# Uso de Kinect para el entrenamiento de actividades físicas


**TFG**

**Víctor Tobes Pérez**
**Raúl Fernández Pérez**

**Departamento de Ingeniería del Software e Inteligencia Artificial**
**Facultad de Informática**
**Universidad Complutense de Madrid**

**Junio 2017**

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

# Uso de Kinect para el entrenamiento de actividades físicas

*Informe técnico del departamento*
**Ingeniería del Software e Inteligencia Artificial**
**IT/2009/3**

*Versión 1.0+*

**Departamento de Ingeniería del Software e Inteligencia Artificial**
**Facultad de Informática**
**Universidad Complutense de Madrid**

**Junio 2017**

ISBN 978-84-692-7109-4

# Agradecimientos

# Resumen

La investigación en *Reactive Virtual Trainers (RVT)* basados en el uso de `MS Kinect` ha dado lugar a una diversidad de soluciones para la danza, las artes marciales y el ejercicio físico en general. Este artículo presenta un prototipo de sistema para el entrenamiento de capoeira, un arte marcial brasileño a caballo entre los deportes de lucha y la danza. Este entorno está pensado para ser utilizado por el alumno para que pueda aprender a realizar diferentes movimientos de capoeira sin la necesidad de que esté presente un profesor. El entrenamiento consiste en imitar una serie de movimientos, los cuales han sido previamente capturados por varios expertos en el arte marcial. El entrenamiento virtual se desarrolla acorde al nivel que posea el alumno, ya sea principiante, aprendiz o avanzado. De este modo, el alumno va aprendiendo a realizar los movimientos de forma progresiva.

# Índice

# Índice de figuras

# Índice de Tablas

# Capítulo 1

# Introducción

## 1.1. Introducción

La capoeira es un arte marcial afro-brasileño que combina facetas de danza, música y acrobacias, así como expresión corporal **?**. Fue desarrollado en Brasil por descendientes africanos con influencias indígenas, probablemente a principios del siglo XVI. Es conocido por sus rápidos y complejos movimientos, que utilizan distinas partes del cuepro (brazos, piernas, codos, rodillas, cabeza...) para ejecutar maniobras de gran agilidad en forma de patadas, fintas y derribos, entre otros. La capoeira como estilo de lucha incorpora movimientos bajos y barridos, mientras que en el ámbito deportivo se hace más énfasis en las acrobacias y las demostraciones ritualizadas de habilidad. Se practica con música tradicional de berimbau.

Una de las características fundamentales de la capoeira, como de muchas otras artes marciales, consiste en la realización de manera repetida de un mismo movimiento para ir mejorando paulatinamente su ejecución, tanto en su trayectoria como en su velocidad o su fuerza. Sin embargo, especialmente en el caso de los principiantes, resulta complejo apreciar si existe mejora sin la supervisión constante de un entrenador, lo cual no resulta posible en la mayor parte de las ocasiones. Por este motivo, se ha planteado el desarrollo de un entrenador de capoeira, inicialmente en un nivel básico, que, mediante la utilización de tecnologías que ya son de uso doméstico (como son un ordenador personal y `Kinect`), supervise el proceso de entrenamiento de una persona que se inicie en el aprendizaje de la capoeira.

# Capítulo 2

# Estado del arte

## 2.1. Captura de movimiento

La captura de movimiento (abreviada `Mocap`, en inglés *Motion Capture*) es el proceso por el cual el movimiento, ya sea de objetos, animales o mayormente personas, se traslada a un modelo digital 3D.

En la actualidad, esta técnica llamada `fotogrametría`, se utiliza en la industria del cine y de los videojuegos, ya que facilita mucho la labor de los animadores al realizar un modelado mas realista. En el cine se utiliza como mecanismo para almacenar los movimientos realizados por los actores, y poder animar los modelos digitales de los diferentes personajes que tenga el film. En cambio, en el sector de los videojuegos se utiliza para naturalizar los movimientos de los personajes, de ese modo se obtiene una mayor sensación de realismo.Rodríguez-Esparragón y Domínguez Quintana Wikipedia.

## 2.2. Historia de la captura de movimiento

### 2.2.1. Precursores

Ya en la antigua Grecia, Aristoteles (384-322 AC) escribió el libro "De Motu Animalium"[1]. Él no solo veía los cuerpos de los animales como sistemas mecánicos, sino que perseguía la idea de como diferenciar la realización de un movimiento y como poderlo hacer realment, por lo que podría ser considerado el primer biomecánico de la historia.

Aproximadamente dos mil años después, Leonardo da Vinci (1452-1519) trató de describir algunos mecanismos que utiliza el cuerpo humano para poder desplazarse, como un humano puede saltar, caminar, mantenerse de pie...

Como pionero en la edad moderna, Eadweard Muybridge (1830-1904) fué

[1]Movimiento de los animales

el primer fotógrafo capaz de diseccionar el movimiento humano y animal, a través de múltiples cámaras tomando varias fotografías para captar instantes seguidos en el tiempo. Este experimento llamado .[el] caballo en movimiento", mostrado en la figura "tal", utilizó esta técnica de fotografía.

Comentario: Meter foto con los caballos

### 2.2.2. Nacimiento de la captura de movimiento

En la década de los 70, cuando empezaba a surgir la posibilidad de realizar animaciones de personajes por ordenador, se conseguía naturalizar los movimientos mediante técnicas clásicas de diseño, como la técnica de rotoscopia. Esta técnica consiste en reemplazar los frames de una grabación real por dibujos calcados cada frame. Los estudios *Walt Disney Pictures* utilizaron esta técnica en la película de 1937 "Blancanieves y los siete enanitos", para animar a los personajes del principe y Blancanieves.

Comentario: Meter foto de Blancanieves

Pero mientras, los laboratorios de biomecánica empezaban a usar los ordenadores como medio para analizar el movimiento humano. En la década de los 80, `Tom Calvert`, un profesor de kinesiología y ciencias de la computación en la universidad Simon Fraser (Canadá), incorporó potenciómetros a un cuerpo y la salida la usó para generar personajes animados por ordenador, con el objetivo de ser utilizados por estudios coreográficos y asistencia clínica para ayudar a pacientes con problemas de locomoción.

Comentario: Meter foto con los caballos

A principios de los años 80, centros como el MIT[2] empezaron a realizar experimentos con dispositivos de seguimiento visual aplicados en el cuerpo humano. Mas tarde, empiezan a cobrar importancia los primeros sistemas de seguimiento visual como el `Op-Eye` y el `SelSpot`. Estos sistemas normalmente usaban pequeños marcadores adheridos al cuerpo (Leds parpadeantes o pequeños puntos reflectantes) con una serie de cámaras alrededor del espacio donde se realizaba la actividad.

Comentario: Referenciar relación con la tecnología de los marcadores

En 1985, Jim Henson Productions[3] intentó crear versiones virtuales de sus personajes, pero no obtuvieron el éxito deseado, debido principalmente a la limitación de las capacidades de la tecnología en ese instante. Con los equipos 4D de Silicon Graphics y la perspicacia de Pacific Data Images[4], en 1988 los

[2] Massachusetts Institute of Technology
[3] Organización de entretenimiento norteamericana
[4] Productora de animación por ordenador

miembros de la compañía encontraron una solución viable para controlar las animaciones. Fueron capaces de regular la posición y los movimientos de la boca de un personaje a baja resolución y en tiempo real a través de la captura de movimiento de las manos de un actor con un aparato llamado Waldo, para su posterior interpretación en un ordenador. De esta manera surgió la primera marioneta virtual conocida como Waldo C. Graphic.

COMENTARIO: Meter foto Waldo

En 1988, Brad deGraf y Wahrman desarrollaron *Mike the Talking Head* de Silicon Graphics, capaz de mostrar las capacidades de sus nuevos equipos 4D en tiempo real. Mike estaba dirigido por un controlador que permitía controlar diferentes parámetros de la cara del personaje: como los ojos, boca, expresión y posición de la cabeza. El hardware de Silicon Graphics proporcionaba una interpolación en tiempo real entre las expresiones faciales y la geometría de la cabeza del personaje y del usuario. En el congreso de SIGGRAPH, Mike fue mostrado al público, donde se demostró que la tecnología *mocap* estaba preparada para su explotación.

COMENTARIO: Imagen de Mike deGraf

Años mas tarde, en los 90, deGraf continuó investigando en solitario en el desarrollo de un sistema de animación en tiempo real conocido como "Alive". Para lograr animar un personaje interpretado con "Alive", deGraf desarrolló un dispositivo especial con cinco pistones, los cuales representaban los dedos de la mano de la persona que controlaba al personaje virtual a modo de titiritero.

Con todo esto, deGraf pasó a formar parte de la compañía Colossal Pictures, donde animó a "Moxy":un perro generado por ordenador que presentaba un programa en Cartoon Network, mediante el sistema "Alive". "Moxy" es interpretado en tiempo real para publicidad, pero su uso en el programa era renderizado. Los movimientos del actor se capturaban mediante un sistema electromagnético con sensores en la cabeza, torso, manos y pies.

COMENTARIO: Imagen de Moxy

Tras estos avances, Pacific Data Images desarrolló un exoesqueleto de plástico, de modo que el actor se colocaría el traje con el objetivo de capturar los movimientos corporales: de la cabeza, pecho y brazos, a través de potenciómetros situados en la capa de plástico. De esta manera, los actores podían controlar los personajes virtuales mimetizando sus movimientos. Pese a que el traje se utilizó en varios proyectos, no se consiguieron los resultados esperados, ya que el ruido de los circuitos y el diseño inapropiado del traje no lo permitían.

En torno a 1992, la empresa SimGraphics desarrolló un sistema de rastreo facial llamado *Face Waldo*. Consiguieron capturar la mayor parte de los

movimientos usando sensores adheridos a la barbilla, labios, mejillas, cejas y en el armazón del casco que llevaba el actor para su posterior aplicación en un personaje virtual en tiempo real. Este sistema logró ser novedoso ya que el actor podía manejar las expresiones faciales de un personaje a través de sus movimientos, logrando unos gestos mas naturales que los capturados anteriormente.

Lo que produjo un éxito mayúsculo con este proyecto fue la interpretación en tiempo real de Mario, el personaje principal de la saga de videojuegos *Mario Bros*. Estaba controlado por un actor mediante *Face Waldo*, Mario conseguía dialogar con los miembros de una conferencia, respondiendo a sus preguntas. A partir de ese momento, SimGraphics se centró en la animación en directo, desarrollando personajes para televisión y otros eventos en directo, mejorando la fiabilidad del sistema para el rastreo facial.

