

---

# Uso de Kinect para el entrenamiento de actividades físicas

---



## Trabajo de Fin de Grado

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+.

# Uso de Kinect para el entrenamiento de actividades físicas

*Trabajo de Fin de Grado*  
**Ingeniería del Software e Inteligencia Artificial**

*Dirigido por los Doctores*  
**Pablo Gervás Gómez-Navarro, Gonzalo Méndez Pozo**

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

**Junio 2017**



# Dedicatoria

*A nuestras familias por darnos todo y apoyarnos para conseguir lo que nos propongamos*

*A nuestras novias, Valentina y Ana, por su apoyo y siempre alentarnos a ser los mejores*

*A todos nuestros amigos que nos han acompañado y ayudado todo este tiempo*

*A todos ellos les dedicamos este trabajo porque sin ellos no habría sido posible*



# Agradecimientos

Quisiéramos agradecer a Gonzalo Méndez Pozo y a Pablo Gervás Gomez-Navarro, los Directores de este Trabajo, todo el apoyo que nos han proporcionado para la realizarlo.

Marco Antonio Gomez-Martin, Pedro Pablo Gomez-Martin por la plantilla TeXiS de Latex.

También nos gustaría agradecer a los miembros de nuestra familia, a nuestros padres y amigos que nos han mostrado su apoyo y ánimos durante la creación de este trabajo.

Ana Galindo Lobato

Por último, quisieramos agradecer a la Associaçao Brasileira de Apoio e Desenvolvimento da Arte-Capoeira(ABADÁ-Capoeira), a la Asociación Cultural Deportiva Rio<sup>1</sup>, al instructor Gato<sup>2</sup> y al graduado Windows<sup>3</sup> por enseñarnos, ayudarnos y ofrecerse para las grabaciones necesarias de este proyecto.

---

<sup>1</sup><https://www.asociacionrio.com/>

<sup>2</sup><http://www.abadacapoeiragato.com/>

<sup>3</sup><http://dsa-research.org/jlvazquez/>



# Resumen

Este proyecto tiene como meta el diseño y desarrollo de un entrenador personal de capoeira con la tecnología de captura de movimiento de Kinect. A este sistema creado, se le ha dado el nombre de VIC (*Virtual Instructor of Capoeira*).

Para entender y abordar de una manera correcta este trabajo, se han revisado los diferentes sistemas y tecnologías que existen para la captura de movimientos. Se han expuesto las ventajas que han llevado a utilizar el dispositivo Kinect y se han mencionado varios estudios de investigación previos con resultados favorables. Además, se ha hecho hincapié en las novedades que aporta este proyecto en el ámbito de los *Reactive Virtual Trainers (RVT)*, ya que ha dado lugar a una diversidad de soluciones para la danza, las artes marciales y el ejercicio físico en general.

Este proyecto presenta un sistema de entrenamiento personal de capoeira, un arte marcial brasileño, definido como una mezcla entre los deportes de lucha y la danza. Este entorno está pensado para que el alumno 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 realizados por expertos y que han sido previamente grabados. Además, se puede escoger el nivel del alumno, ya sea principiante o avanzado. De este modo, el alumno va aprendiendo a realizar los movimientos de forma progresiva.

Este sistema de bajo coste e intuitivo lleva a cabo un seguimiento del alumno en cada movimiento. Así, el sistema se podrá adaptar fácilmente para, por ejemplo, la rehabilitación de alguna parte del cuerpo dañada o el análisis de los atletas en la medicina deportiva. Tras el estudio realizado, se observó que el funcionamiento de la aplicación es rápido y ofrece unas correcciones muy precisas de los movimientos ejecutados de forma errónea. Por todo ello, se puede afirmar que este entorno tiene un excelente presente y un futuro muy prometedor.

**Palabras clave:** Kinect, capoeira, captura movimiento, Mocap, Reactive Virtual Trainer, RVT, Unity 3D, entrenador personal, comparación y corrección de movimientos.

# Índice

<b>Dedicatoria</b>	<b>V</b>
<b>Agradecimientos</b>	<b>VII</b>
<b>Resumen</b>	<b>IX</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Objetivos del trabajo . . . . .	2
1.2. Plan de Trabajo . . . . .	3
1.3. Estructura de la memoria . . . . .	4
<b>2. Estado del arte</b>	<b>5</b>
2.1. Captura de movimiento . . . . .	5
2.2. Historia de la captura de movimiento . . . . .	5
2.2.1. Precursores . . . . .	5
2.2.2. Nacimiento de la captura de movimiento . . . . .	6
2.3. Métodos de captura de movimiento . . . . .	9
2.3.1. Captura de movimiento mecánica . . . . .	9
2.3.2. Captura de movimiento electromagnética . . . . .	10
2.3.3. Captura de movimiento óptica . . . . .	11
2.3.4. Captura de movimiento mediante fibra óptica . . . . .	13
2.3.5. Captura de movimiento mediante ultrasonidos . . . . .	13
2.3.6. Captura de movimiento mediante sistemas inerciales .	14
2.3.7. Captura de movimiento ocular . . . . .	14
2.4. Métodos de captura de movimiento en los videojuegos . . . .	14
2.4.1. Nintendo, Wii Remote . . . . .	15
2.4.2. Sony, PlayStation Move . . . . .	16
2.4.3. Microsoft, Kinect . . . . .	18
2.5. Trabajo relacionado . . . . .	20
2.5.1. Captura de movimiento en la danza . . . . .	20
2.5.2. Captura de movimiento en artes marciales . . . . .	21
2.5.3. Captura de movimiento en actividades físicas . . . . .	21

2.5.4. Captura de movimiento en rehabilitación . . . . .	22
<b>3. Tecnologías empleadas</b>	<b>25</b>
3.1. Software empleado . . . . .	25
3.1.1. Entorno de desarrollo : Unity 3D . . . . .	25
3.1.2. MakeHuman . . . . .	28
3.1.3. Fuse Character Creator . . . . .	29
3.1.4. Mixamo 3D Animation Online Services . . . . .	30
3.1.5. Marvelous Designer . . . . .	31
3.1.6. Blender . . . . .	31
3.2. Hardware empleado . . . . .	31
3.2.1. Kinect versión 2 . . . . .	31
3.2.2. Adaptador de Kinect para Windows . . . . .	33
3.2.3. Equipo para el uso de Kinect . . . . .	33
<b>4. Desarrollo Del Proyecto</b>	<b>35</b>
4.1. Paquetes para Unity . . . . .	36
4.2. Grabar y reproducir movimientos de Usuario . . . . .	38
4.2.1. Investigación base . . . . .	38
4.2.2. Implementacion de la Grabación y reproducción de los movimientos . . . . .	40
4.3. Comparación de los movimientos de Usuario . . . . .	41
4.3.1. Investigación base . . . . .	41
4.3.2. Implementación de la comparación de los movimientos	42
4.4. Creación de avatares y animaciones . . . . .	42
4.4.1. Investigación base . . . . .	42
4.4.2. Implementación de los avatares y animaciones . . . . .	44
4.5. Análisis de los movimientos del Usuario . . . . .	48
4.5.1. Investigación base . . . . .	48
4.5.2. Implementación del análisis de los movimientos . . . . .	48
4.6. Creación de los escenarios 3D . . . . .	50
4.6.1. Escenario del gimnasio . . . . .	50
4.6.2. Escenario de la isla . . . . .	50
4.7. Grabacion de movimientos con gente experta . . . . .	52
4.7.1. Movimientos grabados . . . . .	52
4.7.2. Elección y ajuste de los movimientos . . . . .	53
4.8. Transición de las escenas . . . . .	54
4.8.1. Escena de Inicio . . . . .	56
4.8.2. Escena de Menú Principal . . . . .	56
4.8.3. Escena de Ayuda . . . . .	56
4.8.4. Escena de Créditos . . . . .	56

4.8.5. Escena de Profesor . . . . .	56
4.8.6. Escena de Alumno . . . . .	57
4.9. Conclusiones del desarrollo del proyecto . . . . .	58
<b>5. Discusión, Conclusiones y Trabajo futuro</b>	<b>59</b>
5.1. Discusión . . . . .	59
5.2. Conclusiones . . . . .	60
5.3. Trabajo futuro . . . . .	60
<b>6. Distribución de trabajo</b>	<b>63</b>
<b>I Apéndices</b>	<b>67</b>
<b>A. Así se hizo...</b>	<b>69</b>
A.1. Introducción . . . . .	69
<b>Bibliografía</b>	<b>71</b>



# Índice de figuras

2.1. Sistema de captura de movimiento mecánico . . . . .	9
2.2. Sistema de captura de movimiento electromagnético . . . . .	10
2.3. Sistema de captura de movimiento óptico con indicadores activos	12
2.4. Sistema de captura de movimiento óptico con indicadores pasivos . . . . .	12
2.5. Gafa ETG-2w usada para la captura de movimiento ocular .	15
2.6. Mando inalámbrico Wii Remote . . . . .	16
2.7. Barra sensor de la videoconsola Wii . . . . .	17
2.8. Dispositivo PlayStation Move . . . . .	17
2.9. Dispositivo PlayStation Eye . . . . .	18
2.10. Dispositivo Kinect v2 . . . . .	19
3.1. Logo Unity . . . . .	26
3.2. Apariencia de editor Unity . . . . .	27
3.3. Logo MakeHuman . . . . .	28
3.4. Aspecto del humano estándar de MakeHuman . . . . .	29
3.5. Herramienta verificadora de Kinect . . . . .	32
3.6. Adaptador de Kinect para Windows . . . . .	33
4.1. Cubeman con la lista de todos los Joints . . . . .	38
4.2. Patrón de Singleton . . . . .	39
4.3. Fichero de texto con los datos del BodyData . . . . .	40
4.4. Escena con los dos cubeman . . . . .	41
4.5. Grafo de comparación de movimiento . . . . .	43
4.6. Avatar inicial . . . . .	44
4.7. Avatar con 137 huesos . . . . .	45
4.8. Avatar inicial con ropa . . . . .	45
4.9. Auto-Rig de Mixamo . . . . .	46
4.10. Diagrama de estados de animator . . . . .	47
4.11. Análisis de movimiento . . . . .	49
4.12. Escenario de gimnasio . . . . .	51

4.13. Escenario para el profesor y los animadores . . . . .	51
4.14. Realizando las grabaciones de movimiento . . . . .	54
4.15. Diagrama que ilustra la lógica del sistema . . . . .	55

# Índice de Tablas



# Capítulo 1

## Introducción

En la actualidad los sistemas de captura de movimiento se han convertido en una herramienta imprescindible en la industria del cine y de los videojuegos, ya que facilitan mucho la labor de los animadores y desarrolladores a la hora de crear personajes. Con esta tecnología se generan unos movimientos más realistas y precisos en menos tiempo, con un coste total inferior a los sistemas antiguos. Estos sistemas pueden ser particularmente útiles no solo en los dos ámbitos anteriormente mencionados, sino también para la realización de diagnósticos médicos, estudio de prototipos de máquinas, análisis de los atletas en la medicina deportiva, recuperación de las capacidades motrices de gente discapacitada, etc. Todo ello se puede realizar con una herramienta mocap de uso doméstico a bajo coste y sin la necesidad de utilizar trajes especiales, como es el caso de Kinect.

