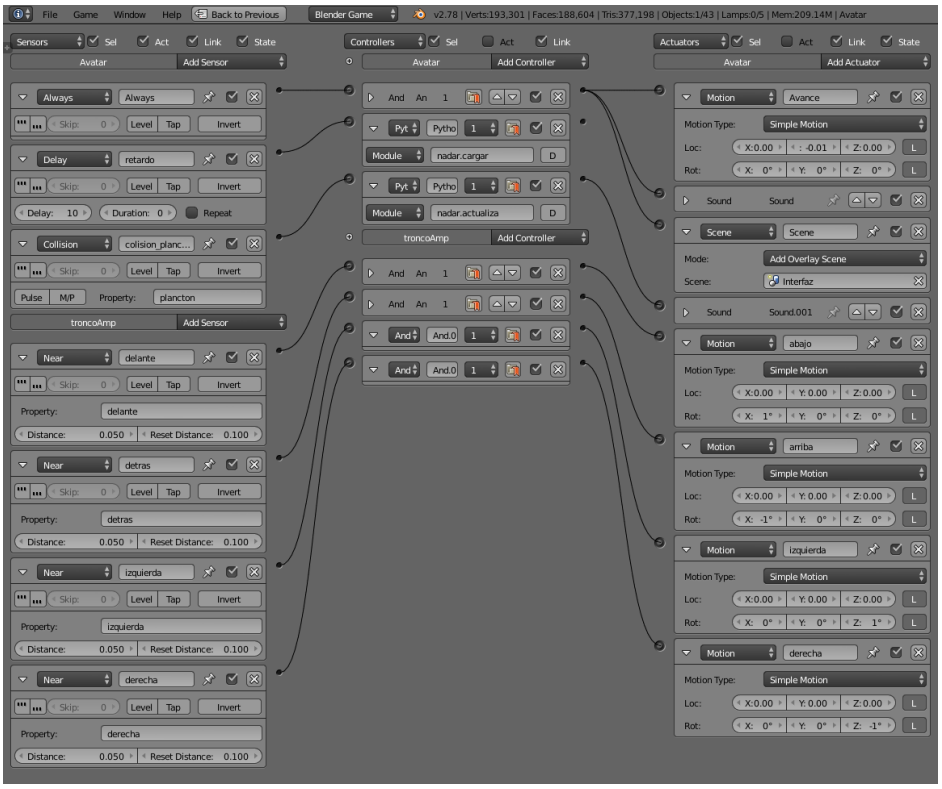

Figura 33. Sensores, Controladores y Actuadores de "Avatar"

Por último, una configuración necesaria es la modificación de la física general de la escena del lago [15], por la que se van a regir los objetos que en esta existan. Según el Editor de físicas que se elija (*Character, Soft Body, Rigid Body,* etc.), automáticamente los objetos se moverán en la dirección del eje Z negativo, simulando una gravedad. Dentro del lago no se desea que existe gravedad, con lo cual, como se ve en Figura 34, se ajusta el parámetro *Fall Speed Max*, que corresponde con la aceleración en el eje Z Global, a cero.

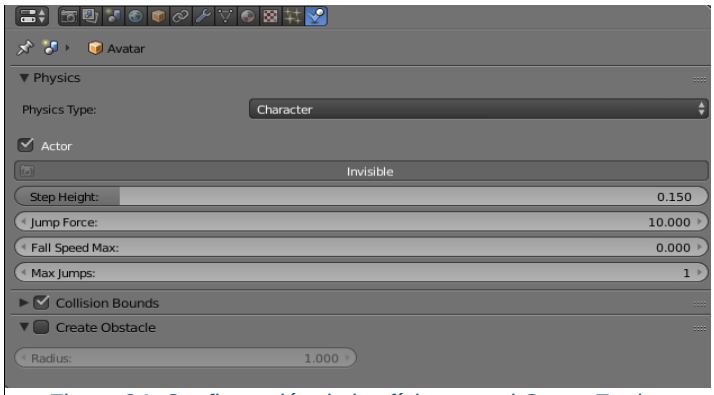

Figura 34. Configuración de las físicas en el Game Engine

3.2.2. Climb the Tree

Para la adaptación del mini-juego *Trepar*, se opta por un escenario sencillo que corresponde a un cuadrante de tierra con árboles, siendo uno de ellos más grande que el resto que será el objetivo a escalar (Figura 35).

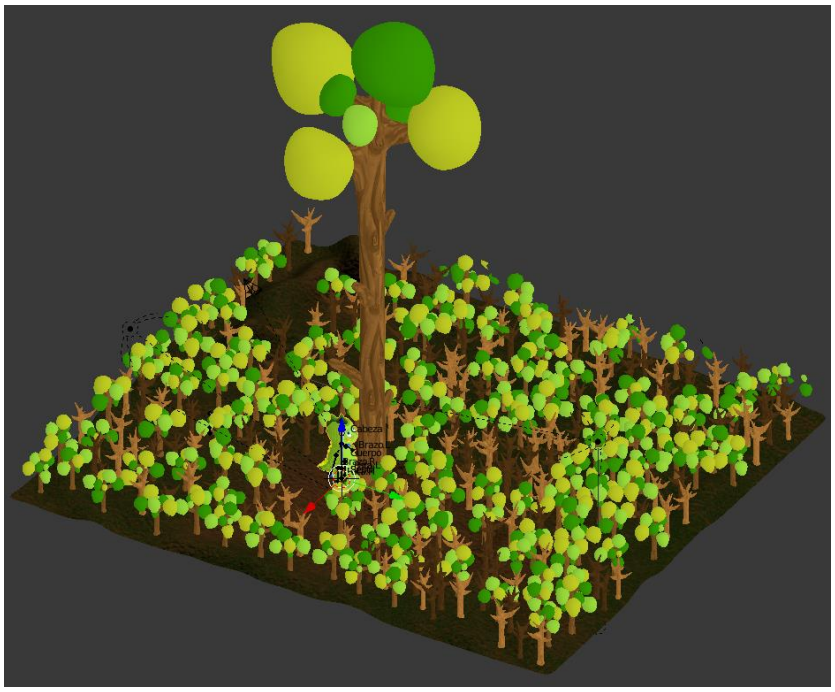

Figura 35. Escena principal del mini-juego Trepar.