COMENTARIO: Imagen de Mario

Poco después, en el congreso de SIGGRAPH en 1993, la empresa Acclaim[5] asombró al público con animaciones realistas y complejas de dos personajes animados en su totalidad mediante captura de movimientos. En los años anteriores, habían desarrollado de forma encubierta un sistema de rastreo óptico de alta definición, muy superior a los citados anteriormente, capaz de seguir 100 marcadores de forma simultanea en tiempo real.

De forma gradual, la técnica *mocap* se fue expandiendo entre las empresas desarrolladoras de sistemas de captura de datos, con el objetivo de crear productos y métodos que albergasen nuevos sectores empresariales. Gracias al desarrollo previo, en 1995 hizo su aparición el primer videojuego en el que se aplicó esta técnica de forma extensa, se trataba de Highlander: The Last of the Macleods de la compañía Atari, el cual marcó el inicio del desarrollo tecnológico sobre videojuegos y productos audiovisuales.

A lo largo de los años, fueron apareciendo largometrajes que mostraban como la captura de movimientos ha evolucionando día tras día y con una gran proyección de futuro, ha pasando de ser algo que utilizaban de forma esporádica algunos personajes, a ser indispensable en cualquier producción. Como por ejemplo en los *films*: Jar Jar Binks en la saga de *Star Wars*, el personaje de La Momia, Gollum en la trilogía de El Señor de los Anillos y de El Hobbit, Final Fantasy: La fuerza interior, Avatar, etc. También existen videojuegos recientes que utilizan esta tecnología en su desarrollo: The Last os Us, Beyond: Two Souls, la saga Uncharted, Until Dawn, L.A. Noire, etc.

COMENTARIO: Imagen de pelicula y videojuego con mocap

[5]Empresa encargada del desarrollo, publicación, venta y distribución de videojuegos para diferentes compañias (Sega, Nintendo, Sony, Microsoft...)

## 2.3. Métodos de captura de movimiento

En la actualidad existen numerosos sistemas para la capturar de movimientos. Dependiendo de las necesidades de la producción, ya estén relacionados con el presupuesto disponible, así como las posiciones, velocidades, impulsos del actor o el nivel de realismo al que se quiera llegar. Atendiendo a su tecnología, se pueden encontrar los diferentes métodos que se desarrollan a continuación.

### 2.3.1. Captura de movimiento mecánica

En el proceso de captura de movimiento mecánica, el actor viste unos trajes especiales adaptables al cuerpo humano. En su mayoría, estos trajes están compuestos por estructuras rígidas con barras metálicas o plásticas, unidas mediante potenciómetros colocados en las principales articulaciones. Los potenciómetros se componen de un elemento deslizante acoplado a una resistencia, la cual produce una variación de tensión que puede medirse para conocer el grado de apertura de la articulación donde se encuentre acoplado. Los sensores que recogen la información pueden transmitirla mediante cables, pero normalmente lo hacen con radiofrecuencia.

Figura 2.1: Sistema de captura de movimiento mecánico

Captura de movimientos óptica

Captura de movimientos en vídeo o Markerless , LUZ ESTRUCTURADA de kinect

Captura de movimientos inercial

# Capítulo 3

# Sensor Kinect

## 3.1. Versiones de Kinect

### 3.1.1. Kinect V1

#### Características

Video: 640x480 @30 fps

### 3.1.2. Kinect V2

COMENTARIO: Enlaces sobre las caracteristicas de Kinect

https://msdn.microsoft.com/library/jj131033.aspx
https://msdn.microsoft.com/library/dn782025.aspx
https://developer.microsoft.com/es-es/windows/kinect/hardware

# Capítulo 4

# Materiales

En este capítulo se expondrán los materiales y métodos utilizados para el desarrollo del proyecto. A continuación se detallarán de forma técnica las tecnologías y los dispositivos utilizados para realizar el proyecto.

## 4.1. Software y hardware empleados

El software principal utilizado en este proyecto es Unity 5.5 Technologies (c), que es un motor de desarrollo de videojuegos. Para la edición , compilación y depuración de la programación de Unity se ha utilizado Visual Studio 2015 Microsoft con el lenguaje de programación C#.

COMENTARIO: Aqui raul pones los programas de modelado de avatar. Animaciones de mixamo,.

El hardware principal usado en este proyecto es el dispositivo de captura de movimiento kinect v2 (referencia) y el cable de conexion(nose como se llama, referencia).

### 4.1.1. Entorno de desarrollo : Unity 3D

El objetivo de este proyecto es la captura y el análisis de movimiento relacionados con el arte marcial afro-brasileño capoeira. Con esto en mente, se llega a la primera decisión de que plataforma escoger para el desarrollo de este proyecto.

Los entornos a elegir fueron Visual Studio 2015 Microsoft, Unity Technologies (a) y Unreal Engine Games. El primero descartado fue Visual Studio 2015 porque se busca crear también un escenario 3D y, tanto Unity como Unreal ,ofrecen herramientas útiles para la creación de estos escenarios.

En cuanto a la decisión de elegir entre Unity o Unreal Engine fue basada

en la comunidad que hay detrás de cada uno de ellos, y sobre todo, en lo que se quiere abarcar con este proyecto. Unreal suele ser utilizado por las empresas para juegos grandes y mas profesionales , mientras que Unity se puede aplicar, tanto a pequeñas como a grandes aplicaciones, siendo su aprendizaje de este mas intuitivo que Unreal.

Un motor de juegos se especifica como la rutina de programación que permiten el diseño, la creación y la representación de un videojuego. En este caso, se adecua correctamente ya que se pretende realizar un entorno 3D donde los usuarios se perciban y aprendan movimientos controlando un avatar.

La funcionalidad de un motor de juegos consiste en proveer al juego de un motor de renderizado para los gráficos 2D y 3D, motor físico, sonidos, scripting, animación, un escenario gráfico, etc..

Unity es un motor de juegos desarrollado por la compañía `Unity Technologies` y utiliza los lenguajes de programación C ,C++ , C# y javascript.


Figura 4.1: Logo Unity

En un primer momento Unity salió al mercado ofreciendo un software de calidad y barato añadiendo el objetivo de que pequeñas y grandes empresas pudieran utilizarlo por igual, dando además, la posibilidad de que su contenido sea compatible con prácticamente cualquier medio o dispositivoTechnologies (b). Gracias a ello, Unity creció exponencialmente en el mercado, generando así, una gran comunidad que puede servir de pilar de ayuda para los nuevos desarrolladores.

Unity ofrece diferentes tipos de versiones como la personal, que es gratuita y la profesional, que es de pago. En la versión personal es necesario promocionar el logotipo de Unity y carece de funcionalidades adicionales de renderizado del motor que la profesional si contiene.

Otro aspecto importante de Unity es el Asset Store. Es una biblioteca del editor que contiene una multitud de paquetes o *Assets* comerciales y gratuitos creados, de forma continua, por la comunidad o por `Unity Technologies`. Estos paquetes pueden contener modelados 3D, texturas, animaciones y demás de contenido que servirá de ayuda al proyecto al no poder contar con

un experto diseñador.

La apariencia del editor de Unity se observa en la figura 4.2

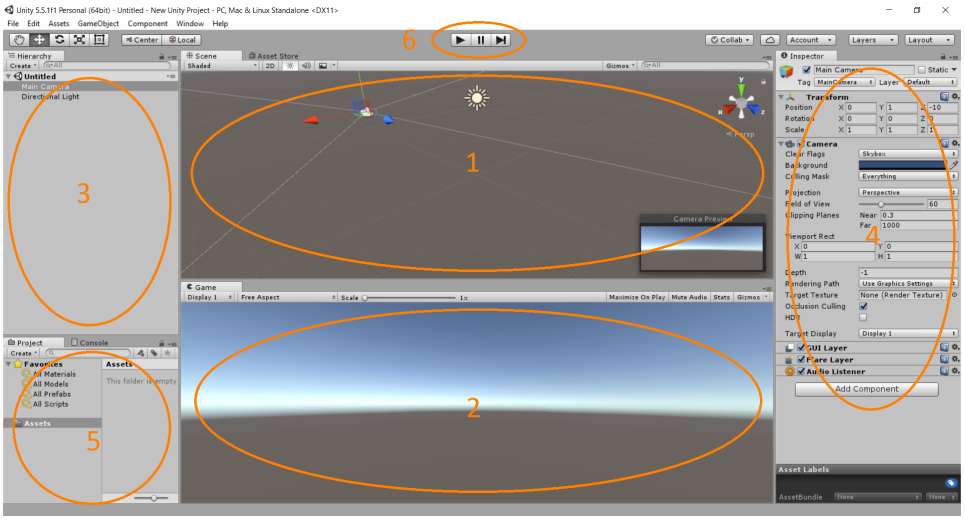


Figura 4.2: Apariencia de editor Unity

Como se observa en la imagen, el editor de Unity esta formado por varios paneles. La estructuración básica de un juego de Unity se realiza mediante escenas, para tener una previsualizanción de los elementos que aparecen en una escena se utiliza el panel uno denominado *scene*. Este panel representa una dimensión 3D donde se podrá rotar y desplazar en los tres ejes ,para poder así, modificar todos los elementos de la escena en cualquier ángulo.

El segundo panel denominado *Game* muestra la vista generada por la cámara, es decir, la imagen que se verá en el juego.

Los elementos incorporados a la escena se archivan en el panel tercero con el nombre *Hierarchy*. Estos pueden ser objetos 3D, cámara, sonidos, luces,e incluso objetos de tipo GUI para la implemetación de interfaces. Los elementos se muestran de una forma jerárquica y al seleccionar uno se expondrán sus características y componentes en el panel cuarto denominado *Inspector*. Por ejemplo, al seleccionar la cámara en el *Hierarchy* muestra su componente *transform*, este muestra la información de su posición y rotación, pudiendo ser modificable.

El panel quinto llamado *Project* agrupa ,en forma de directorio, todos los scripts , paquetes , imágenes y archivos relacionados con el proyecto.

La ejecución de pruebas del juego se realiza por medio del *Play mode* situado en el panel sexto. Esta funcionalidad permite poner en funcionamiento

el juego , pararlo y detener su ejecución, siendo esto, fundamental para comprobar y depurar un juego Technologies (c).

Una vez estudiado el editor de Unity se toma la decisión de que lenguaje de programación utilizar. Unity soporta dos lenguajes:

- C#: lenguaje estandar similar a C++ y Java.
- *UnityScript* : lenguaje basado en javascript propio de Unity.

Se optó por la decisión de utilizar C# porque se acomoda mejor a la aplicación, puesto que no está orientada a web, y además ofrece unas características mejores que el *UnityScript*.

Para controlar los eventos de un objeto o los eventos de las entradas de los usuario, Unity utiliza scripts.De forma básica los scripts tienen implementado un método *Start* para inicializar las variables del objeto y un método *Update* encargado de actualizar los parámetros a partir del tratamiento de eventos.