Los Reactive Virtual Trainers (RVT) son una tecnología que se viene utilizando desde hace algunos años para la danza, las artes marciales y los ejercicios físicos en general. Teniendo en cuenta esta funcionalidad, se decidió unir la práctica del arte marcial afro?brasileño, conocido como capoeira, con el sistema de captura de movimientos de Kinect, dando lugar al desarrollo de un entrenador personal en dicho deporte. Todo ello ha sido posible gracias a la disposición de varios profesionales de la escuela Abadá?Capoeira que nos han permitido grabarles para el presente proyecto.

Una de las características fundamentales de la capoeira, como de muchas otras artes marciales, es la realización de manera repetida de un mismo movimiento para ir mejorando paulatinamente su ejecución, tanto en su trayectoria como en su velocidad o 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 ma-

yor parte de las ocasiones. De este modo, la aplicación desarrollada sobre un entrenador personal solventaría esta carencia.

Con el objetivo de realizar un entrenador personal de una forma precisa y exhaustiva, primero se realizaron varias visitas a la Asociación Cultural Deportiva Rio, donde se reúnen los expertos en capoeira, para determinar qué movimientos serían capturados de una forma óptima por Kinect. Posteriormente se realizaron diversas grabaciones de los movimientos escogidos para incorporar las diferentes técnicas de capoeira al entrenador virtual.

Los estudios realizados sobre los RVT determinan que los usuarios ponen más atención en el objeto que se mueve en pantalla, la música, los colores, los movimientos, etc, que con un entrenador personal cotidiano. Por ello, en los entrenamientos realizados con tecnología RVT se ha observado una evolución mayor, ya que existe más competitividad por realizar los movimientos con una técnica más adecuada. Otra ventaja destacable sobre el uso de Kinect con un entrenador personal es poder utilizar esta tecnología en cualquier sitio, ya sea en el hogar, una residencia o un hospital para realizar algún tipo de rehabilitación, entre otros. Por lo tanto, se posibilita su uso a todo tipo de públicos, sin la necesidad de tener que desplazarse hasta el lugar donde habitualmente se realiza la actividad.

Con el fin de conseguir un entorno más realista y dinámico, se desarrollaron diferentes avatares con una apariencia bastante humana. Además, se elaboraron dos escenarios diferentes, ambientados cada uno de ellos según su propósito. El primero de ellos escenifica un entorno de entrenamiento, como es el caso de un gimnasio. El segundo, está orientado hacia el profesor y los animadores, representando una playa Brasileña. Con todo ello, se logró crear una aplicación con la que el usuario sentía estar en un entorno más real.

## 1.1. Objetivos del trabajo

Los sistemas RVT desarrollados mediante mocap se centran esencialmente en la representación realista de cómo ayudar a los usuarios a formarse y perfeccionar distintas técnicas. Para ello, se basan en la comparación entre el movimiento realizado por ellos mismos y movimientos grabados previamente por un experto. Siguiendo estas pautas, se determinaron unos objetivos que cumplir en este Trabajo de Fin de Grado (TFG):

- Identificar las limitaciones que presenta el dispositivo **Kinect** para lograr una captura de movimientos de capoeira de una forma correcta.
- Implementar y diseñar una aplicación, con la ayuda de un motor de juego, que simule un entrenador virtual para que realice las correcciones de los movimientos de los usuarios.
- Realizar un análisis de los movimientos de capoeira capturados con

Kinect y lograr así, clasificar cuáles han sido los movimientos mejor registrados para el entrenamiento.

## 1.2. Plan de Trabajo

Este proyecto se ha elaborado en tres fases: especificación, implementación y documentación del proyecto.

En la primera fase se establecieron cuales fueron los objetivos y el alcance del TFG, siendo necesario organizar fechas para concretar reuniones con los tutores y determinar, así, el seguimiento del desarrollo del proyecto.

Posteriormente, en la segunda fase, se ponen en práctica las especificaciones planteadas en la fase anterior. Esta fase se divide a su vez en tres pasos: el primero fue la elección del entorno a utilizar, de modo que se adecue a la tecnología de Kinect y se ajuste a nuestra forma de trabajo; el segundo paso consistió en crear la aplicación con su entorno 3D, añadiendo la lógica comparativa del movimiento e incorporando diferentes avatares; y, en el último paso, una vez acabada la aplicación, se realizaron las grabaciones con gente experta en capoeira, analizando y ajustando sus movimientos para incorporarlos al entrenador virtual.

Finalmente, en la fase de documentación, se recopiló toda la información y contenido necesario para la elaboración del Trabajo de Fin de Grado. Esta fase se ejecuta paralelamente a la fase de implementación.

A continuación se muestra una lista con las tareas de cada fase y sus fechas estimadas.

- Especificación del proyecto
  - Definición del proyecto : 2015 - julio 2016.
  - Plateamiento de los objetivos : diciembre 2016 - enero 2016.
  - Reuniones semanales con los tutores : diciembre 2016 - junio 2017
- Implementación del proyecto
  - Elección del entorno de desarrollo : diciembre 2016.
  - Creación del entrenador virtual : febrero 2017 - mayo 2017
  - Diseño de avatares : abril 2017
  - Creación de escenarios 3D : mayo 2017
  - Grabación con gente experta : mayo 2017
- Documentación del proyecto
  - Documentación para el proyecto : abril 2017 - junio 2017
  - Realización de la memoria : mayo 2017 - junio 2017

### 1.3. Estructura de la memoria

Este trabajo se estructura en seis capítulos definidos de la siguiente manera:

- En el capítulo uno se describe la introducción al proyecto, los objetivos principales y el plan del trabajo.
- En el capítulo dos se describe los entrenadores virtuales, la historia de la captura de movimiento y los diferentes métodos de captura de movimiento que existen.
- El capítulo tres engloba las tecnologías utilizadas, tanto hardware como software, y se justifica la toma de decisión de los entornos utilizados.
- En el capítulo cuatro se describe todo el desarrollo del proyecto expli- cando en diferentes secciones el estudio previo y las implementaciones necesarias para la creación del entrenador virtual de capoeira. Además, se explica la transición de escenas en la aplicación.
- En el capítulo cinco se presenta las discusiones, la conclusión obtenida del proyecto y las propuestas de cara al futuro.
- En el capítulo seis se describe la distribución del trabajo por parte de los integrantes de grupo.

# 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. (Wikipedia, 2017).

COMENTARIO: Meter otras referencias

### 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*” (Movimiento de los animales). 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 realmente, 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, etc.

Como pionero en la edad moderna, Eadweard Muybridge (1830-1904) fué 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 en 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 príncipe y Blancanieves.

COMENTARIO: Meter foto de Blancanieves

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 (Massachusetts Institute of Technology) 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 (Organización de entretenimiento norteamericana) 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 (Productora de animación por ordenador), en 1988 los miembros de la compañía encontraron una so-

lución 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 más 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<sup>1</sup> 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 simultánea 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 pasado 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 of Us, Beyond: Two Souls, la saga Uncharted, Until Dawn, L.A. Noire, etc.

COMENTARIO: Imagen de película y videojuego con mocap

<sup>1</sup> Empresa encargada del desarrollo, publicación, venta y distribución de videojuegos para diferentes compañías (Sega, Nintendo, Sony, Microsoft...)

## 2.3. Métodos de captura de movimiento

En la actualidad existen numerosos sistemas para la captura 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

Estos sistemas tienen el problema de que son incapaces de medir translaciones globales. Pueden medir posiciones relativas de los miembros, pero no como se desplaza el actor por el escenario. Además, el único valor que utilizan para medir es el grado de apertura, no teniendo en cuenta rotaciones complejas que poseen las articulaciones humanas, como es el caso de los hombros, cadera, tobillos, etc. También tienen el inconveniente de ser pesados, restringir el movimiento del actor y su corto tiempo de vida. En cambio,

tienen la ventaja de tener un coste relativamente bajo (en torno a 17.000 y 50.000 €), capaz de registrar los movimientos del actor en tiempo real con un alcance mayúsculo.

### 2.3.2. Captura de movimiento electromagnética

Los sistemas de captura de movimiento electromagnética, están formados por sensores creados por tres espirales ortogonales que miden el flujo magnético, determinando tanto la posición como la orientación del sensor. Un transmisor genera un campo electromagnético de baja frecuencia que los receptores detectan y transmiten a la unidad electrónica de control, donde se filtra y amplifica. A continuación, los datos se envían a un ordenador central, donde se deduce la orientación y posición de todos los sensores del escenario.

Un sistema magnético estándar consta de 18 sensores, una unidad de control electrónica y un software para el procesamiento. En cambio, un sistema de última generación puede tener hasta 90 sensores capaces de capturar hasta 144 muestras por segundo, teniendo un coste medio (en torno a 4.000 y 11.000 €).



Figura 2.2: Sistema de captura de movimiento electromagnético

Existen dos tipos de rastreadores electromagnéticos:

- *Flock of Birds de Ascension Technology Corporation*: usa pulsos magnéticos cuadrados.
- *Patriot de Polhemus*: usa campos magnéticos sinusoidales.

Ambos sistemas tienen el inconveniente de producir interferencias con materiales metálicos debido a su conductividad, ya que se crean campos magnéticos que interfieren con el campo magnético del emisor. De ese modo,

se trata de un sistema difícil de transportar a diferentes escenarios. El proceso de captura no es en tiempo real, aunque se aproxima bastante, pese a que tiene un número de capturas por segundo demasiado bajo. Como punto a favor, es más fácil procesar los datos que en otros sistemas *mocap*, ya que los datos obtenidos están en relación con la posición del actor.