En este caso, el personaje o avatar combina dos movimientos: desplazamiento vertical y el movimiento de los brazos. El primero es una simulación de que trepa el árbol, realizado mediante una animación del personaje. El movimiento de brazos, por otro, es la copia del movimiento del jugador amplificado.

Para su realización, se ha de hacer el *rigging* del personaje evolucionado de forma similar al ejercicio anterior. En Figura 36, en la parte superior izquierda (1), se pueden ver los dos huesos que corresponden a las manos, que son las partes del cuerpo del usuario que van a controlar el movimiento de los brazos del avatar. En cada uno de estos dos puntos se añade un amplificador, que corresponden con los esqueletos de la imagen superior izquierda (2).

Los amplificadores indican posición, por lo que hay que crear un esqueleto intermedio que genere el movimiento final. La configuración de estos esqueletos se realiza mediante el fichero con extensión ".pkl" como en el apartado de *Dive and Eat*. En Figura 36 parte inferior izquierda (3) se aprecia el esqueleto "brazosAmpli" que sigue el movimiento amplificado de la mano.

La copia de los movimientos de un esqueleto a otro es posible mediante las restricciones de huesos (*Bone Constraint*), en la opción *Copy Rotation* como se ve en la parte derecha de Figura 36 (4).

La lógica se ha realizado mediante una combinación entre bloques lógicos y un *script* de Python al igual que en el mini-juego anterior. También se utiliza un objeto cilíndrico "Avatar" que rodea la malla del personaje. En él se aloja la mayoría de la lógica.

Las propiedades que pertenecen a "Avatar" son: subir (*Boolean*), metros (*Integer*), movimientos (*Integer*), fin (*Boolean*), bloqueado (*Boolean*), temporizador (*Timer*) y tiempoMax (*Float*).

El mini-juego Trepar que antecede al que aquí se presenta posee mucha lógica que no resulta útil en esta versión, ya que el personaje utilizado en ese tenía una armadura compleja. Sin embargo, para detectar el movimiento del paciente solo es necesaria la detección del movimiento de las manos.

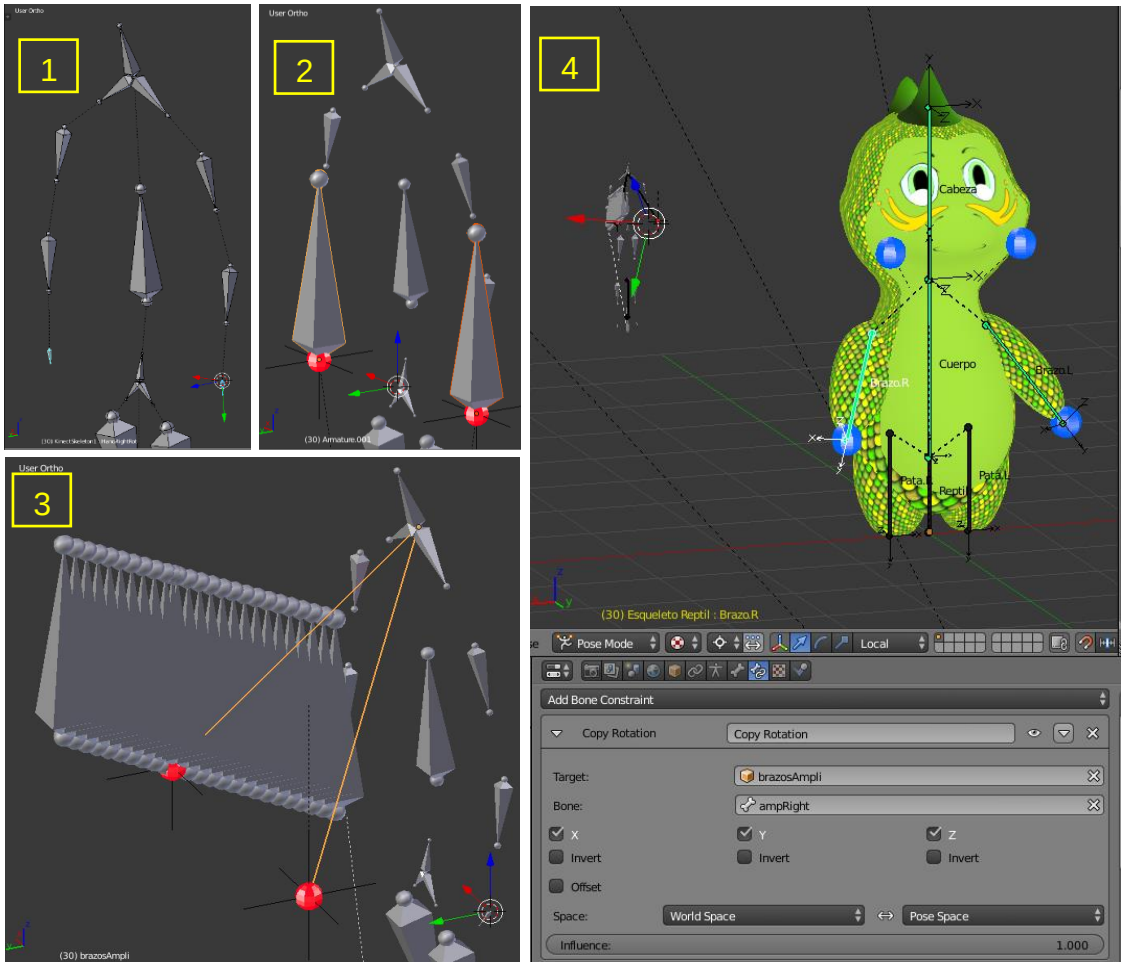

Figura 36. Izq: Esqueleto con amplificador y esqueleto que genera los movimientos amplificados. Derecha: Personaje con rigging