Se utilizó Visual Studio 2015 Microsoft para la depuración de scripts, ya que proporciona visualización de las variables utilizadas en tiempo de ejecución, y además, da la posibilidad de ejecutar el script paso a paso. También otro factor clave fue que este entorno de programación se tenía experiencia previa de uso.

### 4.1.2. MakeHuman

La aplicación MakeHuman sirve para crear modelos humanos que pueden ser utilizados en diferentes plataformas 3D, como animaciones, videojuegos, etc. Promueve el arte digital y artistas digitales, proporcionando una herramienta de alta calidad liberada al dominio público con la licencia CCO, mientras que la base de datos y el código se publicaron mediante una licencia AGPL3.

MakeHuman utiliza un humano estándar y mediante varios deslizadores o scrolls se podrán alterar los parámetros de la estatura, anchura, sexo, edad, etc.

En un principio se pensó utilizar esta aplicación para el modelado de los personajes debido a su intuitiva interfaz y la facilidad de exportación en formato fbx el cual utiliza Unity. Pero tras investigar posibles alternativas, se encontró la aplicación Fuse Character Creator, la cual posibilitaba el desarrollo de los personajes con mayor calidad. De forma simultanea se podría realizar la configuración de los personajes para su posterior animación.


Figura 4.3: Logo MakeHuman

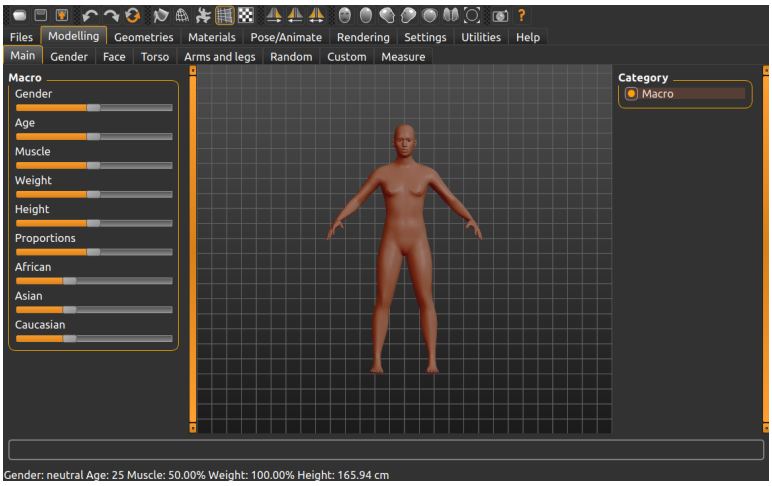


Figura 4.4: Aspecto del humano estándar de MakeHuman

### 4.1.3. Fuse Character Creator

Se trata de una aplicación desarrollada por Mixamo, la cual permite crear personajes únicos mediante una intuitiva interfaz que separa el avatar en diferentes partes, como son la cabeza, el torso, las piernas y los brazos. Posteriormente a la selección de las diferentes partes del cuerpo ya predefinidas por el sistema, se puede editar cada una de las partes del personaje, ya sean los rasgos de la cara, el tamaño de los brazos, la longitud de las piernas, etc. Al seleccionar la parte del cuerpo que se quiere modificar, simplemente será necesario pinchar y realizar un scrool o deslizamiento hacia la izquierda o derecha para aumentar su tamaño, y para seleccionar el punto exacto donde se ubicará por ejemplo el ombligo, los ojos, las rodillas, etc, se realizará

un scrool o desplazamiento hacía arriba o abajo.

Al terminar de modelar el personaje, se pasará a la sección de vestuario. Donde existen prendas predefinidas, como camisas, pantalones, zapatos, etc. Al terminar de vestir al avatar, esta aplicación posee la peculiaridad de poder subir al servidor de Mixamo el modelado realizado y completar la configuración de los huesos además de incluir una posible animación.

### 4.1.4. Mixamo 3D Animation Online Services

Mixamo es una compañía tecnológica encargada del desarrollo y venta de servicios para la animación de personajes 3D.

### 4.1.5. Marvelous Designer

Maneja un software encargado en la simulación de telas, el cual permite a los artistas y diseñadores de moda modelar ropa en 3D.

### 4.1.6. Blender

Es una aplicación dedicada al modelado, renderizado, animación y creación de gráficos 3D.

### 4.1.7. MonoDevelop

| Comentario: Enlaces sobre las caracteristicas de Kinect |
|---|

https://msdn.microsoft.com/library/jj131033.aspx
https://msdn.microsoft.com/library/dn782025.aspx
https://developer.microsoft.com/es-es/windows/kinect/hardware

# Capítulo 5

# Desarrollo Del Proyecto

Una vez elegida la tecnología y las plataformas que se van a utilizar, se empieza a preparar el desarrollo del proyecto. En primer lugar se investiga que librerías y paquetes se necesitan para tener una buena conexión con `Kinect` y, además, que información obtenida del sensor de movimiento podemos aprovechar para el proyecto.

## 5.1. Paquetes para Unity

El primer paquete que se prueba es el que nos ofrece `Microsoft` [1],que es un paquete destinado principalmente para `UnityPro`.[2] Al añadir este primer paquete al proyecto se empieza a manifestar una serie de errores de scripts, esto es debido, a la incongruencia de versiones de Unity. El proyecto se desarrolla en Unity Personal, que es una versión gratuita de Unity , mientras que el paquete que ofrece Microsoft esta especificado para Unity Pro. Aun así, se corrigen errores de comandos y llamadas a funciones obsoletas para ver si se puede aprovechar este paquete.

En una primera instancia se ejecuta la escena que viene como ejemplo en el paquete de Unity para ver su funcionamiento y se observa que tiene un comportamiento intermitente, es decir, cuando nos colocamos enfrente de la cámara de kinect a veces mostraba un esqueleto verde que emulaba la persona captada y otras veces dejaba de funcionar sin saber el error producido. Debatiendo e intentando comprender si estos errores se producían por conexión de kinect o por funcionamiento incorrecto de la librería que nos ofrecía microsoft se opto por la opción de descartar este paquete para evitar futuras fustraciones.

Después de descartar el paquete que nos ofrecía Microsoft decidimos buscar en el Assets Store de Unity(Explicacion o referencia).Encontramos el pa-

[1] https://developer.microsoft.com/es-es/windows/kinect/tools

[2] Unity profesional de pago con mas funcionalidad que Unity personal.

quete `Kinect v2 Examples with MS-SDK`[3] que lo elegimos por su valoración positiva y también porque hay poca variedad de paquetes relacionados con kinect v2.

Este paquete tiene todo lo necesario para reconocer que la kinect está conectada y para poder utilizar los datos que se obtienen del sensor. En una primera toma, el paquete nos ofrece una serie de ejemplos sencillos para poder comprender mejor el funcionamiento de kinect. Para que este paquete funcione es necesario tener instalado Kinect SDK 2.0 que son los drivers de la kinect v2 .

Los ejemplos que tiene implementado el paquete de Kinect v2 Examples with MS-SDK son variados:

- AvatarsDemo, simulador que muestra un avatar en tercera persona que correspondería a la persona captada y se puede controlar sobre el escenario 3D.
- BackgroundRemovalDemo, son ejemplos que cambian el fondo que se encuentra detrás del usuario captado.
- ColliderDemo, una serie de ejemplos para ver el funcionamiento de colisiones del usuario captado con los objetos que aparecen en la escena.
- FaceTrackingDemo, este ejemplo reconoce la dirección de tu cabeza para girar la cámara de la imagen para simular la vista humana.
- FittingRoomDemo, este ejemplo te da la opción de ponerte ropa encima de tu imagen real captada.
- GesturesDemo, serie de ejemplos de funcionamiento de los gestos de kinect.
- InteractionDemo, este ejemplo muestra como el usuario puede girar,rotar y agrandar un objeto con el movimiento de sus manos.
- KinectDataServer, implementa un servidor de datos para guardar información como gestos de kinect.
- MovieSequenceDemo, este ejemplo mostrará como reproducir un conjunto de frames de película con el cuerpo del usuario.
- MultiSceneDemo, este ejemplo concatenará diferentes escenas de Unity basadas en las componentes de este paquete.
- OverlayDemo, son tres ejemplos que muestras como interactuar con los objetos de la escena, para ello, se basa en el movimientos de los brazos y manos para hacer que los objetos se mueven, roten y se desplazen.

[3] https://www.assetstore.unity3d.com/en/#!/content/18708

- PhysicsDemo, muestra un simulación de físicas que capta el movimiento del brazo para lanzar una pelota virtual.
- RecorderDemo, ejemplo que muestra como grabar y reproducir un movimiento captado por kinect.
- SpeechRecognitionDemo, ejemplo que sirve para realizar acciones por comandos por voz, aunque se producen errores cuando se probó este ejemplo.
- VariousDemos, implementa dos ejemplos, como pintar en el aire moviendo los brazos y el otro dibuja bolas verdes que se colocan en tus articulaciones simulado un esqueleto.
- VisualizerDemo, este ejemplo convierte la escena, según lo ve el sensor, en una malla y la superpone sobre la imagen de la cámara.

Una vez explicado los ejemplos, elegimos cual de ellos podríamos utilizar para aprovechar su funcionalidad y tener un apoyo base para el desarrollo del proyecto. Los ejemplos seleccionados serían el `AvatarsDemo, GestureDemo y RecorderDemo`, más adelante se explicar con mas detalle que se utiliza de estos ejemplos.

### 5.1.1. Grabar y reproducir movimientos de Usuario

Como el objetivo principal de este proyecto es el entrenamiento de actividades físicas, se selecciona como ejemplo de primer estudio, el `Recorderdemo` por su potencial para guardar y reproducir un movimiento.

#### 5.1.1.1. Investigación base

En una primera vista se muestra como `Kinect` capta al usuario, esta representación se hace mediante un cubeman[4] que simula todo el movimiento que realiza el usuario.

La representación del movimiento del usuario se muestra mediante un *cubeman*, que simula todo el movimiento que realiza el usuario. Esta figura se crea gracias al *script* de `Cubeman Controller`, el cual se inicializa con el número del cuerpo que se quiere mostrar, siendo seis las personas que pueden ser detectadas por `Kinect v2`, y también pudiendo escoger una representación del *cubeman* en modo espejo. Los datos del *cubeman* (ver Figura

[4]Esqueleto verde que emula el movimiento de la persona captada.