### 2.3.3. Captura de movimiento óptica

Los sistemas ópticos emplean los datos recogidos por sensores de imagen para obtener la posición de un elemento en el espacio, utilizando indicadores (en inglés *markers*) pegados al cuerpo del actor, aunque los sistemas modernos permiten recoger los datos rastreando la superficie de forma dinámica.

Estos sistemas permiten la grabación en tiempo real, ya que utilizan un ordenador que recibe la entrada de una o más cámaras digitales CCD (Charge-coupled device) sincronizadas produciendo proyecciones simultáneas. Habitualmente se utilizan entorno a 4 o 32 cámaras, siempre y cuando no se añadan de forma innecesaria, ya que complicarían el procesamiento de la información.

Las cámaras utilizadas en este tipo de sistemas tienen una velocidad de captura de entre 30 y 1.000 fotogramas por segundo. Estos sistemas se deben calibrar mediante el rastreo de un objeto visible, de modo que se calcule la posición de cada cámara respecto de ese punto, en el caso de que una cámara se mueva, será necesario recalibrar el sistema.

Existen varios tipos de sensores para este tipo de captura de movimiento:

#### 2.3.3.1. Indicadores activos

Este sistema está compuesto por leds que emiten su propia luz determinando la posición del actor, de esta manera se consigue aumentar la distancia a la que se puede desplazar el artista. La posición de los marcadores se determina iluminando uno o varios indicadores, de manera sincrónica a las cámaras en cada instante de tiempo, a una frecuencia de muestreo muy alta. De modo que los indicadores deben estar sincronizados con todas las cámaras para realizar una sola captura en cada iluminación.

#### 2.3.3.2. Indicadores pasivos

Estos indicadores, se tratan de bolas de goma recubiertas de un material reflectante que se adhieren al traje del actor en puntos estratégicos. De esta forma, la luz que reflejan se origina cerca de las cámaras, siendo recogida por estas.

COMENTARIO: Coger una imagen



Figura 2.3: Sistema de captura de movimiento óptico con indicadores activos



Figura 2.4: Sistema de captura de movimiento óptico con indicadores pasivos

Este tipo de sistemas pueden capturar un gran número de marcadores a una frecuencia de muestreo de hasta 2000 fotogramas por segundo. Se suelen utilizar principalmente para el registro de movimiento facial. El precio de estos marcadores oscila entre 17.000 y 75.000 €.

### 2.3.3.3. Indicadores activos modulados en el tiempo

Utilizando luz estroboscópica, se mejora el sistema respecto a los indicadores activos estándar, y mediante la iluminación de manera grupal se determina la identidad de cada marcador a través de la frecuencia de destello. De esta manera, se consiguen frecuencias de captura mayores, con el inconveniente de aumentar la carga del sistema. Este tipo de marcadores

permiten observar el resultado en tiempo real, ya que cada indicador es único, eliminando el problema del intercambio de marcadores. Además, este sistema permite su utilización bajo la luz directa del sol, sin que se produzcan interferencias.

Existen sistemas de captura de movimiento compuestos por ocho cámaras de 12 *megapixeles* capaces de capturar hasta 480 fotogramas por segundo. Con un precio inferior a los 38.000 €.

#### **2.3.3.4. Indicadores semipasivos imperceptibles**

A diferencia de los sistemas anteriores, en este caso, son los propios indicadores emisores de luz led los que detectan su propia posición y orientación. Se trata de marcadores que determinan la incidencia de iluminación, con un número ilimitado de etiquetas fotosensibles. Permitiendo colocarse en la ropa u otros objetos, además de trabajar bajo la luz solar. Considerando que no se utilizan cámaras de alta frecuencia, se reduce de forma considerable el tráfico de datos. Por lo tanto, son ideales para la captura de movimientos en tiempo real.

#### **2.3.3.5. Sin marcadores**

Se trata de un sistema en el que no se requiere el uso de trajes especiales para el seguimiento de los movimientos de los actores. Se utilizan algoritmos que permiten identificar la silueta humana mediante el análisis de imágenes, identificando sus formas y descomponiéndolas en segmentos para realizar el seguimiento de sus movimientos. Estos sistemas tienen la dificultad de capturar movimientos sutiles, como los realizados con los dedos o la cara.

### **2.3.4. Captura de movimiento mediante fibra óptica**

Inicialmente, este medio de transmisión se empleó para el desarrollo de unos guantes capaces de capturar el movimiento de los dedos, pero en la actualidad, se han creado trajes idóneos para captar cualquier parte del cuerpo. Con el objetivo de registrar los movimientos, se fijan sensores de fibra óptica flexibles, consiguiendo atenuar la luz transmitida, y así, medir las rotaciones de las articulaciones. Estos sensores no son capaces de medir la posición del actor en el escenario, sino que el sistema calcula la posición de las extremidades capturadas.

### **2.3.5. Captura de movimiento mediante ultrasonidos**

Este sistema utiliza emisores que generan pulsos ultrasónicos (imperceptible por el ser humano) capturados por uno o varios receptores (a modo de radar), permitiendo averiguar la posición del emisor en el escenario, incluso

en algunos casos, su orientación. El inconveniente de esta tecnología se debe a que los emisores son demasiado voluminosos, siendo incapaces de capturar movimientos bruscos de una forma correcta. Aunque en comparación con otros sistema de captura de movimiento, el precio supone una gran ventaja, teniendo un coste de 2500 €.

### 2.3.6. Captura de movimiento mediante sistemas inerciales

Con el objetivo de capturar el movimiento, los sistemas inerciales emplean sensores, como giroscopios y acelerómetros, con el propósito de obtener información sobre la velocidad angular y la aceleración del sensor. La información obtenida de los sensores se transmite a un ordenador, donde se puede observar el movimiento capturado sobre una figura ya animada. Este tipo de sistemas no utiliza mecanismos externos como cámaras; y como en el caso de los sistemas ópticos, cuantos más sensores se utilicen, más real será el movimiento capturado, teniendo unos grandes rangos de captura.

El sistema de Nintendo (Wiimote) utiliza esta técnica, no obstante, para la captura de movimientos se utilizan trajes especiales con otros sensores más precisos, con un precio que varía entre 22.000 y 71.000 €.

### 2.3.7. Captura de movimiento ocular

Estos sistemas utilizan la combinación del seguimiento de los ojos con el de la cabeza, obteniendo unas líneas exactas de la vista, mientras que el usuario es capaz de moverse libremente. De este modo, permite a los atletas, pacientes y otros usuarios desempeñar sus tareas de una forma correcta, sin ningún tipo de impedimento.

Como ejemplo de investigación en esta tecnología, la empresa SMI (Senso Motoric Instruments) ha desarrollado una gafa<sup>2</sup> que permite el seguimiento de los ojos y combinado con la cámara, detecta el punto exacto donde está detenienda la mirada. Soportando el acceso de datos en tiempo real y el control de usuario a través de una conexión inalámbrica.

## 2.4. Métodos de captura de movimiento en los videojuegos

En la actualidad, gracias a la necesidad de crear unos personajes con gestos y movimientos más realistas, el sector de los videojuegos ha conseguido que la captura de movimiento se convierta en una herramienta esencial, ya que permite una mayor inmersión en los videojuegos. Llegando a ser interpretados en la mayoría de juegos deportivos, por jugadores profesionales.

<sup>2</sup><https://www.smivision.com/eye-tracking/product/eye-tracking-glasses/>



Figura 2.5: Gafa ETG-2w usada para la captura de movimiento ocular

En consecuencia, las principales empresas de videojuegos como Nintendo, Sony y Microsoft, han desarrollado los siguientes sistemas capaces de capturar movimientos.

#### 2.4.1. Nintendo, Wii Remote

En el año 2006, Nintendo lanzó Wii (Nintendo, 2006), perteneciente a la séptima generación de videoconsolas, sucesora directa de Nintendo GameCube y compitió con la Xbox 360 de Microsoft y la PlayStation 3 de Sony. La compañía Nintendo afirmó que Wii estaba destinada a un abanico de consumidores mas amplio a diferencia de sus rivales.

La característica más distintiva de la consola, es el mando inalámbrico Wii Remote. Se trata de un dispositivo inalámbrico, capaz de detectar la aceleración en un plano tridimensional mediante la utilización de un acelerómetro ofrece la posibilidad de apuntar y señalar, determinando los movimientos en un plano tridimensional mediante la utilización de un acelerómetro. Además cuenta con un sensor óptico, lo que permite determinar el lugar donde el mando Wii Remote está apuntando.

A diferencia de un mando a distancia tradicional, este sistema detecta la luz led que emite la barra de sensores de la consola. Tiene un tamaño aproximado de 20 cm de longitud y cuanta con diez leds infrarrojos, cinco en cada extremo de la barra. Para un correcto funcionamiento, no es necesario señalar directamente a la barra sensor, pero en caso contrario, perturbaría la capacidad de detección debido al limitado ángulo de visión. La barra sensor emite desde cada uno de los extremos, unos puntos de luz que permite al



Figura 2.6: Mando inalámbrico Wii Remote

Wii Remote ser localizado. A partir de estas dos distancias, el procesador de la videoconsola Wii determina la distancia entre el Wiimote y la barra de sensores utilizando la triangulaci?n.

Los movimientos realizados con Wii Remote, permiten al jugador imitar las acciones reales de juego, como mover una espada, lanzar una bola de fuego, disparar una pistola, etc; en lugar de pulsar solo los botones.

#### 2.4.2. Sony, PlayStation Move

Como consecuencia de los buenos resultados obtenidos por Nintendo, Sony sac? al mercado en 2010, PlayStation Move (Sony, 2010), un sistema de control de movimientos mediante marcadores activos, compatible con las videoconsolas PS3 y PS4. Este sistema incorpora un giroscopio, ya que ninguna tecnolog? de captura de movimientos mediante marcadores, puede determinar si se han producido giros sin realizar una triangulaci?n.

Estas videoconsolas utilizan una camara denominada PlayStation Eye (Sony, 2013) que detecta el color que toma la esfera del dispositivo PlayStation Move, determinando de este modo su posici?n. Esta camara dispone de dos lente, una de ellas realiza la funci?n de camara web y la otra activa la captura de movimientos.

El principal inconveniente de este sistema, es la luz y los colores de fondo. Ya que, se vuelve in?til teniendo una luz de fondo del mismo color que el dispositivo PlayStation Move, perdiendo por completo la posici?n de los



Figura 2.7: Barra sensor de la videoconsola Wii



Figura 2.8: Dispositivo PlayStation Move

marcadores. En la actualidad, es el sistema con menor estabilidad en la captura de movimientos en los videojuegos.