Es por esto que se rehace la lógica del ejercicio. Como se muestra en Figura 37, anclados a los extremos de la armadura que moldea el personaje se encuentran dos esferas azules. Cuando el usuario mueve los brazos, dichos objetos siguen el movimiento que este realice. En la parte delantera superior de Avatar hay otras dos esferas azules fijas. Cada una de las cuatro esferas posee una propiedad. Al acercase la que corresponde a cada mano con las que se encuentran a la altura de la cabeza se activa un sensor de cercanía que produce un cambio en las propiedades arriba listadas. Cuando estas propiedades se encuentran a *true*, se lanza un actuador de movimiento que hace que el sistema Avatar-malla se desplace verticalmente, proporcionalmente a la cantidad de movimientos realizados por el usuario.

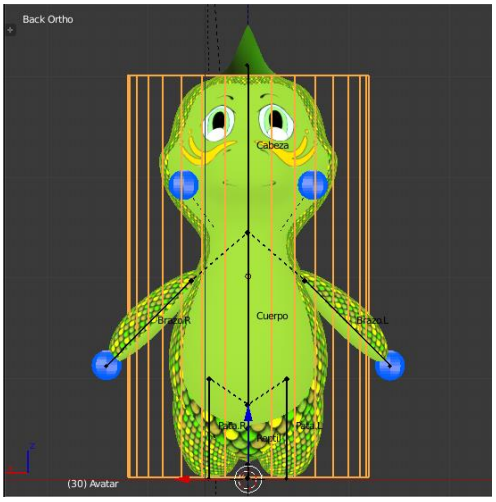

Figura 37.Configuración personaje Climb the tree

La conexión entre Sensores, Controladores y Actuadores son la siguientes:

- Sensor *Always*

  Al igual que en el mini-juego *Dive and Eat,* este sensor inicializa el juego superponiendo la escena Interfaz y reproduciendo un sonido de fondo en bucle.

- Sensor *Delay*

  Está configurado como el ejercicio anterior. Sirve para que se cargue la configuración inicial mediante un controlador tipo Python que lanza el módulo "trepar.cargar" del *script* Python "trepar.py".

- Sensor *Property*

  Existen cuatro sensores de este tipo. Con ellos se detecta el cambio de alguna propiedad. Si la propiedad "subir" vale *TRUE* o "metros" o "movimientos" cambia se lanza el controlador Python "trepar.actualiza". Si la propiedad "fin" es igual a *TRUE* se lanza el controlador "trepar.salvar".

- Controlador Python

  Como en el ejercicio anterior se inicia el juego con la función "init_module()" que inicializa las variables globales y asocia los objetos de la escena "Interfaz".

  La función cargar se lanza al principio con el sensor *Always*. En ella se lee del fichero de configuración en formato JSON y se asocia a la propiedad "metros" que es el número de metros que debe subir el jugador. Si esta carga inicial falla se establecen los valores por defecto.

  En la función "actualiza", el contador de la escena Interfaz se actualiza cuando la propiedad "metros" cambia.

  Por otra parte, las esferas que están unidas a las manos del personaje se identifican por "subirD_mano" y "subirI_mano". Cada una de ellas tiene una propiedad "mano". La lógica asociada a ambas esferas es un sensor *Delay* unido a un Actuador de tipo Mensaje que envía a todos los objetos el asunto "esperandoI" y "esperandoD" respectivamente.

Las esferas a la altura de los ojos tienen las propiedades “subirD” y “subirI”. Cuando las esferas de las manos se acercan a estas, detectan con un sensor *Near* sus propiedades y lanzan la función “trepar.subir”.

Esta función analiza si la propiedad “metros” es igual a 0. Si es el caso, cambia la variable “fin” a *TRUE*, lo que hará que un actuador reemplace la escena por la pantalla de fin. Además, cada vez que una esfera de mano se acerca a la del árbol, se le resta el valor 1 a la propiedad “metros” del objeto Avatar. Esta función solo permite contar una subida si se hace un movimiento de brazos alternativo.

Cuando la propiedad “subir” está a *TRUE* se lanza mediante un controlador “And” un actuador “Motion” que desplaza el Avatar y la malla 0.5 valores en el eje Z del mundo del juego.

El resultado final de este mini-juego se muestra en Figura 38, en el que se aprecia el personaje en una vista de tercera persona.

Figura 38. Escena desde el punto de vista de 3a persona.

3.3. Comunicación con la plataforma médica “Blexer-med”

Debido a que una de las innovaciones clave del actual proyecto es la adaptabilidad de los juegos en función de las capacidades del paciente, surge la necesidad de crear un módulo que permita esta gestión.

Por una parte, se busca la configuración de los parámetros de cada mini-juego, ya sea el número de Plánctones o metros a subir en el caso actual o cualquier parámetro necesario para juegos futuros.

Además, se quiere almacenar los resultados obtenidos de cada paciente para tener una visión de su histórico, así como un guardado del punto en el que se quedó en la partida o de si la mejora es tan considerable que haya necesidad de reajustar parámetros.

Por otro lado, se desea que todos los juegos que integren el sistema Blexer sean jugables desde casa. Se requiere de una BBDD que registre los pacientes, terapeutas y hospitales a los que pertenecen para poder tener un control total del paciente.

Por todo esto, en paralelo al proyecto actual, Mónica Jiménez [2] ha trabajado en estos requerimientos, desarrollando la plataforma "Blexer-med". Se trata de una plataforma web basada en un Web Service, al que podrán acceder los terapeutas registrados tanto desde el navegador o el middleware Chiro que ha sido modificado para que los usuarios se autentifiquen.