??) están definidos con 25 `GameObjects` llamados `Joints`, que hacen una representación de las articulaciones del cuerpo humano. Estos `Joints` se representan en `Unity` con unas coordenadas (X,Y,Z) y una rotación. Toda la información del movimiento se obtiene a partir del sensor de `Kinect`. Los 25 `Joints` serían : cadera central y laterales , pecho, clavícula, cuello, cabeza, hombros, codos, muñecas, pulgares, manos centrales, rodillas, tobillos y pies.

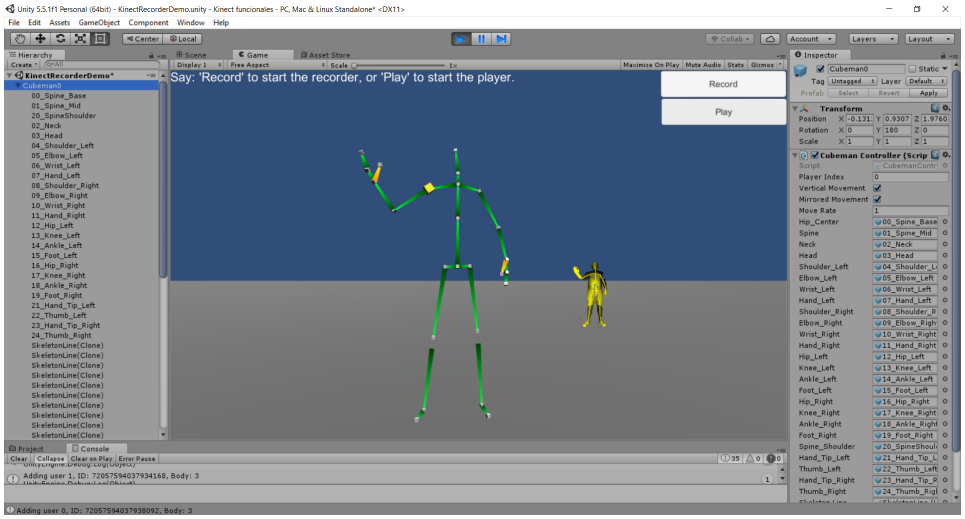


Figura 5.1: Cubeman con la lista de todos los Joints

Para hacer una representación correcta del esqueleto y se mueva como una entidad, el *script* `Cubeman Controller`, pone la posición del `Joint` de la cadera base como la posición y rotación del objeto padre y todos los demás joint serán hijos de este `GameObjects`. Para calcular la posición relativa de todos los `Joints` se resta la posición del padre con la posición de cada `Joint` dada por el `Kinect Manager`.

El *script* `Kinect Manager` es un `Singleton`[5] encargado de la comunicación entre el sensor de `Kinect` y las aplicaciones de Unity.

Este *script* al ser común a todos los ejemplos nombrados anteriormente, implementa todas las funciones y comunicaciones necesarias para su funcionamiento, pero en este caso, se investiga como obtiene la información que le envía al `cubeman Controller`. `Kinect Manager` se encarga de analizar la información del `SensorData`, siendo este, una estructura de datos ofrecida por el *script* `KinectInterop`[6] donde aparece la información de la imagen, pronfundidad, color y datos de los usuarios captado con el sensor. Los datos de los usuarios captados se encapsulan en el `BodyDataFrame`, esta estructura,

[5] Es un patrón que consiste en que existe una única instancia y es la propia clase la responsable de crear la única instancia. Permite el acceso global a dicha instancia mediante un método de clase

[6] *script* encargado de tener comunicación directa con el sensor de la `Kinect`

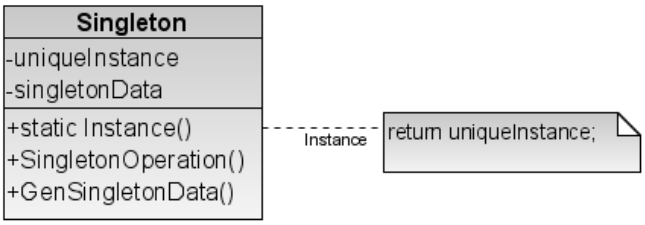


Figura 5.2: Patrón de Singleton

a vez, contiene la información del número de usuarios captados, su identificador, la información de los BodyData[7], etc.

Una vez comprendido el `SensorData`, se observa como el `Kinect Manager` ,analizando el `BodyDataFrame`, va actualizando y asignando a cada usuario el identificador correspondiente de un `BodyData` en todo momento ,y este a su vez, va pasando al `Cubeman Controller` esta información para que reproduzca el movimiento.

Otro *script* que se debe mencionar es el `KinectRecorderPlayer`, que es el encargado de guardar y reproducir un movimiento a partir de un fichero de texto. La información se le pide al `Kinect Manager`, y este a su vez, al `KinectInterop` devolviéndola en una cadena de caracteres. Esta cadena hace referencia a un *frame*[8] que tiene la información del instante de tiempo, los *BodyData* captados y las coordenadas (x,y,z) respecto al mundo de todos sus `Joints`. Los *BodyData* captados aparece primero su identificador y después sus 25 `Joints`, los *BodyData* no captados aparecerá un cero (Figura 5.3). `KinecRecorderPlayer` también discierne entre grabación y reproducción activando solo una de la dos a la vez.

```
0.022|kb,95616099991060000,6,25,0,1,72057594037972009,2,0.037,-0.116,2.546,2,0.048,0.130,2.483,2,0.056,0.371,2.413
0.065|kb,95616103283010000,6,25,0,1,72057594037972009,2,0.037,-0.116,2.546,2,0.047,0.131,2.484,2,0.056,0.372,2.414
0.089|kb,95616106582580000,6,25,0,1,72057594037972009,2,0.036,-0.115,2.546,2,0.045,0.131,2.484,2,0.053,0.372,2.414
0.131|kb,95616109991070000,6,25,0,1,72057594037972009,2,0.036,-0.115,2.547,2,0.044,0.132,2.485,2,0.050,0.373,2.414
0.155|kb,95616113283760000,6,25,0,1,72057594037972009,2,0.036,-0.115,2.547,2,0.043,0.132,2.485,2,0.048,0.374,2.414
0.199|kb,95616116582770000,6,25,0,1,72057594037972009,2,0.036,-0.114,2.547,2,0.042,0.133,2.484,2,0.047,0.375,2.413
0.222|kb,95616119990830000,6,25,0,1,72057594037972009,2,0.035,-0.114,2.547,2,0.040,0.134,2.485,2,0.043,0.376,2.414
0.265|kb,95616123282920000,6,25,0,1,72057594037972009,2,0.034,-0.113,2.548,2,0.039,0.135,2.486,2,0.042,0.377,2.414
0.289|kb,95616126583000000,6,25,0,1,72057594037972009,2,0.034,-0.112,2.548,2,0.038,0.135,2.486,2,0.041,0.377,2.414
0.333|kb,95616129983880000,6,25,0,1,72057594037972009,2,0.034,-0.112,2.548,2,0.037,0.136,2.486,2,0.039,0.377,2.414
0.355|kb,95616133282640000,6,25,0,1,72057594037972009,2,0.033,-0.111,2.548,2,0.037,0.136,2.486,2,0.039,0.378,2.413
0.400|kb,95616136585460000,6,25,0,1,72057594037972009,2,0.032,-0.111,2.550,2,0.036,0.136,2.486,2,0.038,0.377,2.413
0.422|kb,95616139993730000,6,25,0,1,72057594037972009,2,0.032,-0.111,2.550,2,0.035,0.136,2.485,2,0.036,0.376,2.411
0.465|kb,95616143282610000,6,25,0,1,72057594037972009,2,0.032,-0.111,2.550,2,0.035,0.136,2.485,2,0.035,0.376,2.409
0.489|kb,95616146582650000,6,25,0,1,72057594037972009,2,0.032,-0.111,2.550,2,0.031,0.137,2.479,2,0.025,0.381,2.393
0.532|kb,95616149990760000,6,25,0,1,72057594037972009,2,0.031,-0.111,2.549,2,0.029,0.136,2.476,2,0.024,0.378,2.390
0.556|kb,95616153283180000,6,25,0,1,72057594037972009,2,0.033,-0.110,2.540,2,0.029,0.135,2.465,2,0.023,0.376,2.380
```

Figura 5.3: Fichero de texto con los datos del BodyData

[7] Información de los joint de un cuerpo especifico
[8] Es un fotograma, `Unity` por defecto reproduce a 60 *frames* por segundo

#### 5.1.1.2. Cambios e implementaciones

Después de haber realizado la investigación y compresión del material que se puede aprovechar para grabar y reproducir un movimiento, se plantean una series de desafíos e implementaciones que se deben realizar.

En primer lugar se debe implementar la forma de poder reproducir un movimiento grabado a la vez que muestra al usuario captado en tiempo real, para ello, se implementa una función llamada `SetBodyFrameFromCsvAndPoll` en el `KinectInterop`. El funcionamiento de esta función es el de obtener primero el `SensorData` con los datos del usuario captados por la `Kinect` y después, se busca un *BodyData* que esté libre de ese mismo SensorData, para así, poder rellenarlo con los datos del fichero de texto. También se añade un campo más a la estructura de *BodyData* para saber si los datos son procedentes del sensor o de un fichero de texto. Posteriormente de hacer estos cambios, se realiza una modificación en el *script* `Cubeman Controller` para poder escoger si los datos proceden del sensor o de un fichero texto. Con todas estas mejoras implementadas, se empieza a modificar el escenario de `Unity` para mostrar dos cámaras. La primera cámara enfoca un escenario donde se encuentra el *cubeman* que representa al usuario captado por el sensor de `Kinect` y la segunda cámara muestra otro escenario colocado en la esquina superior derecha que contiene otro *cubeman* que está a la espera de reproducir el movimiento que está guardado en un fichero de texto(Figura 5.4).

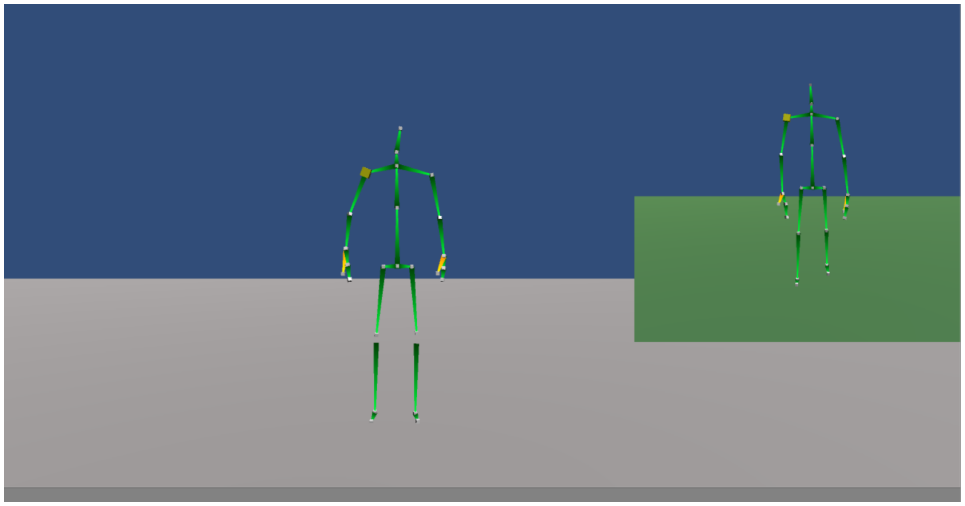

Figura 5.4: Escena con los dos cubeman

### 5.1.2. Comparación de los movimientos de Usuario

Otra etapa en el desarrollo de este proyecto se trata de comparar un movimiento grabado con el que esté realizando el usuario.