Figura 2.9: Dispositivo PlayStation Eye

#### 2.4.3. Microsoft, Kinect

Por otra parte, en 2010 Microsoft desarrolló **Kinect** (Microsoft, 2010) para Xbox 360, en sus inicios conocido como *Project Natal*. Después de unos años, Microsoft ofreció la posibilidad de conectar **Kinect** por medio de ordenadores o tabletas, con el sistema operativo Windows 8 o superior, mediante un adaptador (Microsoft, 2014a) con una conexión usb 3.0.

El dispositivo **Kinect** es un periférico para videojuegos que prescinde de mandos gracias a un sensor de detección de movimientos. Este sistema reconoce el rostro, la voz, los movimientos y gestos de los jugadores, mediante una cámara conectada a la videoconsola, el ordenador o *tablet*.

Se trata de una barra horizontal conectada a una plataforma diseñada para mantenerse en una posición horizontal. En su interior contiene un hardware compuesto por una cámara RGB, un sensor de profundidad, el proyector de luz infrarroja, un micrófono bidireccional, un firmware y un procesador que utiliza algoritmos para procesar las imágenes tridimensionales. De forma conjunta capturan el movimiento, además de ofrecer un reconocimiento facial y de voz.

En 2013, Microsoft anunció su próxima consola Xbox One y el nuevo sensor de **Kinect** que incluía mejoras sustanciales respecto a la versión anterior. A continuación se destacan las más significativas a nivel de desarrollo. (Microsoft, 2014b)

- **Mayor campo de visión y un seguimiento mejorado del cuerpo**

- Con un campo de visión de 70° en horizontal y 60° en vertical. La versión inicial disponía de 57° y 43° respectivamente.



Figura 2.10: Dispositivo Kinect v2

- Esta nueva versión, permite detectar hasta seis jugadores de forma simultanea con 25 articulaciones por persona dentro de un mismo campo de visión. La versión anterior solo podía detectar dos jugadores y 20 articulaciones.
- Como desventaja, destacar la carencia de motor de inclinación en esta nueva versión de *Kinect*.

- **Mayor resolución con mas detalle**

- Cámara a color capaz de realizar vídeos con una resolución Full HD de 1920 x 1080 ppp. La versión inicial disponía de 640 x 480 ppp.
- Detecta de una manera más precisa el entorno.
- Capacidad de diferenciar la orientación del cuerpo incluyendo sus manos y dedos.
- El *face tracking* tiene mucho más detalle, permitiendo captar los gestos de la cara.
- Mayor calidad de imagen.

- **Mejora el rango de profundidad del sensor**

Se incrementa el rango de actuación, pasando de 0,5 a 4,5 metros.

- **Aumento de la velocidad en el puerto USB 3.0**

La velocidad de la comunicación con el ordenador se transmite mas rápido, disminuyendo la latencia del sensor, pasando de 90 ms a 60 ms.

- **Mejora en la captación de sonidos**

Esta nueva versión de Kinect incluye una gran mejora en cuanto al reconocimiento de voz. De modo que eliminando los ruidos de ambiente, permite captar instrucciones sonoras con mayor detalle.

- **Captación de movimientos a oscuras**

Nuevas funciones de infrarrojos (IR), capaces de reconocer y captar los movimientos aunque la sala este a oscuras. Genera una vista independiente de la iluminación

Kinect tiene dos funcionalidades principales, generar un mapa 3D de la imagen que capta la cámara y realizar un reconocimiento humano en movimiento a partir de diferentes segmentos del cuerpo, además de un esquema en escala de grises del rostro. El periférico, transmite una luz infrarroja a través del escenario, lo que permite conocer el tiempo que tarda la luz en ser reflejada por los objetos. El sistema actúa como un sonar, determinando el tiempo que tarda en reflejarse la luz, se establece la distancia a la que se encuentran los objetos en tiempo real.

Como base para el estudio de la captura de movimientos y las tecnologías disponibles en este ámbito, se han realizado diferentes consultas a los siguientes proyectos: (Alemán Soler, 2014) (Menéndez Mendoza y Rodríguez Marante, 2015) (MEJÍAS, 2014)

COMENTARIO: Meter mas enlaces

## 2.5. Trabajo relacionado

Para la realización de este proyecto se han revisado trabajos anteriores en el área de los *Reactive Virtual Trainers*. En este apartado se describirán los análisis de esos proyectos, y de este modo, poder aclarar y enfocar el desarrollo de este trabajo.

### 2.5.1. Captura de movimiento en la danza

Uno de los campos en donde la captura de movimiento aparece es en la danza, y un claro ejemplo es (Kyan et al., 2015). En este artículo se describe una forma de evaluar y visualizar en tiempo real, los movimientos de la danza de ballet. Para desempeñar esa tarea utilizan como recurso la tecnología de captura de movimiento *kinect*, y gracias a ella, registran los movimientos de bailarines profesionales, de modo que , sean una base para las comparaciones de movimientos. Estos, a su vez , son representados como un *espacio de posturas* en forma de trayectorias de gestos. La evaluación

de la coreografía empezará cuando detecte al bailarín aficionado y con el fin de proporcionar una puntuación para cada coreografía, se compara la posición alineada y velocidad del bailarín aficionado con las correspondientes del bailarín profesional.

### 2.5.2. Captura de movimiento en artes marciales

Otros de los campos donde es útil la captura de movimiento, ya que es necesario la repetición para lograr entender y dominar las técnicas, son en las artes marciales. En este proyecto de máster (Keerthy, 2012) se diseña un maestro virtual de Kung fu con el dispositivo de Kinect. Previamente se graban las técnicas del maestro en el arte marcial, para realizar futuras comparaciones con usuarios aficionados que quieran adentrarse y aprender en el mundo del Kung fu. Aunque existen muchos algoritmos de comparación, en este trabajo se ha elegido, el algoritmo *Dynamic Time Warping*, que utiliza la fórmula distancia euclídea. Una de las principales ventajas del algoritmo *Dynamic Time Warping* es, superar los problemas de análisis de movimiento en velocidad y tiempo. La distancia euclídea se define como la distancia entre dos puntos de un espacio euclídeo, la cual se deduce a partir del teorema de Pitágoras.

La distancia euclídea( $d_e$ ) de dos puntos P(X,Y,Z), Q(x,y,z) viene definida por :

$$d_e(P, Q) = \sqrt{(X_i + x_i)^2 + (Y_i + y_i)^2 + (Z_i + z_i)^2}$$

La fórmula se ejecuta en un espacio de tres dimensiones.

Dentro del mismo ámbito de captura de movimiento en artes marciales, se encuentra este proyecto(Chua et al., 2003), el cual desarrolla un entrenador virtual de Tai Chi. La diferencia con el proyecto de (Keerthy, 2012) se refleja en la forma de comparar el movimiento, debido a que coloca los diferentes movimientos del profesor alrededor del estudiante, de este modo, le ofrece la opción de superponer su cuerpo directamente sobre el movimiento del profesor virtual para efectuarlo correctamente.

### 2.5.3. Captura de movimiento en actividades físicas

Uno de los factores importantes en la ejecución de actividades físicas es el evitar lesión por mala práctica. En esta tesis (Staab, 2014) implementan un entrenador virtual para promover la actividad saludable, y enseñar de una forma correcta, la realización de las ejercicios para prevenir lesiones. Los movimientos del entrenador virtual son generados a partir de la clasificación y el análisis de ejercicios previamente grabados, creando así, un modelo estadístico de cada actividad. Cuando el usuario realiza la tarea se proporcionado

un porcentaje de acierto para la posición, una indicación para su corrección y los grados de rotación que se deben de aplicar. Aunque, realiza una corrección precisa, tiene una decadencia cuando los movimientos son giros sobre el mismo eje.

El siguiente proyecto denominado **OnlineGym** (Paredes et al., 2014) , es un trabajo basado en plataformas de mundos virtuales 3D que permite a los usuarios interactuar mediante el uso de un dispositivo de captura de movimiento, en concreto con **kinect**. Este escenario proporciona la experiencia de una participación conjunta en una sesión de gimnasia grupal. El proyecto está dirigido a usuarios de edad avanzada, que tal vez no puedan participar en sesiones regulares de entrenamiento fuera de sus hogares. Para activar las habilidades motoras y de socialización, se realizan sesiones de *fitness* en grupo para contribuir a su bienestar físico y mental.

Otro proyecto que falta por mencionar es el (Ruttkay y van Welbergen, 2008), el cual desarrolla un entrenador virtual mejorado, que además de presentar los ejercicios físicos a realizar, proporciona un *feedback* en el momento apropiado. Gracias a esta faceta, el *Intelligent Virtual Agent* (IVA) será capaz de introducir y estructurar los diferentes ejercicios. El entrador virtual tendrá una motorización del pulso asociado al usuario para ajustar el ritmo del ejercicio.

#### 2.5.4. Captura de movimiento en rehabilitación

Un punto fuerte para la utilización de la captura de movimiento sería aplicarla para ayudar a la gente con lesiones, y de este modo, realizar terapia de rehabilitación. Con esta idea se ha desarrollado el proyecto (Li et al., 2014), que elabora un entrador virtual para el uso de fisioterapeutas y pacientes en programas de fisioterapia con ejercicios físicos. Permite al terapeuta adaptar los ejercicios a la necesidades de cada paciente de un modo individual. Los pacientes pueden escoger entre diferentes programas y seguir un avatar de entrenamiento, a su vez, los movimientos que simulan los avatares son grabados previamente en función de las necesidades del paciente. Uno de los puntos que sacan en claro en este proyecto es, que los juegos de ordenador han demostrado el potencial para mejorar el apego a la rehabilitación por parte de los pacientes. Del mismo modo, se encuentra que el *feedback* visual ofrecido a los pacientes le hace involucrarse de manera más efectiva con las terapias.