Para que el sistema global funcione es necesario que exista una comunicación entre el juego en Blender y el *middleware* Chiro. De esta forma, cuando el usuario se registrase conectaría con el servidor y descargaría sus datos necesarios para la configuración en formato JSON. Cuando el paciente terminase su partida, se guardaría un fichero en formato JSON con los resultados obtenidos, para ser almacenados en la BBDD.

En los mini-juegos creados, en el Editor Lógico del motor de juegos, se ha realizado una conexión de prueba para testear el funcionamiento de esta comunicación. El formato escogido para codificar la información es JSON (*JavaScript Object Notation*) [16].

En primer lugar, con la ayuda de un editor online de JSON, se ha obtenido un fichero ".txt" con el formato que se muestra en Figura 39.

Por otra parte, se ha creado un script que utiliza la librería "json" de Python para decodificar este tipo de formato y así acceder a sus campos y asociarlo a alguna de las propiedades que se utilizan en los objetos del mini-juego.

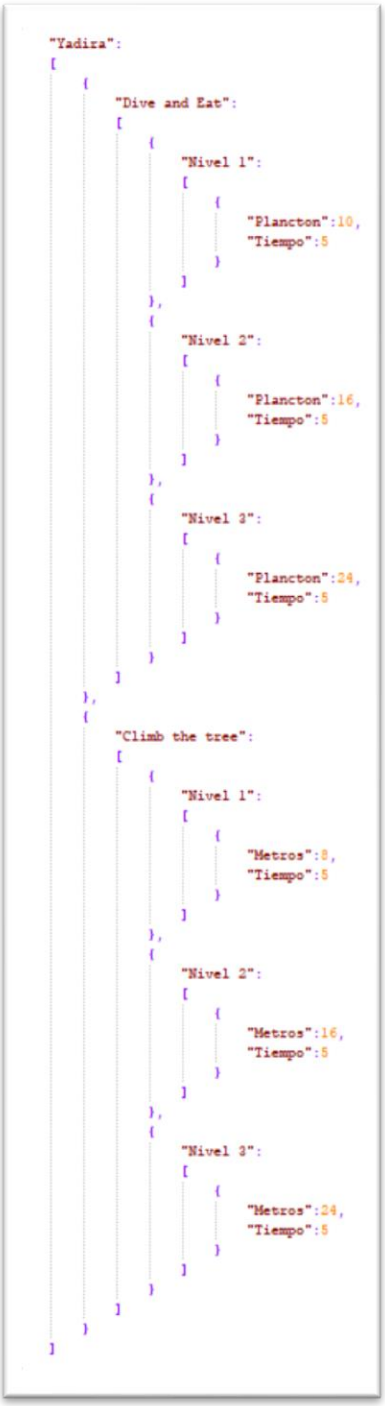

Figura 39. Archivo de configuración en formato JSON

En los mini-juegos *Dive And Eat* y *Climb The Tree* se llama a la función "cargar()" en la que se comprueba si, como indica su nombre, se han cargado las variables que van a ser utilizadas por el juego. A continuación se indican los algoritmos para estas funciones:

- *Dive And Eat*

```
#Cargar variables del fichero de texto
Función cargar:

  #comprobación de si se ha inicializado el objeto
            test = interfaz.objects["Contador"]

  #Si no, se controla con una excepción y se lanza la función que inicializa las escenas
            init_module()

  Con el fichero de configuración abierto "configuracion_Yadira.txt" como json_file:

            #leer el fichero
            data = deserializar(json_file)

            #Se añaden los planctons en posiciones aleatorias
            objprop["numTotal"] = data["Yadira"][0]["Dive and Eat"][0]["Nivel 1"][0]["Plancton"]

            #Se añade el actuador de tipo Edit Object Add y se añaden en las posiciones aleatorias
            add=logic.getCurrentScene().addObject
            for i in range(objprop["numTotal"]):
                      aleatorio=r.randint(1,16)
                      print(str(aleatorio))
                      add("plancton",str(aleatorio))

  #Si no existe el fichero se controla con una excepción y se inicializan los valores por defecto y se guarda
            init_data()
            salvar()

  #Se da valor al contador de la interfaz
  TextoCantidad["Text"]=objprop["numTotal"]
```

- *Climb The Tree*

```
#Cargar variables del fichero de texto
Función cargar:

  #comprobación de si se ha inicializado el objeto
            test = interfaz.objects["Contador"]

  #Si no, se controla con una excepción y se lanza la función que inicializa las escenas
            init_module()

  Con el fichero de configuración abierto "configuracion_Yadira.txt" como json_file:

            #leer el fichero
            data = deserializar(json_file)

            #Se añaden los metros que tiene que escalar el personaje
            avatar["metros"]=data["Yadira"][1]["Climb the tree"][0]["Nivel 1"][0]["Metros"]
            TextoContador["Text"]=avatar["metros"]
            avatar["tiempoMax"]=data["Yadira"][1]["Climb the tree"][0]["Nivel 1"][0]["Tiempo"]

  #Si no existe el fichero se controla con una excepción y se inicializan los valores por defecto y se guarda
            init_data()
            salvar()
```

De la misma forma, cuando el juego finaliza se guarda en un fichero los resultados en formato JSON:

- *Dive And Eat*

```
#Salvar variables en fichero de texto
Función salvar:

            Se abre el fichero de resultados:
                      archivo = open(nombrefichero, "w")

            #Se crea un array con los valores a guardar
            newarray = {}
            newarray['fecha'] = time.strftime("%Y-%m-%d %H:%M:%S")
            newarray['ejercicio'] = "Dive"
            newarray['duracion'] = objprop["tiempo"]
            newarray['cantidadPlancton'] = objprop["cantidadPlancton"]

            data_string = serializar en formato JSON array (newarray)

            Se cierra el fichero de resultados.
```