#### 5.1.2.1. Investigación base

La primera forma de comparar un movimiento en la que se debatió, fue la de estudiar las trayectorias de cada movimiento, calculando así, la gráfica que generaban las coordenadas de los `Joints` a lo largo del tiempo y determinar la pendiente. Esta forma de comparación era muy especifica y laboriosa para cada movimiento, por lo que se intento tener otro camino para hacer la comparación de forma más genérica.

Con esta idea en mente ,se estudió el ejemplo de `GestureDemo` que reconoce gestos del usuario para rotar y hacer *zoom* sobre un cubo. Para que reconozca estos gestos previamente, hay que añadirlos a las lista de gestos del *script* `KinectGesture` e implementar su funcionalidad en el método *CheckForGesture*. Posteriormente hay que notificar al `KinectManager` que gesto de los registrados se quiere detectar.

En cuanto a la funcionalidad del gesto, se realiza por estados y progresión, permaneciendo en el primer estado hasta que identifique una posición especifica. Para completar el gesto se tendrá un tiempo determinado para llegar a la progresión final del movimiento, sino el gesto se cancelará.

#### 5.1.2.2. Cambios e implementaciones

Una vez analizado la funcionalidad de los gestos de `Kinect`, se apoyó en esa idea para implemenentar la comparación del movimiento como un gesto de `Kinect`. Para empezar añadimos el gesto al `KinectGesture` con el nombre *Move*, acto seguido, se implementa la funcionalidad en *CheckForGesture* haciendo que tenga dos estados. El estado cero se encarga de calibrar la posición inicial ,dando al usuario, la oportunidad de colocarse en esa posición y ofreciéndole un margen de tres segundos para que se prepare.

Para el siguiente estado hay que explicar antes, que la información del movimiento esta guardada previamente en una lista de *strings* y cada elemento corresponde a un *frame*. Después de este inciso se define el siguiente estado como el estado por defecto ,y es así, porque engloba todos los siguiente estados al estado cero. El número del estado será utilizado para acceder y obtener el *frame* de la lista del movimiento. En este momento se decide que para que sea más eficiente la transisción de estados, se incrementa en más de una unidad el estado, ya que la comparación de dos *frames* consecutivos es muy similar, ahorrandose así comparaciones innecesarias. Para pasar al siguiente estado habrá una comparación de cada uno de los 25 `Joints` del usuario con los `Joints` del *frame* correspondiente. La comparación consiste

en igualar las coordenadas x e y con un margen de error modificable por el usuario, si cumple este requisito pasará al siguiente estado. La comparación tiene un tiempo límite que será el tiempo del movimiento más dos segundos de espera, en ese tiempo el usuario deberá superar todos los estados para que el movimiento sea realizado correctamente, en caso contrario fracasará y se cancelará el gesto (Figura 5.5). El resultado del gesto se muestra por pantalla para dar un *feedback* inmediato al usuario.

Como `Kinect` es una cámara frontal surgen algunos problemas de profundidad haciendo que la comparación de la coordenada z se cumpla ocasionalmente, por ello, se toma la decisión de obviarla por su comportamiento extraño.

### 5.1.3. Avatares y animaciones

En esta etapa del proyecto, se han desarrollado los diferentes avatares con una fisionomía humana para dar una sensación mas realista a los *cubeman* y de este modo se proporcionará una perspectiva mas amena al usuario.

#### 5.1.3.1. Investigación base

Comentario: Foto de la configuración de rig en unity

La primera aplicación que se utilizó para crear los personajes que darían vida al proyecto fue `MakeHuman`, ya que mostraba una interfaz amigable e intuitiva (ver Figura 5.6 ). El primer avatar utilizado fue simple, ya que al principio interesaba ver el resultado que proporcionaba el personaje junto con el *cubeman* y de esta forma determinar la configuración de los huesos que mas se ajustaba a las necesidades dadas. Para comprobar que esqueleto se debía emplear,se tuvieron que crear cuatro avatares diferentes, con 31, 163, 137 (ver Figura 5.7 ) y 53 huesos respectivamente.

Después de realizar varias pruebas, se determinó que el avatar que mejor se adaptaba a los movimientos capturados por *Kinect* era *Default no toes* ver Figura 5.7 ). Este personaje poseía 137 huesos, de los cuales 25 serían utilizados por el *cubeman*.

Para comprobar como se visualizaban los movimientos de un avatar con ropa, se utilizó el personaje inicial vestido con un peto y una camiseta (ver Figura 5.8 ). Al observar de los movimientos capturados con `Kinect` que las texturas de la piel y la ropa no plasmaban un realismo razonable, se optó por buscar otro software que ayudase a realizar esa tarea.

En la búsqueda se encontraron aplicaciones dedicadas al modelado de

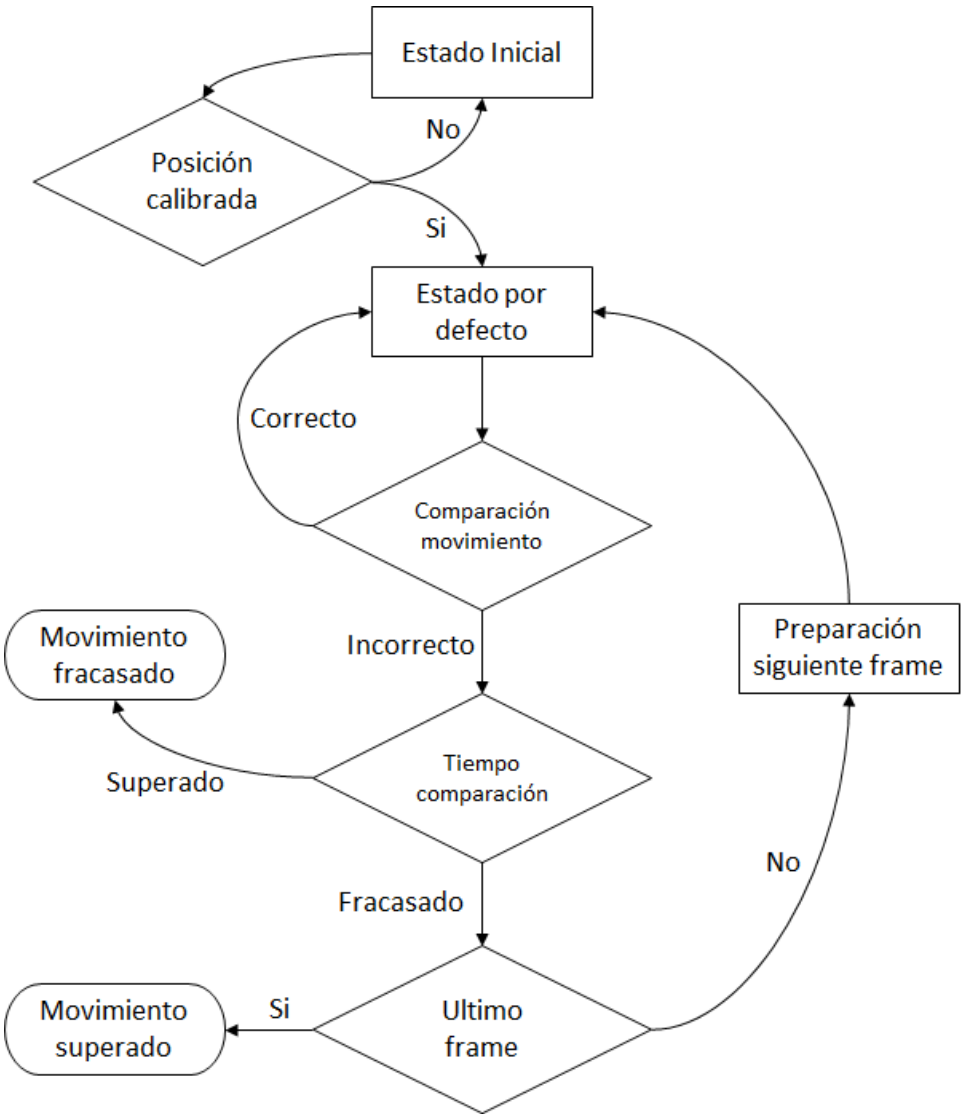


Figura 5.5: Grafo de comparación de movimiento

personajes 3D como `Blender`, pero con una curva de aprendizaje demasiado larga para las necesidades dadas. También se examino una aplicación llamada `MarvelousDesigner` la cual se emplea para desarrollar prendas de vestir. El problema surgía cuando se intentaba importar el avatar creado con `MakeHuman`, ya que no eran compatibles los formatos de las dos aplicaciones y por lo tanto se descartó seguir por esa vía. En el camino se observó que existía una compañía llamada `Mixamo` que ofrecía una aplicación para el desarrollo de avatares y a su vez, un entorno dedicado a la animación de los personajes creados. Por este motivo, se tomó la decisión de cambiar a `Fuse Character Creator de Mixamo` como aplicación para el desarrollo de

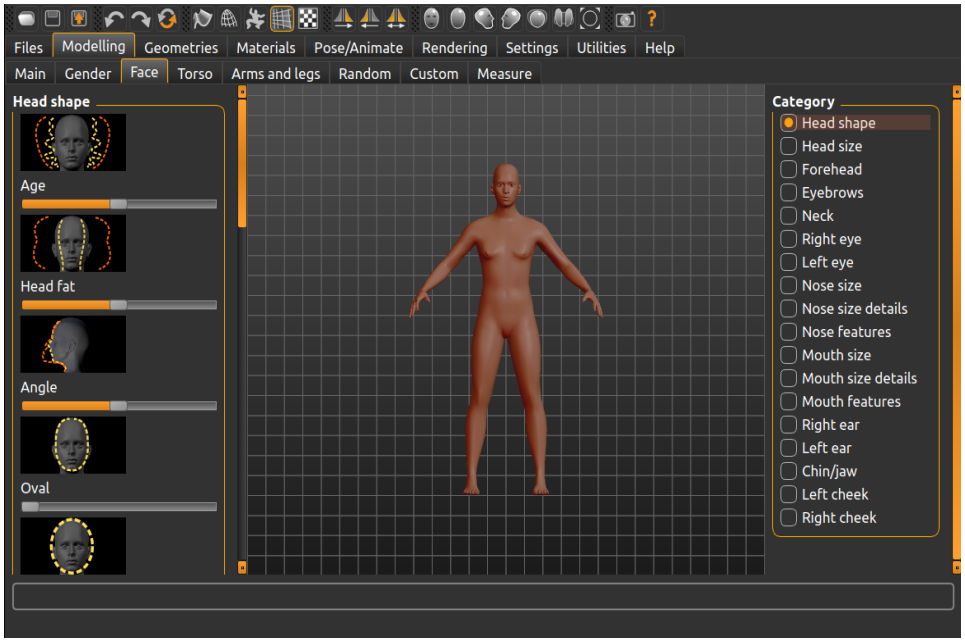


Figura 5.6: Avatar inicial

los personajes, ya que se amoldaba perfectamente a las necesidades de este proyecto.