Otro proyecto con una terapia mas específicas es (Sin y Lee, 2013), que estudia los efectos de un entrenador virtual para los paciente con hemiplejía<sup>3</sup>, haciendo énfasis en la función de las extremidades superiores, incluyendo el rango de movimiento, la función motora y la destreza manual. El grupo de usuarios que padecen esta parálisis participaron en diversas pruebas,

<sup>3</sup>Parálisis de un lado del cuerpo causada por una lesión cerebral o de la médula espinal.

en las que se utilizaba la tecnología implementada con **Nintendo Wii** y la tecnología de captura de movimiento de **Xbox** con **Kinect** entre otras. Las pruebas se centraban en juegos que utilizaban movimientos que afectaban las extremidades superiores. Los resultados de los ensayos demuestran una mejoría significativa de los pacientes que realizaron la prueba con la tecnología **kinect**, esto es debido, a que pueden desempeñar toda la tarea sin tener que sujetar un control remoto como pasa en **Nintendo Wii**. Siendo este, un factor añadido para realizar un movimiento con mayor amplitud o ejecutar mas cantidad de movimientos.



# Capítulo 3

## Tecnologías empleadas

En este capítulo se presentan las tecnologías y dispositivos empleadas para el desarrollo del proyecto. Detallando de forma técnica el *software* y *hardware* utilizado para realizar.

En este capítulo se presenta la metodología empleada y los materiales necesarios para realizar este proyecto. A continuación se detallan a nivel técnico los dispositivos y tecnologías empleadas para, posteriormente, pasar a explicar la metodología seguida en el desarrollo del proyecto

### 3.1. Software empleado

El software principal utilizado en este proyecto es Unity 5.5 (Technologies, 2017b), 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, 2017) con el lenguaje de programación C#.

#### 3.1.1. 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 para la creación de escenarios 3D fueron: **Visual Studio 2015** (Microsoft, 2017), **Unity 3D Engine** (Unity-Technologies, 2017) y **Unreal Engine** (Games, 2017). El primero descartado fue **Visual Studio 2015**, ya que se busca el desarrollo de escenarios 3D de una manera mas intuitiva y, tanto **Unity** como **Unreal**, ofrecen herramientas mas 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 3.1: 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 dispositivo( *Technologies, 2017a*). 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 también 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 3.2

Como se observa en la imagen, el editor de **Unity** esta formado por varios

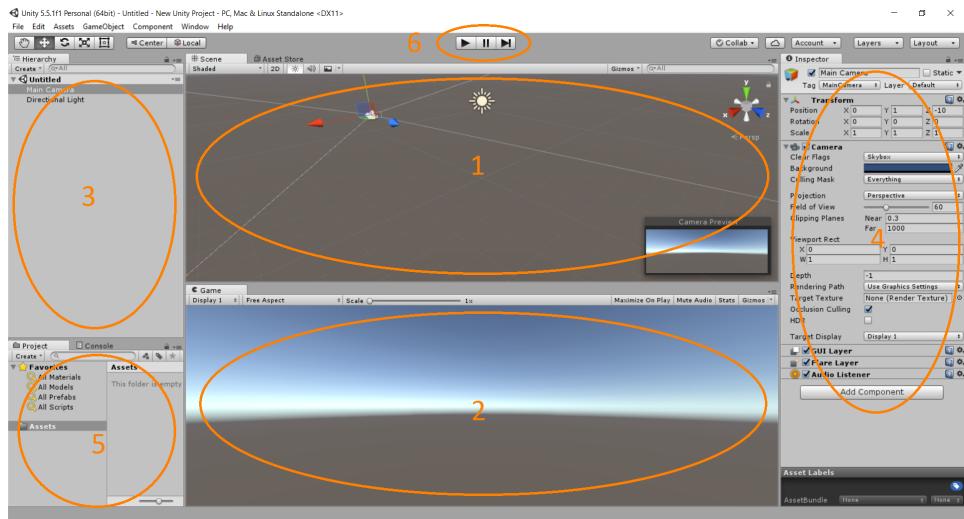


Figura 3.2: Apariencia de editor Unity

paneles. Cabe destacar que la estructuración básica de un juego de **Unity** se realiza mediante escenas y para poder tener una previsualización 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 conseguir de esta manera, 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(Interfaz gráfica de usuario) para la implementación 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, 2017b).

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, 2017) 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 con este entorno de programación se tenía experiencia previa de uso.

El otro programa utilizado para la depuración de *scripts* en **Unity** fue **MonoDevelop**, ya que proporciona un entorno de desarrollo libre y gratuito, diseñado primordialmente para C#.

### 3.1.2. 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.



Figura 3.3: Logo MakeHuman

MakeHuman utiliza un humano estándar y mediante varios deslizadores

o scrolls se podrán alterar los parámetros de la estatura, anchura, sexo, edad, etc.

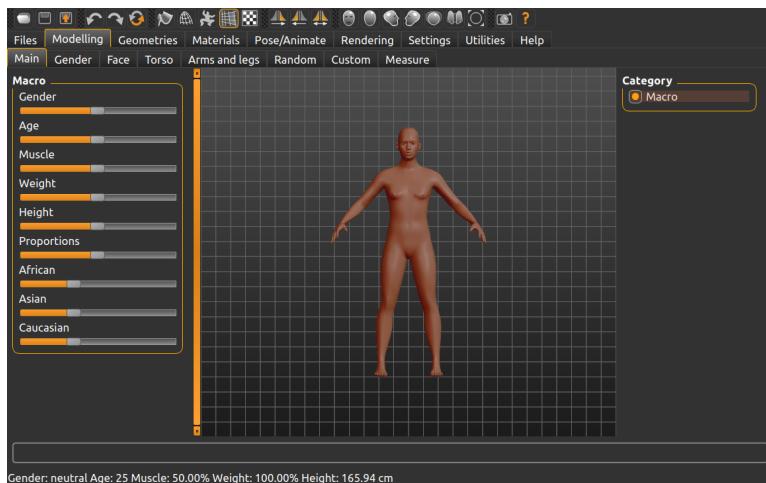


Figura 3.4: Aspecto del humano estándar de MakeHuman

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 simultánea se podría realizar la configuración de los personajes para su posterior animación.

### 3.1.3. Fuse Character Creator

Se trata de una aplicación desarrollada por Mixamo, la cual permite crear personajes 3D 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 *scroll* o deslizamiento hacia la izquierda o derecha para aumentar su tamaño. Por otro lado, para seleccionar el punto exacto donde se ubicarán las diferentes partes del cuerpo como por ejemplo el ombligo, los ojos, las rodillas, etc, se realizará un *scroll* o desplazamiento hacia 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. Pudiendo com-

pletar la configuración de los huesos (*rigger*), con la posibilidad de crear varios esqueletos para un mismo personaje y además incluir las animación deseadas.

Para la utilización de esta aplicación es necesario el uso del servidor Steam<sup>1</sup>, ya que se trata de una plataforma de distribución digital, comunicaciones y servicios multijugador, donde está alojado *Fuse Character Creator*.

COMENTARIO: Terminar con alguna imagen y meter enlaces a los programas

### 3.1.4. Mixamo 3D Animation Online Services

Mixamo<sup>2</sup> es una compañía tecnológica encargada del desarrollo y venta de servicios para la animación de personajes 3D. Proporcionando un servicio *online* que permite buscar entre una amplia gama de movimientos, que van desde el combate, pasando por deportes, hasta poder tocar un instrumento.

Los productos y características de Mixamo son compatibles con todos los paquetes de software 3D, pudiendo realizar una exportación eligiendo entre varias aplicaciones 3D como: Unity3d, UDK, 3ds Max, Maya, Motionbuilder, zBrush, Houdini entre otros.

Una de las características más sorprendentes de Mixamo es el *Auto-Rigger*. Ya que ahorras horas de trabajo modelando el personaje y con este sistema el proceso sucede en cuestión de minutos. Todo lo que se necesita hacer es subir al servidor de Mixamo el personaje 3D (soporta .fbx, .bvh, .zip, .obj), seleccionar el numero de huesos que mejor se adapte al esqueleto del personaje (véase Figura 4.9) y esperar que termine el proceso de *rigger*.

Tras completar la fase de modelado, Mixamo ofrece la posibilidad de animar al personaje con movimientos, que permiten, no sólo aplicar cualquier animación, sino también personalizarlas, ya que se pueden modificar en tiempo real con el personaje obteniendo el resultado deseado. Por último descargarlos en cualquier formato que necesita (.fbx, .bvh, .dae) y fácilmente combinar de nuevo en sus escenas en 3D

Estas animaciones utilizan una base matemática, lo que facilita su uso para los diferentes personajes que se deseen animar. Tradicionalmente era extremadamente lento, costoso y complicado, lograr modelar personajes 3D con resultados de alta calidad. En cambio mediante esta técnica, Mixamo intenta reducir los tiempos de desarrollo, pudiendo obtener unas animaciones de calidad, sin la necesidad de contar con altos recursos.

<sup>1</sup><http://store.steampowered.com/?l=spanish>

<sup>2</sup><https://www.mixamo.com/>

### 3.1.5. 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.

COMENTARIO: terminar

### 3.1.6. Blender

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

COMENTARIO: terminar si se mete el logo

## 3.2. Hardware empleado

Sin embargo, el desarrollo del proyecto se llevó a cabo en ordenadores portátiles personales sin grandes especificaciones técnicas más allá del soporte gráfico para la RV

El hardware principal usado en este proyecto es el dispositivo de captura de movimiento **Kinect versión 2** (Microsoft, 2010) y el adaptador de **Kinect** para conectar el sensor al sistema operativo Windows (Microsoft, 2014a). Sin embargo, el desarrollo del proyecto se llevó a cabo en ordenadores portátiles personales, teniendo que utilizar uno de ellos como enlace con **Kinect**, debido a la limitación de uso que tiene el dispositivo en un solo sistema operativo, como es Windows.

### 3.2.1. Kinect versión 2

Se trata de un periférico que permite a los usuarios interactuar con la aplicación, de modo que gracias a sus sensores, no tiene la necesidad de utilizar algún tipo de mando inalámbrico. Este dispositivo se documenta en la sección 2.4.3 del proyecto.

Pasados unos años tras la salida de **Kinect**, en 2014 Microsoft liberó el controlador *Software Development Kit (SDK)* para entornos Windows de forma gratuita, con el objetivo de sacarle un mayor rendimiento a esta tecnología. De modo que, facilitó su uso en múltiples aplicaciones, como en el caso de la captura de movimientos.

Por lo tanto, para el uso de **Kinect** en un equipo o *tablet* con un sistema operativo Windows (Microsoft, 2014c), deberá cumplir unos requisitos mínimos *hardware y software*:

- Procesador de 64 bits (x64).