- *Climb The Tree*

```
#Salvar variables en fichero de texto
Función salvar:

          Se abre el fichero de resultados:
                    archivo = open(nombrefichero, "w")

          #Se crea un array con los valores a guardar


          newarray = {}
          newarray['fecha'] = time.strftime("%Y-%m-%d %H:%M:%S")
          newarray['ejercicio'] = "Climb"
          newarray['duracion'] = avatar["temporizador"]
          newarray['movimientos'] = avatar["movimientos"]

          data_string = serializar en formato JSON array (newarray)
          #Lanzo el actuador que tengo conectado al Controlador
          controlador = logic.getCurrentController()
          if int(avatar["metros"]) == 0:
                    escena = logic.getCurrentScene()
                    Se sustituye la escena del juego por la escena "fin"
```

En función del esquema elegido por Mónica Jiménez [2] para guardar los parámetros necesarios para cada mini-juego, se puede modificar el código para asociarlo a las variables correspondientes.

4. Pruebas realizadas

Como se ha indicado en los antecedentes de este proyecto, ya se habían realizado pruebas con la primera versión de los mini-juegos. Una vez adaptados al nuevo escenario y personajes, se decide hacer una segunda prueba del proyecto desarrollado, de forma conjunta con la parte realizada por Cristina Esteban [5].

El 12 de Mayo de 2017 se concertó una cita con Maite Manzano, integrante del Servicio de Fisioterapia de la asociación ASEM (Asociación Madrileña de Enfermedades Neuromusculares).

En este encuentro se quiso enseñar a Maite la evolución del proyecto. Probó las versiones anteriores de los mini-juegos comparándolas con las actuales. En Figura 40 se muestra el estado de los cuatro mini-juegos cuando se realizaron estas pruebas.

En cuanto a la parte que corresponde a este proyecto, se detectó que el manejo del renacuajo en el juego *Dive and Eat*, era a través de movimientos muy exagerados, por lo que fue necesario ajustar la posición de los objetos que detectaban el movimiento del tronco.

Desde el punto de vista de la fisioterapia, para los pacientes con movilidad reducida con los que trabaja Maite, era más interesante los movimientos en los que el paciente tiene que mantener la posición del miembro fijo. Este punto era especialmente importante para el mini-juego desarrollado por Cristina, *Chop the Wood.*

Figura 40. Última versión de los cuatro mini-juegos: Arriba: izq: Row The Boat; derech: Climb The Tree; Abajo: izq: Chop The Wood; derecha: Dive And Eat

A finales de dicho mes, se realizó otra prueba con voluntarios de la asignatura SAI (Síntesis y Animación de Imágenes). Estos jóvenes no tenían ninguna restricción física. Todos ellos probaron los cuatro mini-juegos, y al igual que en la prueba anterior, indicaron que el manejo del renacuajo exigía demasiada

inclinación para controlarlo. También se comentó la velocidad de movimiento del renacuajo y se propuso que debería ir más rápido.

A partir de estas reuniones se introdujo como mejora añadir objetos Plancton de manera aleatoria en el mapa del lago, cosa que ha quedado plasmada en el capítulo anterior.

También fue necesario añadir un sonido que se reprodujera cada vez que se cogiera un Plancton, así como retrasar el seguimiento de la cámara para apreciar mejor el juego en 3ª persona.

5. Conclusiones

En el actual trabajo se presenta una parte del sistema Blexer, el primer prototipo de juego para propósitos terapéuticos para personas con movilidad reducida, más en concreto para personas en silla de ruedas con enfermedades neurológicas que provocan debilidad muscular. Se trata de la primera versión que servirá para poder testear los posibles errores que tiene el sistema, para ser posteriormente integrado en un juego completo.

- Se ha conseguido crear una historia junto con Cristina Esteban, modelar un entorno global con el *software* Blender que será la base de cualquier nueva escena que se quiera integrar en el futuro juego.
- Se ha creado el personaje principal, en dos de sus evoluciones, renacuajo y adulto, lo que ha permitido crear escenas separadas temporalmente en la historia.
- En los dos avatares se integra la recepción de Kinect, permitiendo alternar dos movimientos en el juego: movimientos copiados y de control. Los movimientos copiados permiten añadir Amplificadores de movimiento en caso de que el paciente tenga limitaciones físicas. Los movimientos de control permiten el avance de los avatares sin necesidad de desplazarse frente a la Kinect.
- Se han transformado dos de los mini-juegos antecedentes al actual proyecto, con el nuevo personaje y con una lógica adaptada y mejorada para el nuevo entorno.
- Se han implementado nuevos mecanismos de juego, como retos o puntuaciones, con el fin de motivar al usuario.
- Se ha conseguido leer y exportar ficheros con datos de los juegos, como tiempo empleado o número de aciertos. Estos ficheros serán utilizados para el proyecto “Blexer-med” que permitirá la gestión del futuro juego completo.

6. Trabajo futuro

Aunque no se trata de trabajo futuro propiamente, tras este proyecto, Ignacio Gómez-Martinho González [1] comenzó la tarea de integrar los cuatro mini-juegos en un juego completo. En esta segunda versión de *Phibys Adventures* se tiene un ejecutable en el que el usuario ha de *logearse* para poder descargar su configuración. Esta configuración es realizada por su terapeuta o médico, el cual tiene acceso a la plataforma Web "Blexer-med". En ella se registran las sesiones jugadas por el paciente y los juegos a los que tiene acceso, con la posibilidad de configurar los parámetros. Si el jugador no tiene una cuenta puede descargar la configuración por defecto.

Por otro lado, en cuanto al juego, tiene un mapa que va desbloqueando zonas según se avanza en el juego realizando los ejercicios, lo que motivará al paciente para seguir jugando. Cada celda posee entre 2 y 3 ejercicios. En esta última versión se han añadido mecánicas de juego como conseguir madera o vidas [17].