### 5.1.4. Cambios e implementaciones

A fin de realizar un entorno mas realista se crearon cuatro avatares diferentes, cada uno con una fisionomía propia. A la hora de elaborar el vestuario de los personajes, se observó la ropa que utilizan los integrantes de la escuela `Abadá-Capoeira` para tener un modelo a seguir. Se trata de un pantalón largo y una camiseta de tirantes en color blanco con el logo de la escuela.

Con el propósito de crear una ropa muy similar a la utilizada por la escuela `Abadá-Capoeira`, se usaron prendas prediseñadas en la aplicación `Fuse Character Creator` como los pantalones, camisetas y el top que utilizarían los avatares del sistema con los retoques apropiados para darle ese color blanco característico.

Los dos avatares principales, como son el alumno y el profesor se desarrollaron con la misma vestimenta que utiliza la escuela `Abadá-Capoeira`, en cambio los dos avatares que se emplean para dar un mayor dinamismo a la aplicación utilizan el mismo pantalón largo, mientras que el avatar masculino no utilizará prenda superior y el avatar femenino vestirá un top en color blanco.

Tras completar la estética de los personajes, `Fuse Character Creator de Mixamo` ofrece la posibilidad de configurar los huesos de los avatares para

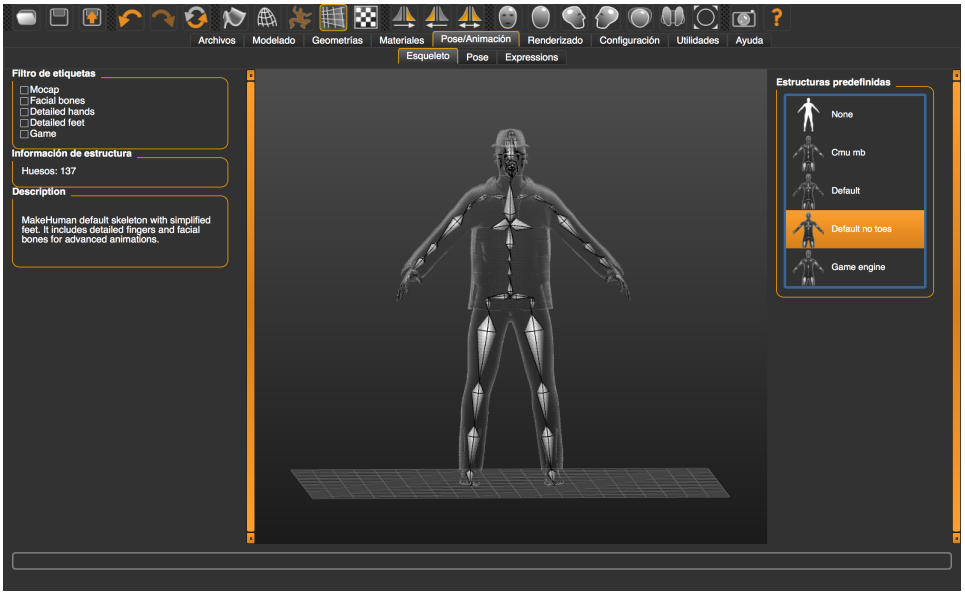


Figura 5.7: Avatar con 137 huesos

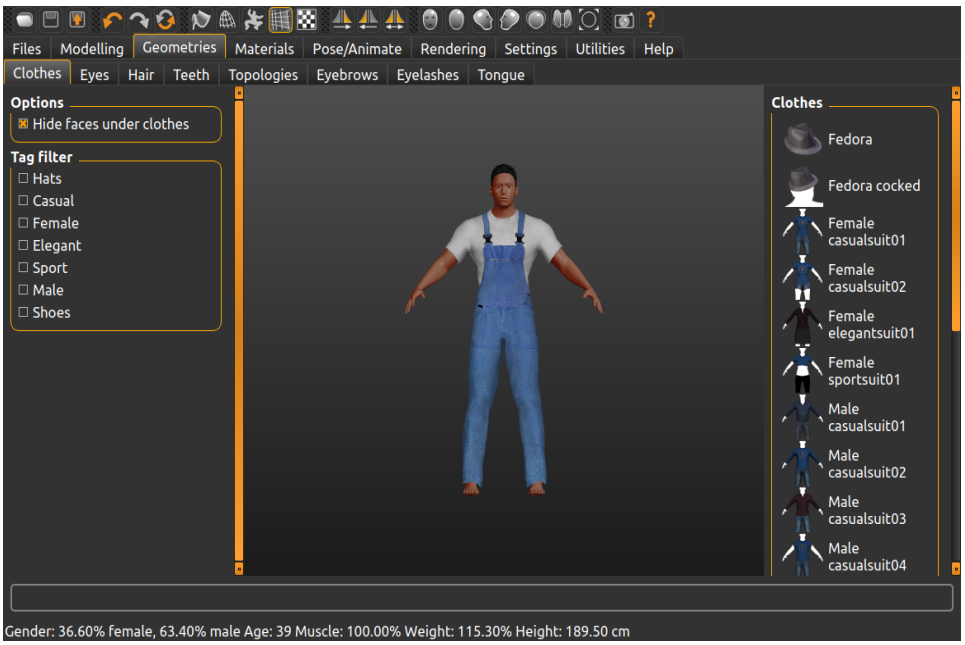


Figura 5.8: Avatar inicial con ropa

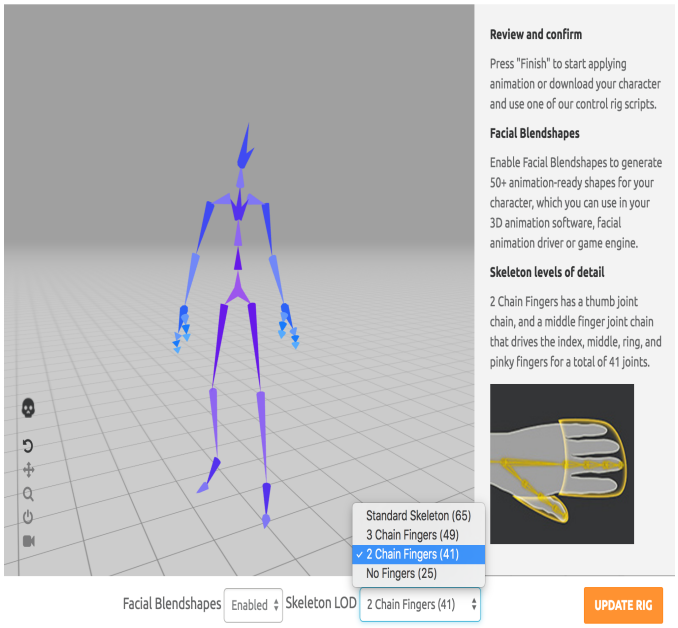


Figura 5.9: Auto-Rig de Mixamo

posteriormente utilizar las animaciones que suministra `Mixamo`. Para ello será necesario subir a los servidores de `Mixamo` dichos personajes y cuando se complete el proceso, la aplicación redirigirá su actividad al navegador web. Como ocurría con `MakeHuman`, `Mixamo` ofrece la posibilidad de crear varios esqueletos para un mismo personaje, en este caso se presentan cuatro casos con 25, 41 (ver Figura 5.9), 49 y 65 huesos respectivamente.

Una vez comprendida la generación de avatares. El avatar elegido que mejor se adapta a `Kinect` es el que contiene 41 huesos, ya que demuestra una movilidad más semejante al *cubeman* inicial.

Con el propósito de utilizar las animaciones que suministra `Mixamo` se observó el ejemplo `AvataresDemo`, el cual utiliza un script `AvatarController` que realiza una conexión entre los *joints* y los huesos incorporados en el avatar.

Posteriormente para que los avatares no se queden en una posición estática, se incorporaron las animaciones de la ginga, victoria y derrota para obtener un feedback mas ilustrado de la comparación de movimiento.

Comentario: tres fotos ginga victoria y derrota

Para incorporar las animaciones al avatar, se crea el componente `Animator Controller` de *Unity*, encargado de añadir y gestionar animaciones. Este componente contiene una máquina de estados capaz de coordinar que animación se ejecuta en cada momento. El estado inicial será la *ginga*, y se

realizará una transición a la animación de victoria o derrota que dependerá de la comparación del movimiento del usuario con el del profesor. Una vez determinado el estado, este volverá a la animación de *ginga* para continuar con el siguiente movimiento.

COMENTARIO: imagen Grafo animator

Se constató que existía un conflicto entre el `Animator Controller` y el sensor de movimiento por quien tiene el control del avatar, por ello, se implementó un script que desactiva el componente `Animator Controller` cuando el sensor de *kinect* detecta al usuario y se vuelve a activar cuando el usuario se sale de la escena.

### 5.1.5. Análisis de los movimientos de Usuario

Una parte importante de este proyecto consiste en dar una corrección del movimiento, el cual, el usuario esta imitando en tiempo real. En esta sección se explicará como se implementó este análisis.

#### 5.1.5.1. Investigación base

Para este apartado se observó el desarrollo de la comparación de movimiento implementada anteriormente. Esa sección respondía al usuario dándole un *feedback* inmediato si había realizado el movimiento correctamente o incorrecta, sin embargo , no le detallaba que partes del cuerpo había colocado de manera errónea.

El planteamiento del análisis comienza reproduciendo el movimiento del avatar del usuario junto al avatar de la grabación, deteniéndose ,si llegara el caso, en el *frame* donde el usuario cometió el error. En ese instante cambiará los huesos del avatar controlado por el usuario a un color rojo, en consecuencia de una colocación incorrecta del hueso. Adicionalmente mostrará un panel con un texto describiendo las correcciones que se deben aplicar. Esta información se compondrá de la parte de cuerpo que se está analizando , añadiendo seguidamente, la dirección y la altura que servirán de pautas para que el usuario corrija su posición.