- Doble núcleo físico de 3,1 GHz.
- Controlador USB 3.0 (versión 2.2.1610.17001 o superior)
- Adaptador de **Kinect** para conectar el sensor al sistema operativo Windows.
- 4 GB de RAM.
- Tarjeta gráfica que admita DirectX 11.
- Windows 8 u 8.1, Windows Embedded 8 o Windows 10.

Microsoft aporta una herramienta<sup>3</sup> para verificar que el PC o la *tablet* cumplen con los requisitos de compatibilidad y para comprobar el sistema en busca de cualquier problema conocido. Además, asegurándose de estar utilizando el controlador mas reciente para la GPU. (véase Figura 3.5)

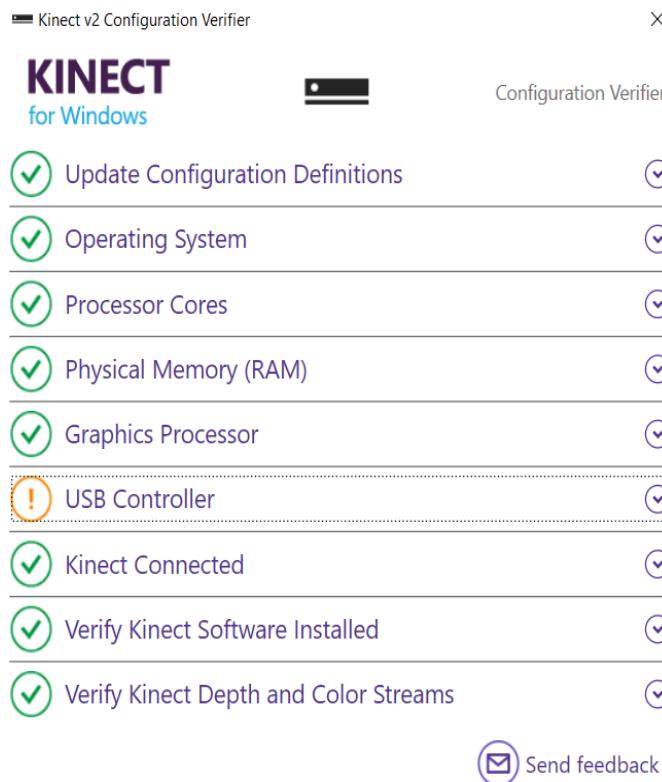


Figura 3.5: Herramienta verificadora de **Kinect**

Para el correcto funcionamiento de **Kinect**, será necesario instalar varios *softwares* añadidos:

<sup>3</sup><http://go.microsoft.com/fwlink/?LinkID=513889>

- SDK (versión 11)<sup>4</sup>
- *Speech Platform Runtime* (versión 11)<sup>5</sup>
- *DirectX Software Development Kit*<sup>6</sup>

Debido a los problemas ocasionados con algunos equipos utilizados, se deberá comprobar previamente que el adaptador *host* USB 3.0 es compatible con **Kinect**, ya que no todas las versiones del controlador proporcionan el ancho de banda del USB requerido por el sensor. Se deberá tener en cuenta, que se puede necesitar probar diferentes ranuras PCI-e para encontrar un controlador que se adapte a las necesidades del dispositivo **Kinect**.

### 3.2.2. Adaptador de Kinect para Windows

Conecta el sensor **Kinect** para Xbox One al ordenador o tableta mediante el adaptador **Kinect** para Windows (véase Figura 3.6) y de ese modo, poder desarrollar soluciones personalizadas para diferentes aplicaciones.

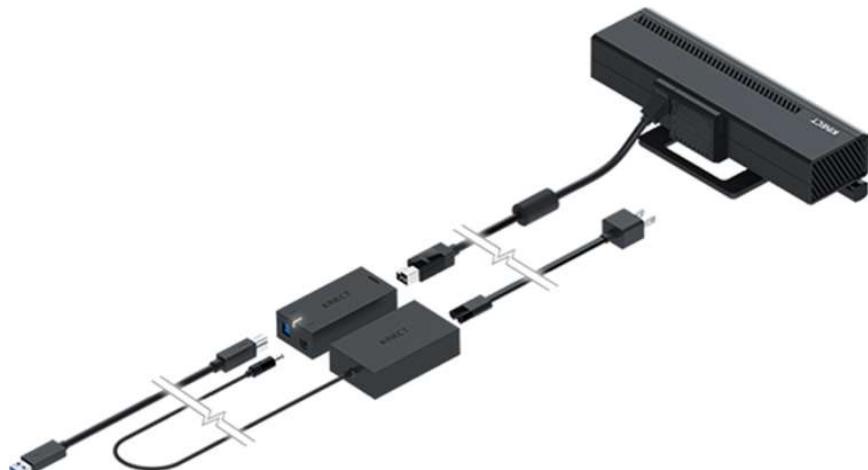


Figura 3.6: Adaptador de **Kinect** para Windows

### 3.2.3. Equipo para el uso de Kinect

Este equipo no posee una características muy especiales, mas allá de utilizar el sistema operativo Windows. De modo que será el PC utilizado para conectar **Kinect**, obteniendo un correcto funcionamiento.

Teniendo las siguientes especificaciones técnicas:

<sup>4</sup><https://www.microsoft.com/en-us/download/details.aspx?id=44561>

<sup>5</sup><https://www.microsoft.com/en-us/download/details.aspx?id=27225>

<sup>6</sup><https://www.microsoft.com/en-us/download/details.aspx?id=6812>

- Procesador: Intel® Core i5-4210U (1.7 GHz)
- Memoria RAM: 8GB DDR3L SODIMM PC3-12800 1600Mhz
- Disco duro: 500GB (5400 rpm S-ATA)
- Display: 15.6"LED Full HD (1920x1080)
- Controlador gráfico: Nvidia GeForce 820M 2GB DDR3
- Conexiones: 1 USB 3.0 (versión 2.2.1610.17001) , 2 x USB 2.0, 1 HDMI

# Capítulo 4

## 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 este capítulo se explicará que librerías y paquete de **Unity** se necesitan para tener una buena conexión con **Kinect** y, además, que información obtenida del sensor de movimiento podemos aprovechar para el proyecto. Después de esta aclaración, se expondrá como grabar y reproducir los movimientos del usuario captado, analizando los datos ofrecidos por el sensor de **Kinect**.

En la siguiente sección se especificará el proceso realizado para la comparación de los movimientos capturados, de modo que, el usuario obtendrá una respuesta concisa. Para ofrecer una aplicación dinámica y que el usuario reciba un *feedback* gestual, se ilustra como se diseña e implementan avatares con animaciones en **Unity**.

De forma sucesiva, se describe el diseño de los escenarios 3D para ubicar a los avatares principales y secundarios de la aplicación. De este modo se han diseñado dos escenarios, uno para el entrenamiento del usuario y otro para reproducir el movimiento a realizar.

En el apartado posterior, se describe una manera de acabar la comparación del movimiento de forma más completa. Ofreciendo así, un análisis de cada parte del cuerpo describiendo el modo de corregir el movimiento.

Posteriormente se elaboran una serie de escenas en **Unity** que servirán de interfaz para la aplicación. En las diferentes escenas se encuentran la de inicio, el menú principal, ayuda, créditos, etc. En cada una de ellas se ha creado un entorno diferenciado para cada una de las necesidades dadas.

Y por último, una vez acabada la funcionalidad de la aplicación, se realizan la grabaciones de los movimientos con los profesionales de la escuela **Abadá-Capoeira**. Elaborando así, una batería de pruebas para seleccionar y ajustar el movimiento más preciso.

## 4.1. Paquetes para Unity

El primer paquete que se prueba es el que nos ofrece Microsoft <sup>1</sup>, que es un paquete destinado principalmente para UnityPro.<sup>2</sup> 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 está 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 optó por la opción de descartar este paquete para evitar futuras frustraciones.

Después de descartar el paquete que nos ofrecía Microsoft decidimos buscar en el Assets Store de Unity(Explicación o referencia). Encontramos el paquete **Kinect v2 Examples with MS-SDK**<sup>3</sup> 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.

---

<sup>1</sup><https://developer.microsoft.com/es-es/windows/kinect/tools>

<sup>2</sup>Unity profesional de pago con más funcionalidad que Unity personal.

<sup>3</sup><https://www.assetstore.unity3d.com/en/#!/content/18708>

- *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 desplazan.
- *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 simulando 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 explica con mas detalle que se utiliza de

estos ejemplos.

## 4.2. 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.

### 4.2.1. Investigación base

En una primera vista se muestra como Kinect capta al usuario, esta representación se hace mediante un *cubeman*<sup>4</sup> 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 4.1) 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.

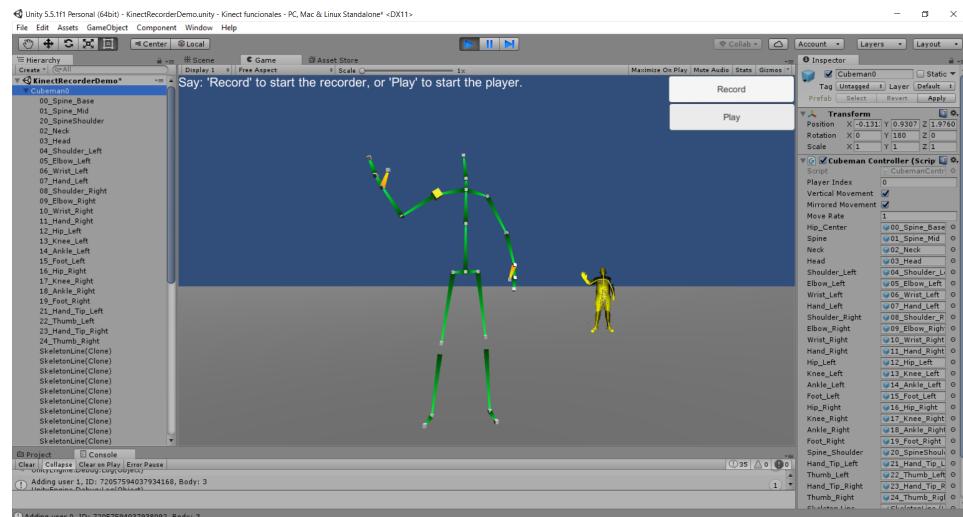


Figura 4.1: Cubeman con la lista de todos los Joints

<sup>4</sup>Esqueleto verde que emula el movimiento de la persona captada.

Para hacer una representación correcta del esqueleto y que 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*<sup>5</sup> encargado de la comunicación entre el sensor de *Kinect* y las aplicaciones de *Unity*.