Este trabajo está en fase de testeo por un gran número de voluntarios. Los resultados de esto ayudarán a la mejora del sistema.

Por otro lado, en paralelo, en el mismo grupo de investigación en el CITSEM, se pretende la migración de Kinect XBOX a la Kinect One. Esta nueva versión de Kinect 2013, genera la posibilidad de mejorar el reconocimiento de la rotación de las articulaciones y la precisión en la detección de ángulos. Posee un *hardware* mejorado, mayor resolución (1080p) y mejor respuesta en tiempo. Estas mejoras podrían dar la posibilidad de añadir características al juego como detección de la fuerza del músculo [18] o monitorización cardiaca, al igual que una representación más precisa del esqueleto humano. Al igual que se cambia la versión de Kinect, se reutilizarán los objetos modelados en Blender para ser utilizados en el motor de juegos Unity, más potente que *Blender Game Engine*.

Como futuras mejoras, se podría implementar un tutorial de movimientos que se van a realizar a lo largo del juego, para que se practique su correcta actuación.

Lo ideal sería que el juego tuviera inteligencia artificial para que se adapte al estado actual del jugador y optimice su rendimiento. Que alerte al jugador con sonidos o visualmente de posturas incorrectas y que se reporte al terapeuta. También sería interesante añadir una opción multijugador, en modo servidor. Para diferentes ejercicios el profesional debería poder ajustar las articulaciones que deben funcionar, las partes del cuerpo, el máximo y el mínimo ángulos necesarios para el paciente y el plan para la ejecución del movimiento.

Hace unos meses se presentó una necesidad más importante que todas estas, en octubre de 2017 Microsoft hizo oficial que dejaba de fabricar su sistema Kinect para XBOX. Sin embargo, en este mes se ha anunciado la aparición de Project Kinect para Azure [19], el servicio de la nube de Microsoft. Este dispositivo estará orientado para la utilización en investigación y no en consolas como solía ser.

7. Referencias

[1] I. Gómez-Martinho González, Desarrollo e implementación de middleware entre Blender, Kinect y otros dispositivos, Madrid: ETSIST, 2016.

[2] M. J. Ramos, Plataforma Médica para el entorno de Videojuego Terapéutico "Blexer", Madrid: Universidad Politécnica de Madrid, 2017.

[3] P. Parra Iglesia, Memoria de práctica externa, Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, UPM, Madrid, 2016.

[4] M. Eckert, I. Gómez-Martinho, J. Meneses y J. F. Martínez, «New Approaches to Exciting Exergame-Experiences for People with Motor Function Impairments,» [En línea]. Available: http://www.mdpi.com/1424-8220/17/2/354.

[5] C. Esteban, Diseño e implementación del entorno de videojuego serio de rehabilitación del proyecto “Blexer”, Madrid: Universidad Politécnica de Madrid, 2017.

[6] «Virtual Reality Kinect Rehabilitation,» Virtual Reality, 2018. [En línea]. Available: http://www.virtual-reality-rehabilitation.com/products/seeme/what-is-seeme. [Último acceso: 1 Marzo 2018].

[7] «KineLabs: Rehabilitation Games for Elderly and Stroke Patients using Kinect,» 2012. [En línea]. Available: https://www.youtube.com/watch?v=2bePNpyuP5k. [Último acceso: 1 Marzo 2018].

[8] A. Fernández-Baena, A. Susín y X. Lligadas, «Biomechanical Validation of Upper-body and Lower-body Joint Movements of Kinect Motion Capture Data for Rehabilitation Treatments,» de *Fourth International Conference on Intelligent Networking and Collaborative Systems*, Barcelona, 2012.

[9] Mira, «Mira rehab,» 2017. [En línea]. Available: http://www.mirarehab.com/. [Último acceso: 04 03 2018].

[10] W. Zhao, H. Feng, R. Lun y D. D. E. y. M. A. Reinthal, «A Kinect-based rehabilitation exercise monitoring and guidance system,» de *IEEE 5th International Conference on Software Engineering and Service Science*, Beijing, 2014.

[11] B. A. Valdés, C. G. E. Hilderman, C. T. Hung y N. S. y. H. F. M. V. d. Loos, «Usability testing of gaming and social media applications for stroke and cerebral palsy upper limb rehabilitation,» de *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Chicago, 2014.

[12] «Python,» 2018. [En línea]. Available: https://pypi.python.org/pypi/python-osc. [Último acceso: 04 03 2018].

[13] Y.-K. Chou, Actionable Gamificacion. Beyond Points, Badges and Leaderboards, Octalysis Media, Abril 2015.

[14] J. C. López, Baking en Blender, Madrid: Universidad Politécnica de Madrid, 2016.

[15] «Wiki Blender,» [En línea]. Available: https://wiki.blender.org/index.php/Doc:ES/2.4/Manual/Game_Engine/Physics. [Último acceso: 17 02 2018].

[16] «ECMA-404 The JSON Data Interchange Standard.,» [En línea]. Available: https://json.org/json-es.html. [Último acceso: 16 05 2018].

[17] M. Eckert, I. Gomez-Martinho, C. Estéban, Y. Peláez, J. Meneses y L. Salgado, «Blexer – Full Play Therapeutic Blender Exergames for People with Physical Impairments,» de *3rd EAI International Conference on Smart Objects and Technologies for Social Good*, Pisa, Italia, Noviembre 2017.

[18] D. Stanev y K. Moustakas, «Virtual Human Behavioural Profile Extraction Using Kinect Based Motion Tracking,» de *International Conference on Cyberworlds*, Santander, 2014.

[19] «SOMOSXBOX,» [En línea]. Available: https://www.somosxbox.com/kinect-para-azure-la-nueva-innovacion-de-microsoft/779835. [Último acceso: 20 05 2018].