#### 5.1.5.2. Cambios e implementaciones

Una vez planteada las especificaciones del análisis se empieza con su implementación. En primer lugar se creó un *script* llamado `Registrar_movimientos`, el cual, guarda toda la información del movimiento hasta donde el usuario comete el error. Esta información contiene el estado[9] donde se produce el error y los datos de los `Joints` de los avatares. Posteriormente se necesita hacer una modificación funcional en el gesto *Move* de `KinectGesture`, para conseguir

[9]Corresponde al frame que se esta comparando.

así, rellenar esa información. Esta modificación consiste en enviar la información del estado y los *joints* de los avatares al `Registrar_movimientos`, una vez superada la comparación de ese *frame*. Con esta lógica se quedará registrado el movimiento hasta que el usuario cometa el error.

A partir de este momento, para poder mostrar la información recogida se implementó una funcionalidad nueva en el *script* `Registrar_movimientos`, que consiste en desplazar los avatares a primera escena, modificando los materiales para que se vean transparentes ,y crear así un esqueleto verde parecido a los *cubeman* del principio. Estos avatares simularán el movimiento hasta el *frame* donde se equivocó el usuario y realizará una comprobación de cada `Joint` para cambiar el hueso de color de verde a rojo. A la vez, se va generando un *string* con una corrección de posicionamiento compuesto por tres partes. Para la primera parte se crea un diccionario que,a partir del número del `Joint`, devuelve la parte del cuerpo correspondiente. En la implementación de este diccionario se han agrupado `Joints`, ya que varios forman una misma parte del cuerpo, como por ejemplo, las manos que lo forman tres `Joints` diferentes. La segunda parte del *string* esta formada por la elección de la dirección , que se comprueba, realizando la diferencia de la coordenadas X del `Joint` del usuario con la del `Joint` de la grabación. Si la diferencia es positiva la dirección será hacia la derecha, en caso contrario, será hacia la izquierda. Y para la tercera parte del *string* se realiza la diferencia con la coordenada Y definiendo así la altura. Si la diferencia es positiva la altura será hacia abajo , en caso contrario hacia arriba. Esta lista de correcciones se mostrará en un panel una vez completada evitando los *strings* repetidos.

COMENTARIO: Imagen completa de análisis con panel

### 5.1.6. Creación de los escenarios 3D

Para el desarrollo de los escenarios, se han creado dos ambientes diferentes: uno escenificando un entorno de entrenamiento como es el caso de un gimnasio para la visualización del alumno (Figura **??**) y otro más orientado al profesor y los animadores, como se trata de una playa Brasileña (Figura **??**).

#### 5.1.6.1. Escenario de gimnasio

La creación del escenario de gimnasio se diseñó a partir de diferentes componentes, distribuidos en distintos paquetes y obtenidos en el `Asset Store` de `Unity`. En el diseño de suelo se creó un `Gameobject` de tipo panel y se le incrustó un material, dispuesto por el paquete `Yughues Free Wooden Floor Materials`,[10]para simular un suelo de madera. En la construcción de las pa-

[10]https://www.assetstore.unity3d.com/en/#!/content/13213

redes del escenario se crearon tres paneles, colocándoles de forma vertical formando un cubo abierto, y además se añadió a todos los paneles, un material del paquete `18 High Resolution Wall Textures`[11] que se asemejan a una textura tipo piedra. Se colocan unos bancos y un casillero para dar una sensación de sala de entrenamiento, estos objetos son modelos del paquete `Locker Room Props`[12]. Para no dar una impresión de una habitación cerrada se añadieron dos ventanas y un ventilador , estos objetos vienen implementados en el paquete `Props for the Classroom` [13],a su vez, se observó también que en el mismo paquete venía implementado un objeto con la forma de una pizarra de clase, entonces para agregar una huella educativa se añadió este elemento a escena.

Figura 5.10: Escenario de gimnasio

#### 5.1.6.2. Escenario de isla

### 5.1.7. Grabacion de movimientos con gente experta

En esta etapa del proyecto se visitó la Asociación Cultural Deportiva Rio[14], donde se reúnen los practicantes expertos en capoeira, para realizar una serie de grabaciones y pruebas.

[11] https://www.assetstore.unity3d.com/en/#!/content/12567
[12] https://www.assetstore.unity3d.com/en/#!/content/3355
[13] https://www.assetstore.unity3d.com/en/#!/content/5977
[14] https://www.asociacionrio.com/

#### 5.1.7.1. Movimientos grabados

Este proceso consistía en grabar una batería de movimientos de capoeira Capoeira (2007) para seleccionar los captados correctamente por `Kinect`. Para facilitar la grabación al profesor de capoeira, se optó por capturar varias repeticiones del mismo movimiento en la misma grabación. De este modo, y con la supervisión del profesor, se registraron 20 técnicas de capoeira:

- Ginga, es la técnica fundamental de la capoeira, consiste en realizar un movimiento constante que prepara al usuario para acciones como evadir, fintar o atacar, así como conservar el impulso. La forma general de la ginga es un continuo paso triangular, retrocediendo un pie y luego con el otro en diagonal mientras se mantiene las piernas separadas. Para este movimiento se realizaron tres grabaciones diferentes porque es un movimiento que puede tener diversas variantes.
- Aú, movimiento que consiste en dar una voltereta lateral.
- Armada, técnica en donde el usuario gira sobre si mismo en vertical golpeando con el talón al oponente.
- Queixada, movimiento que consiste en dar un paso quedándose de costado al oponente y se realiza una patada con un movimiento circular.
- Bênção, técnica que que se ejecuta con una patada frontal básica, realizada contra el abdomen o el pecho del oponente.
- Chapa-Pisào, movimiento similar al Bênção pero se realiza con la planta del pie y paralelo al suelo.
- Gancho, técnica que consiste en levantar la pierna en un diagonal hasta una posición alta, para después contraer la rodilla y golpear con el talón en dirección descendente.
- Martelo, movimiento de patada que tiene como objetivo golpear con el empeine en la sien del oponente.
- Meia Lua de Compasso, técnica que consiste en apoyar una mano en el suelo mientras se gira 180º y se lanza la pierna opuesta en un movimiento circular y ascendente, para golpear con el talón en la cabeza del oponente.
- Meia Lua de Frente, movimiento que realiza una patada frontal de fuera a dentro
- Ponteira, movimiento que consiste en estirar la pierna hacia el oponente y golpearlo con la punta del pie.

- Joelhada, movimiento que se realiza alzando la rodilla hacia delante con la intención de golpear.
- Galopante, técnica que realiza un golpe con la mano abierta parecido a una bofetada.
- Telefone, técnica que sirve para golpear con las dos manos en los tímpanos del oponente.
- Despreço, movimiento que consiste en dar una bofeta pero con la parte externa de la mano.
- Escada,
- Esquiva de frente, es un técnica para esquivar un ataque elevado y consiste en desplazar el cuerpo hacia abajo.
- Negativa, es una posición defensiva utilizado para esquivar o para desplazarse por el suelo.

COMENTARIO: Foto de alguien grabandole

#### 5.1.7.2. Elección y ajuste de movimientos

Después del proceso de grabación ,se tuvieron que seleccionar los movimientos que mejor se asemejaban al real, recortando el intervalo de *frames* más adecuado para incluirlo en la aplicación. De los 20 movimientos grabados los seleccionados fueron :Bênção, Ginga, Joehjada, Escada, Galopante, Meia Lua de Frente, Chapa-Pisào y Punteira. Estos 8 movimientos fueron factibles para su uso porque el resto, al incluir giros, presentaron problemas en la grabación. Esto se debe a que `Kinect` sólo posee una cámara frontal, de modo que al rotar alrededor del eje vertical se acaba perdiendo la referencia de cuáles son las extremidades izquierda y derecha. Como resultado, al darse la vuelta, la captura vuelve a mostrar al usuario de frente, pero con los brazos y las piernas cruzadas.

# Capítulo 6

# Métodos

## 6.1. Escenas

En `Unity`, como se ha mencionado anteriormente, las escenas contienen los objetos del juego. Se pueden usar para crear un menú principal, animaciones, niveles, etc. En cada escena se ha creado un entorno diferenciado para cada una de las necesidades dadas.

Este entorno está pensado para ser utilizado por el alumno para que pueda aprender a realizar diferentes movimientos de capoeira sin la necesidad de que esté presente un profesor. El entrenamiento consiste en imitar una serie de movimientos, los cuales han sido previamente capturados por varios expertos en el arte marcial. El entrenamiento virtual se desarrolla acorde al nivel que posea el alumno, ya sea principiante, aprendiz o avanzado, de este modo, el alumno va aprendiendo a realizar los movimientos de forma progresiva.

Gracias a que el sistema compara *frame* a *frame* la posición del alumno con la del profesor virtual, este podrá realizar un análisis de las transiciones en las que está cometiendo algún error.

Tras realizar una abstracción del flujo de ejecución que realiza el entrenador virtual entre las diferentes escenas, se muestra el diagrama de la lógica del sistema en la figura 6.1

### 6.1.1. Escena 1: Inicio

Se trata de la escena inicial del juego. Presenta una interfaz de tipo UI, la cual contiene el nombre de la aplicación y una breve descripción. Además se muestran los tres emblemas de las diferentes organizaciones presentes, como son la UCM, FDI y Abadá-Capoeira.

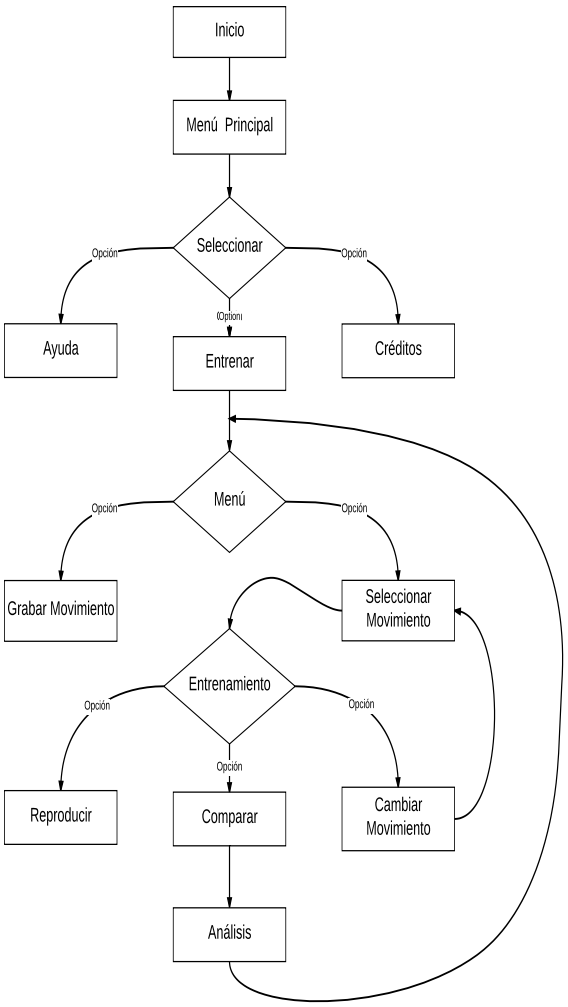


Figura 6.1: Diagrama que ilustra la lógica del sistema

### 6.1.2. Escena 2: Menú Principal

Tras pulsar el botón de inicio en la anterior escena, pasamos al menú principal, en el cual se puede decidir si se quiere empezar a entrenar, visualizar una ayuda sobre como utilizar la aplicación, los créditos o volver a la escena de inicio.