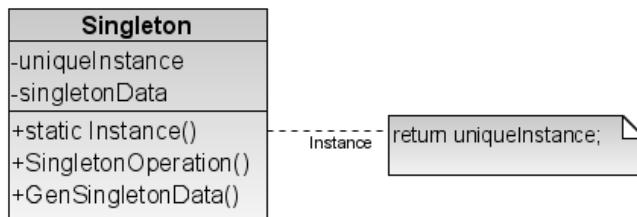


Figura 4.2: Patrón de Singleton

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*<sup>6</sup> donde aparece la información de la imagen, profundidad, color y datos de los usuarios captado con el sensor. Los datos de los usuarios captados se encapsulan en el *BodyDataFrame*. Esta estructura, a vez, contiene la información del número de usuarios captados, su identificador, la información de los *BodyData*<sup>7</sup>, 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

<sup>5</sup>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

<sup>6</sup>*script* encargado de tener comunicación directa con el sensor de la *Kinect*

<sup>7</sup>Información de los joint de un cuerpo específico

referencia a un *frame*<sup>8</sup> 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 4.3). *KinecRecorderPlayer* también discriñe entre grabación y reproducción activando solo una de la dos a la vez.

```

1 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^
2 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
3 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
4 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
5 0.155|kb,956161116582760000,6,25,0,1,72057594037972009,2,0.036,-0.115,2.547,2,0.043,0.132,2.485,2,0.048,0.374,2.414
6 0.199|kb,956161116582770000,6,25,0,1,72057594037972009,2,0.036,-0.115,2.547,2,0.042,0.133,2.484,2,0.047,0.375,2.413
7 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
8 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
9 0.289|kb,9561612658300000,6,25,0,1,72057594037972009,2,0.034,-0.112,2.548,2,0.038,0.135,2.486,2,0.041,0.377,2.414
10 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
11 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
12 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
13 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
14 0.465|kb,95616143282610000,6,25,0,1,72057594037972009,2,0.032,-0.111,2.550,2,0.035,0.136,2.485,2,0.035,0.376,2.405
15 0.489|kb,95616146582650000,6,25,0,1,72057594037972009,2,0.032,-0.111,2.550,2,0.031,0.137,2.479,2,0.025,0.381,2.395
16 0.532|kb,95616149990760000,6,25,0,1,72057594037972009,2,0.031,-0.111,2.549,2,0.029,0.136,2.476,2,0.024,0.378,2.396
17 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 4.3: Fichero de texto con los datos del BodyData

#### 4.2.2. Implementacion de la Grabación y reproducción de los movimientos

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 4.4).

<sup>8</sup>Es un fotograma, *Unity* por defecto reproduce a 60 *frames* por segundo

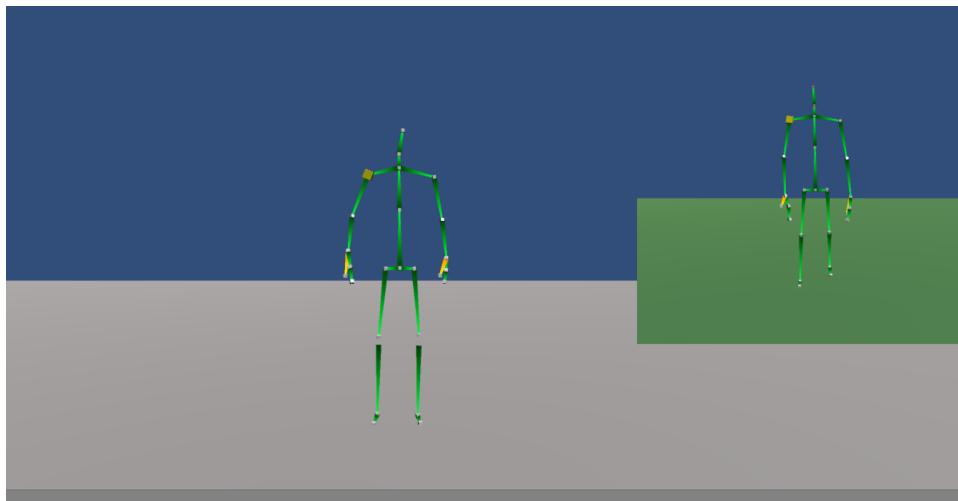


Figura 4.4: Escena con los dos cubeman

### 4.3. 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.

#### 4.3.1. 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 específica y laboriosa para cada movimiento, por lo que se intentó 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 la 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 específica. Para completar el gesto se tendrá un tiempo determinado para llegar a la progresión final del movimiento, sino el gesto se cancelará.

### 4.3.2. Implementación de la comparación de los movimientos

Una vez analizado la funcionalidad de los gestos de **Kinect**, se apoyó en esa idea para implementar 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 siguientes 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 transición de estados, se incrementa en más de una unidad el estado, ya que la comparación de dos *frames* consecutivos es muy similar, ahorrando 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, Y, Z) con un margen de error modifiable 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 4.5). El resultado del gesto se muestra por pantalla para dar un *feedback* inmediato al usuario.

## 4.4. Creación de avatares y animaciones

En esta etapa del proyecto, se han desarrollado los diferentes avatares con una estética humana. Con el objetivo de dar una sensación mas realista a los *cubeman* y de este modo, proporcionar una perspectiva mas amena al usuario.