[20] Microsoft, «Tracking Users with Kinect Skeletal Tracking,» 2013. [En línea]. Available: https://msdn.microsoft.com/en-us/library/jj131025.aspx. [Último acceso: 04 03 2018].

8. Bibliografía

- M. Eckert, I. Gómez-Martinho y J. M. y. J. F. M. Ortega, «A modular middleware approach for exergaming,» de International Conference on Consumer Electronics, Berlín, 2016.
- M. Eckert, I. Gómez-Martinho y J. M. y. J. F. M. Ortega, «A multi functional plug-in for exergames,» de International Symposium on Consumer Electronics (ISCE), Madrid, 2015.
- R. C. Menezes, P. K. A. Batista, A. Q. Ramos y A. F. C. Medeiros, «Development of a complete game based system for physical therapy with kinect,» de 2014 IEEE 3nd International Conference on Serious Games and Applications for Health (SeGAH), Rio de Janeiro, 2014.
- M. Eckert, I. Gómez-Martinho, J. Meneses y J.-F. Martínez, «New Approaches to Exciting Exergame-Experiences,» State-of-the-Art Sensors Technology in Spain, 12 febrero 2017.
- C.-J. Su, «Personal Rehabilitation Exercise Assistant with Kinect and Dynamic Time Warping,» International Journal of Information and Education Technology, vol. 3, nº 4, pp. 448-454, 2013.
- I. Gómez-Martinho González, Memoria final de "Practicas Externas en CITSEM", Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación, UPM, Madrid, 2015.

Anexo I: Glosario

- Actuador (Actuator)

  Bloque lógico que actúa como un músculo. Puede mover objetos o reproducir sonido.

- Esqueleto (Armature)

  Objeto constituido por huesos. Usado para manipular personaje, propiedades, etc.

- Baking

  Proceso de computación y almacenaje de resultados que potencialmente consumirán bastante tiempo de cálculo para evitar la necesidad de realizarlo nuevamente.

- Bloque lógico (Logic Brick)

  Representación gráfica de una unidad funcional de la lógica del motor de juegos de Blender. Un bloque lógico puede ser un Sensor, Controlador o Actuador.

- Constraint (Restricción)

  Manera de controlar un objeto con datos de otro.

- Controlador (Controller)

  Bloque lógico que actúa como el cerebro. Toma decisiones de los "músculos" activos (actuadores), usando lógica simple o compleja mediante scripts de Python.

- Empty (Vacío)

  Objeto sin vértices, aristas o caras.

- Hijo

  Objeto que es afectado por su padre.

- Inverse Kinematic (Cinemática Inversa)

  Proceso de determinación de los movimientos a partir de la conexión entre segmentos de un cuerpo o modelo. Usando IK en una estructura jerárquica se puede, por ejemplo, mover el hombro para que produzca un movimiento encadenado en todo el brazo hasta la mano. Sin IK la mano no se movería.

- Malla (Mesh)

  Tipo de objeto consistente de vértices, aristas y caras.

- Render (Representación)

  Proceso computacional de generar una imagen 2D a partir de geometría 3D

- Rig

  El acto de construir un sistema de relaciones que determina cómo algo se mueve, por ejemplo, crear un esqueleto para dar movimiento a una malla.

- Sensor

  Bloque lógico que actúa como un sentido. Reacciona a ser tocado, visto, golpeado, etc.

- Subdividir

  Técnica para añadir más geometría a una malla. Crea nuevos vértices en las aristas, nuevas aristas entre subdivisiones y nuevas caras a partir de las nuevas aristas.

- Textura

  Especifica los patrones visuales en las superficies y simula la estructura física de una superficie.

  Transformar

  La idea de combinar localización, rotación y escalado en un objeto.

- UV Map

  Define una relación entre la superficie de una malla y una textura 2D. En detalle cada cara de la malla es mapeada para corresponder a una parte de la textura.

Anexo II: Blender

El Motor de Juegos de Blender (Blender Gamen Engine o BGE) supervisa o monitorea un juego en bucle, cuya lógica, sonido, físicas y *rendering* es simulado en orden secuencial.

El usuario puede acceder al Editor Lógico, el cuál proporciona una interacción de simulación y funcionalidad compleja.

Los pasos para crear una simulación con el BGE son:

- Crear elementos visuales para poder ser renderizados. Esto puede ser en modelos 3D o imágenes.
- Habilitar interacciones entre objetos y escenas usando bloques lógicos o scripts personalizados y determinar cómo se desenvuelven (con los sensores apropiados como el teclado o el ratón).
- Crear una o más cámaras para dar un punto de vista desde el que renderizar la escena, y modificar los parámetros para manipular el ambiente en el cuál el juego se desarrolla.
- Lanzar el juego, usando el ejecutable interno, depurarlo y posteriormente exportarlo para que sea jugable en la plataforma correspondiente.