### 6.1.3. Escena 3: Ayuda

Esta escena muestra varios mensajes de ayuda en los que se explica detalladamente el funcionamiento de la aplicación, además de como obtener un mayor rendimiento, ya que serán necesarias una pequeñas pautas para su utilización. También se resolverán las diferentes dudas que le puedan surgir al alumno.

### 6.1.4. Escena 4: Créditos

Al igual que en la escena de ayuda, solo se podrá acceder desde el menú principal. Se muestran los créditos sobre los desarrolladores, organismos implicados, agradecimientos y las licencias utilizadas en el proyecto.

### 6.1.5. Escena 5: Entrenar

Tras seleccionar el tipo de usuario que utilizará la aplicación en la escena anterior, se desplegarán dos secciones diferentes.

#### 6.1.5.1. Profesor

Cuando el usuario inicia el sistema como profesor, puede acceder a la funcionalidad de grabación de nuevos movimientos. Al comenzar a realizar la captura es necesario introducir el nombre deseado para el nuevo movimiento y colocarse a unos tres metros de `Kinect` para ser capturado correctamente. A partir de ese momento, el sistema graba los movimientos realizados por el usuario y los guarda en ficheros de texto para poder utilizarlos tanto para mostrárselos a los alumnos como para compararlos con los movimientos de estos cuando se encuentren practicando. Con el fin de de obtener una mayor inmersión en la aplicación, se utiliza el modo espejo como efecto para reproducir los movimientos captados.

El profesor también puede utilizar la funcionalidad del alumno, ya que será necesario revisar y reproducir el correcto funcionamiento de los nuevos movimientos capturados con `Kinect`. Inicialmente, la aplicación incluye ocho movimientos capturados a instructores de la escuela `Abadá-Capoeira`.[1]

#### 6.1.5.2. Alumno

Si el usuario es un alumno, se entrará en la escena principal del proyecto, donde se desarrolla toda la funcionalidad del entrenamiento virtual. En el interior de la escena se muestra la misma interfaz que en el caso del profesor, con la diferencia de que este no podrá grabar movimientos para su posterior análisis.

[1] http://www.abadacapoeira.com.br/

Considerando que al comenzar a formarse en un arte marcial se desconocen los nombres de los movimientos que se desean poner en práctica, se le ofrece al alumno la posibilidad de visualizar previamente los movimientos antes de empezar la práctica para poder seleccionar más fácilmente el que desea. Para iniciar el entrenamiento acorde a su experiencia, el alumno tendrá que seleccionar el nivel que le corresponda, ya sea principiante, aprendiz o avanzado.

Tras elegir el nivel, se pasará a la selección del movimiento que se quiere aprender o practicar, de este modo, el alumno podrá comenzar a entrenar acorde a su nivel.

Posteriormente a la selección del movimiento, el alumno deberá iniciar la comparación mediante el botón correspondiente. Al pulsar sobre el botón, el alumno deberá adoptar una pose inicial correspondiente al movimiento seleccionado previamente. Hasta que el alumno no tome esa misma pose, el sistema estará esperando la calibración. En el instante en que la pose del alumno sea la misma que la del entrenador virtual, el sistema estará calibrado y empezará una cuenta atrás de tres segundos, tras la cual se empezará a reproducir el movimiento seleccionado.

En el momento en que el entrenador virtual inicie una reproducción de un movimiento, el alumno deberá imitarlo lo más fielmente posible. Mientras se realizan las transiciones, el sistema comparará las posición de los 25 *joints* que componen el esqueleto del entrenador virtual, con las posiciones de los 25 *joints* correspondientes al alumno capturado por `Kinect`.

Tras finalizar el movimiento, el asistente mostrará una animación de éxito en el caso de que el movimiento se haya realizado de forma correcta (ver Figura **??**), o una animación de fracaso en el caso de que se produzca algún error en la ejecución del movimiento. Posteriormente, se presenta un esqueleto en el que se muestran de color rojo las partes del cuerpo cuya colocación debe ser corregida, y en color verde las que se han utilizado de manera correcta (ver Figura **??**). A demás, se visualizará una serie de indicaciones en forma de texto para que el alumno pueda corregir la posición errónea de una forma mas efectiva (ej. *Gira el brazo izquierdo hacia atrás*).

# Capítulo 7

# Conclusiones y Trabajo futuro

## 7.1. Conclusiones

A lo largo del presente artículo se ha descrito el desarrollo de un entrenador virtual de capoeira basado en la captura de los movimientos del usuario a través de `Kinect` y en su comparación con los movimientos grabados previamente de un profesor de capoeira, con la consiguiente explicación de cómo conseguir realizar correctamente los movimientos ejecutados de manera errónea.

Aunque aún no se ha realizado una evaluación extensa con practicantes novatos y expertos de capoeira, sí se han realizado pruebas preliminares para comprobar que la aplicación funciona correctamente y que los comentarios proporcionados para corregir los errores son adecuados. Estas pruebas se han realizado con una muestra pequeña de usuarios de distintos niveles y los resultados preliminares muestran que, al nivel al que se han realizado las pruebas, y con movimientos no excesivamente complejos de cuerpo entero, el funcionamiento de la aplicación es rápido y los resultados de los análisis y las explicaciones de los errores se acercan bastante a la realidad, por lo que actualmente nos hallamos preparando una evaluación más extensa de la aplicación.

## 7.2. Trabajo futuro

Este proyecto tiene potencial en el campo de entrenamiento de actividades físicas ya que, analizando el movimiento del usuario, le ofrece un feedback inmediato basándose en el movimiento de un experto, y así transmitirle una mejora de su actividad. Además tiene un alcance en el ámbito doméstico ya que el los dispositivos necesarios son productos asequibles y baratos.

Una funcionalidad que se puede implementar en un futuro sería la gestión de diferentes usuarios y así tener un histórico de la progresión de los

movimientos. Cuando el usuario acceda con su cuenta en la aplicación le aparecerá el porcentaje con el avance de cada movimiento.

Otra característica que se propone es añadir más funcionalidad al experto que graba los movimientos. Con ello tendrá los privilegios de tratar y modificar los movimientos grabados ajustando y recortado sus *frames*.

Por otro lado, se plantea también la necesidad de incluir métodos más sofisticados para el análisis de los movimientos, similares a los usados en Keerthy (2012); **?** para permitir un análisis más fino de los movimientos que pueda proporcionar resultados de mayor utilidad para practicantes avanzados de capoeira.

Este procedimiento de análisis de movimiento se puede aplicar también en el ámbito de la medicina, para realizar diferente tipos de rehabilitación. Por ejemplo, para comunicarle a un paciente que tenga una lesión en un brazo ,si el movimiento que realiza para su rehabilitación, lo está ejecutando de forma correcta o incorrecta.

Otro factor para mejorar la captura de movimiento sería añadir otra kinect para lograr reconocer los movimientos que impliquen giros. Esta solución similar se plantea en Gao et al. (2015) donde utilizan dos `Kinect` para solucionar este problema, aunque esta opción aleja la solución de su uso doméstico.

Finalmente, también se contempla el análisis de los movimientos de más de un usuario de manera simultánea, de forma que se puedan reproducir situaciones similares al trabajo por parejas que tiene lugar en una *roda* de capoeira, donde se ponen en práctica técnicas de combate cuerpo a cuerpo.

## 7.3. Conclusions

# Parte I

# Apéndices

# Apéndice A

# Así se hizo...

...

...

**Resumen:** ...

## A.1. Introducción

...

# Bibliografía

*Y así, del mucho leer y del poco dormir,*
*se le secó el celebro de manera que vino*
*a perder el juicio.*

Miguel de Cervantes Saavedra


Capoeira, N. *The little Capoeira book*. Blue Snake Books, 2007.

Games, E. *Unreal - Game Engine Technology. [online] . [Accessed 07 June 2017]..* https://www.unrealengine.com/en-US/blog, ????

Gao, Z., Yu, Y., Zhou, Y. y Du, S. Leveraging two kinect sensors for accurate full-body motion capture. *Sensors*, vol. 15(9), páginas 24297–24317, 2015.

Keerthy, N. K. *Virtual Kung fu Sifu with Kinect*. Proyecto Fin de Carrera, San José State University, CA, USA, 2012.

Microsoft. *Microsoft Visual Studio. [online] . [Accessed 07 June 2017]..* https://www.visualstudio.com/, ????

Rodríguez-Esparragón, D. y Domínguez Quintana, L. Solución de bajo coste de captura de movimiento basada en kinect. ????

Technologies, U. *Unity - Game engine, tools and multiplatform. [online] . [Accessed 07 June 2017]..* https://unity3d.com/es/unity, ????a.

Technologies, U. *Unity - Game engine, tools and multiplatform. [online] . [Accessed 07 June 2017]..* https://unity3d.com/es/public-relations, ????b.

Technologies, U. *Unity - Manual: Unity Manual. [online] . [Accessed 07 June 2017]..* http://docs.unity3d.com/Manual/index.html, ????c.

Wikipedia. Captura de movimiento — wikipedia, la enciclopedia libre. 2016. [Internet; descargado 20-marzo-2017].

*–¿Qué te parece desto, Sancho? – Dijo Don Quijote –*
*Bien podrán los encantadores quitarme la ventura,*
*pero el esfuerzo y el ánimo, será imposible.*

*Segunda parte del Ingenioso Caballero*
*Don Quijote de la Mancha*
*Miguel de Cervantes*

*–Buena está – dijo Sancho –; fírmela vuestra merced.*
*–No es menester firmarla – dijo Don Quijote–,*
*sino solamente poner mi rúbrica.*

*Primera parte del Ingenioso Caballero*
*Don Quijote de la Mancha*
*Miguel de Cervantes*