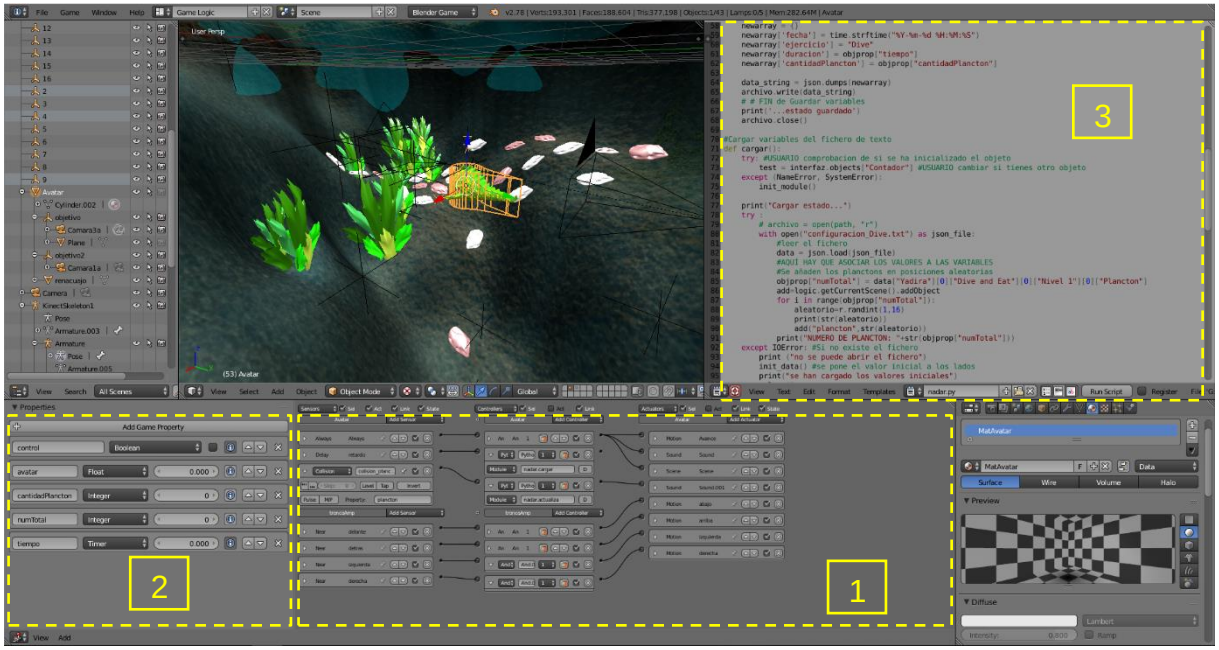

Figura 41. Blender Game Engine en Modo Editor Lógico

En Figura 41 se muestra la pantalla que corresponde con el Editor Lógico. En la sección 1, el panel de Editor Lógico es dónde se establece la lógica, propiedades y estados para controlar el comportamiento de los diferentes objetos en el juego. En el indicador 2 se encuentra el Panel de propiedades. Las Propiedades equivalen a las variables en programación. Son almacenadas en los objetos, y pueden ser usadas para representar características de estos. El punto 3 corresponde con el Editor de Texto. En él se pueden realizar scripts de Python para personalizar la lógica del videojuego sin necesidad de utilizar los bloques lógicos, o combinando estos con funcionalidades más complejas.

Algunas de las técnicas de Blender utilizadas en este proyecto son:

- Sculpt Mode

  Es un recurso indispensable en la creación de personajes con Blender 3D. Este modo de trabajo posibilita, entre otras cosas, la creación de superficies extremadamente complejas y con un nivel de detalles que difícilmente podría ser superado por el modelado tradicional. Una de las cosas más importantes a considerar, es la planificación y la preparación de una buena "base" para esculpir, haciendo posible alcanzar un nivel razonable de detalles al final del proceso. Para esto es necesario modelar un bloque en baja resolución, que tenga la forma y los detalles principales de lo que se va a tallar.

- *Render Backing*

  Baking es un término usado frecuentemente en computación gráfica o gráficos por ordenador, en relación con desarrollo de juegos. Hace referencia a la técnica de pre-calcular algo con el fin de acelerar algunos procesos. El *Rendering* toma mucho tiempo dependiendo de las opciones que se elijan. Es por ello que Blender permite "hornear" algunas partes, de forma que cuando ejecutes el Blender Render, la escena entera es renderizada mucho más rápida, ya que los colores de los objetos con *baking* no tienen que ser recalculados.

  Cuando esta técnica es aplicada se crea un mapa de la malla en una imagen 2D de la superficie renderizada.

- *UV Editing* para texturas

  Las texturas son como capas adicionales sobre un material base. Estas afectan en uno o más aspectos de la red de color de un objeto.

  Este editor permite mapear las texturas directamente sobre las caras de las mallas, y combinarse con colores de vértices para hacerla más brillante, o más oscura, o darle color.

Anexo III: Presupuesto

Mi incorporación a esta investigación fue en el segundo semestre del curso 2015-2016 como Prácticas Curriculares. Esto ha permitido que desarrollara previamente los conocimientos del funcionamiento de Blender y el funcionamiento del *middleware* Chiro, así como de los mini-juegos descritos en los antecedentes.

En cuanto al proyecto descrito en esta memoria, fue realizado durante el curso 2017-2018. En ese tiempo se llevaron a cabo las siguientes tareas:

- Investigación de sistemas equiparables a la investigación y estudio de teoría de gamificación [5].
- Tutorización y traspaso de conocimientos a mi compañera de proyecto Cristina Esteban.
- Diseño del entorno, pruebas de técnicas que encajaran con el ambiente que se quería dar en el juego, creación del escenario principal, texturas, renacuajo, primeras versiones del Avatar en su evolución adulta y del *riggin* de los personajes.
- Trabajo colaborativo con mi compañera Mónica Esteban, para el desarrollo de su proyecto de plataforma médica para el proyecto Blexer.
- Adaptación e implementación de los mini-juegos *Dive and Eat* y *Climb the tree* con el nuevo entorno y el nuevo personaje.

En cuanto a los recursos utilizados:

- *Software* libre Blender.
- Sensor Kinect 360 de Microsoft (120€) con adaptador para PC (20€).
- *Add-on* y *middleware* proporcionado por Ignacio Gómez-Martinho.
- Ordenador proporcionado por el CITSEM
- Dell Precision Tower 3620 Ci7 (1195€)
- Monitor HACER 24” K240HQ (123€)

Lo que supone un coste material de 1458 €.

En cuanto el coste personal, en una posición actual de Ingeniera Junior, y con un sueldo de 7€/hora, suponiendo que el proyecto ha sido desarrollado en un periodo de 10 meses, habiendo empleado 25 horas semanales (1125 horas aproximadamente) el sueldo correspondería a 7875 €.

Por lo tanto el coste final del proyecto es de 9333 €.