### 4.4.1. 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 4.6 ). El primer avatar utilizado fue simple, ya que al principio interesaba ver el resultado que proporcionaba el personaje junto

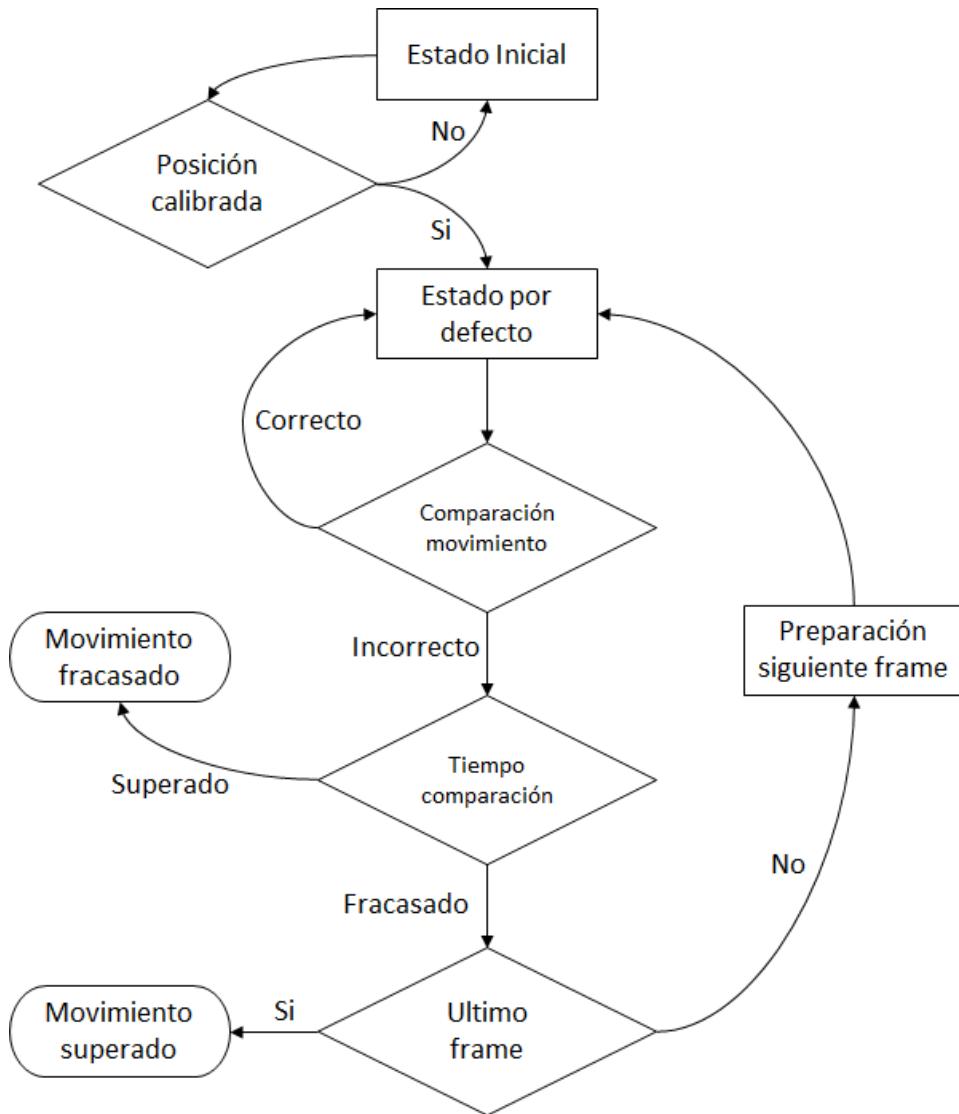


Figura 4.5: Grafo de comparación de movimiento

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 4.7 ) 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 4.7 ). 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

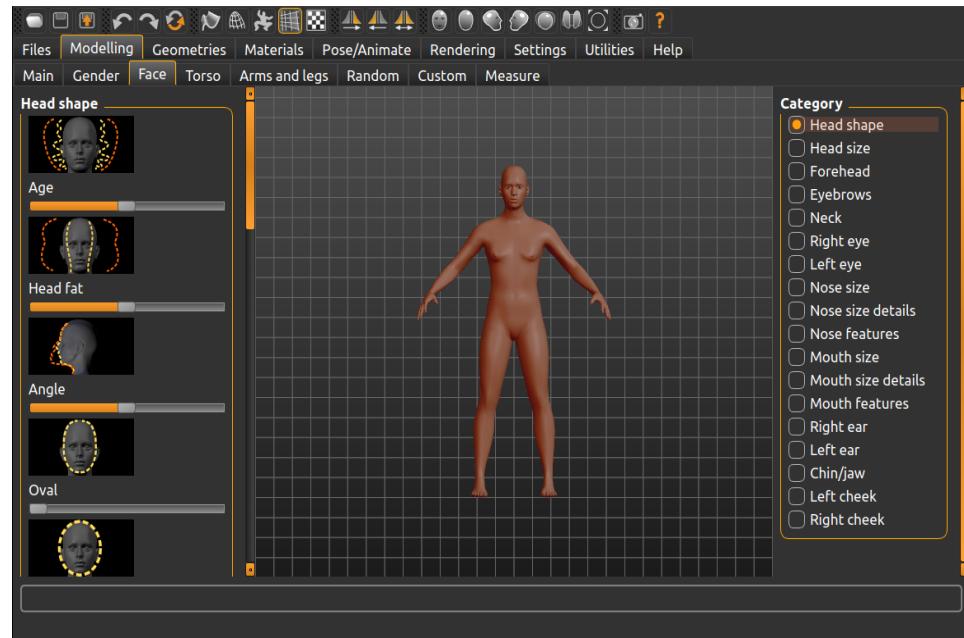


Figura 4.6: Avatar inicial

Figura 4.8 ). 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 personajes 3D como **Blender**, pero con una curva de aprendizaje demasiado larga para las necesidades dadas. También se examinó una aplicación llamada **MarvelousDesigner** la cual se emplea para desarrollar prendas de vestir. El problema surgió 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 los personajes, ya que se amoldaba perfectamente a las necesidades de este proyecto.

#### 4.4.2. Implementación de los avatares y animaciones

COMENTARIO: Cambiar a cuatro avatares si se crea otro

A fin de realizar un entorno más realista, se crearon tres avatares diferentes, cada uno con una estética propia. A la hora de elaborar el vestuario de

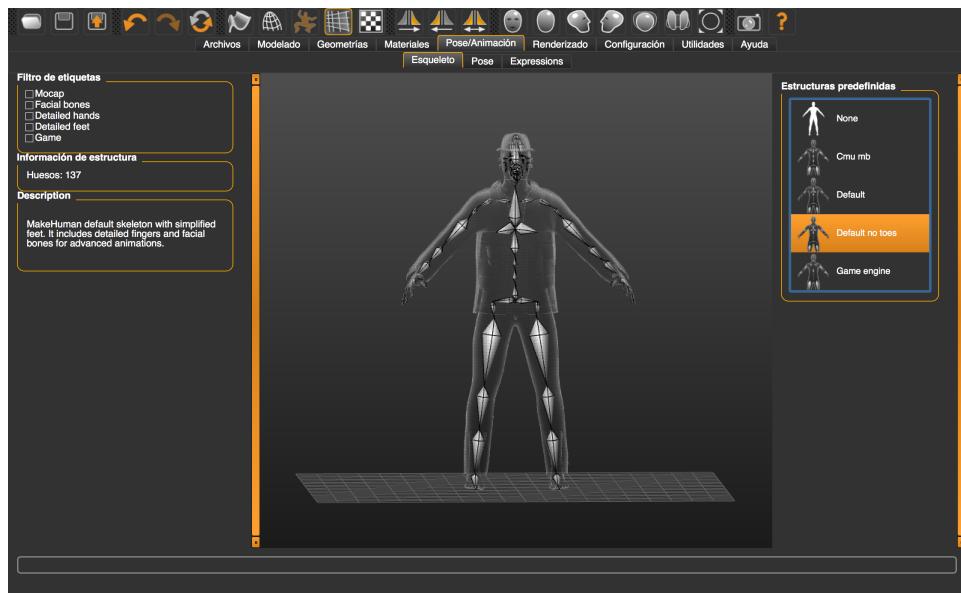


Figura 4.7: Avatar con 137 huesos

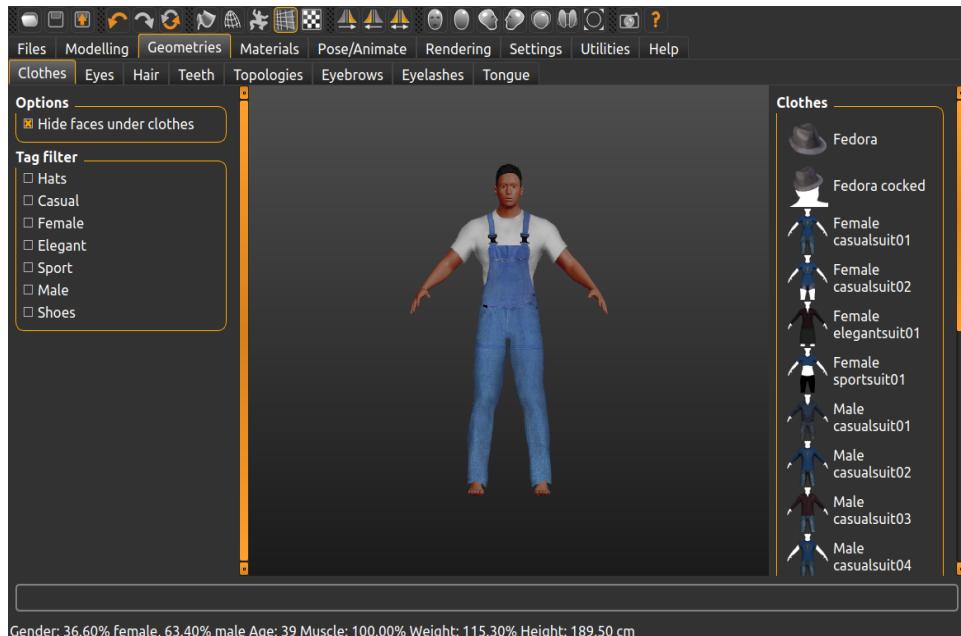


Figura 4.8: Avatar inicial con ropa

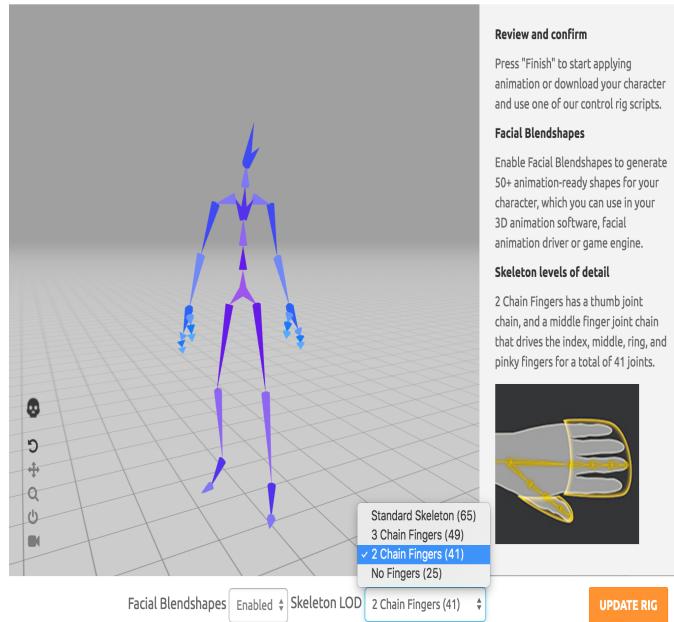


Figura 4.9: Auto-Rig de Mixamo

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 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, 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, obteniendo así, un *feedback* más ilustrado de la comparación del 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 la animación que 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 en todo caso, de la comparación del movimiento por parte del usuario con el del profesor. Una vez determinado el estado, este volverá a la animación de *ginga* para continuar con el siguiente movimiento (ver Figura 4.10).

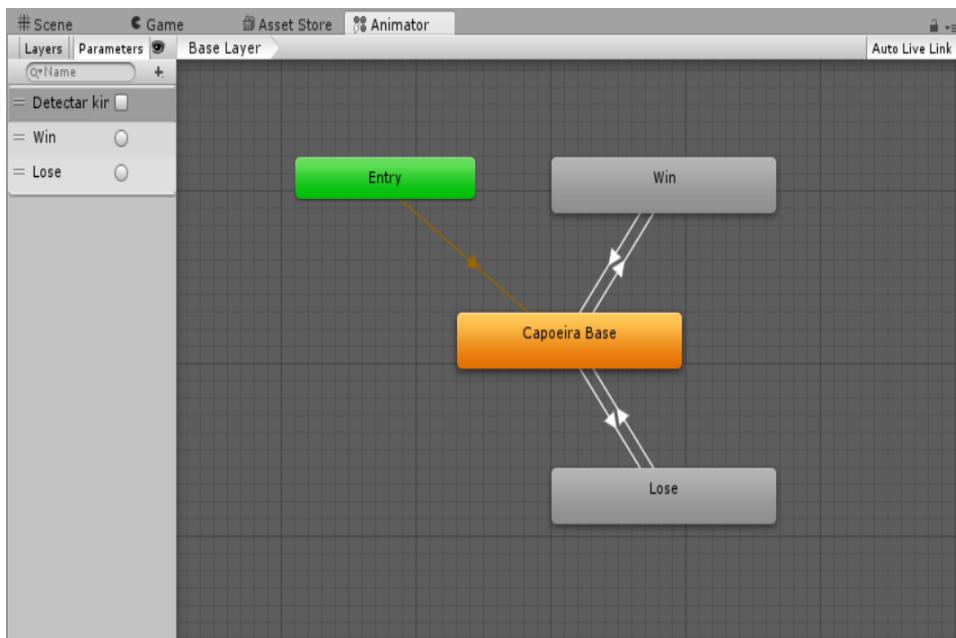


Figura 4.10: Diagrama de estados de animator

Se constató que existía un conflicto entre el **Animator Controller** y el **AvatarController** dando lugar a un lucha 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.

## 4.5. Análisis de los movimientos del Usuario

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

### 4.5.1. Investigación base

Para este apartado se observó el desarrollo de la comparación de movimiento implementada anteriormente. Esta 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.

### 4.5.2. Implementación del análisis de los movimientos

Una vez planteada las especificaciones del análisis se empieza con su implementación. Primero 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<sup>9</sup> donde se produce la equivocación y los datos de los **Joints** pertenecientes a los avatares. Posteriormente se necesitó hacer una modificación funcional en el gesto *Move* de **KinectGesture**, para conseguir así, llenar esa información en cada comparación. 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 la traza del movimiento hasta que el usuario se equivoque.

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

---

<sup>9</sup>Corresponde al frame que se está comparando.

materiales para que se vean transparentes ,y crear así, un esqueleto verde parecido a los *cubeman* del principio. Estos avatares simulan el movimiento hasta el *frame* donde se equivocó el usuario, y en este último estado, realiza 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 cuatro partes. Para la primera pieza se creó un diccionario que, basándose en el número del *Joint*, devuelve en palabras la parte del cuerpo correspondiente. En la implementación de este diccionario se han agrupado *Joints* dispares, ya que siendo diferentes forman una misma parte del cuerpo, como por ejemplo las manos que la forman tres *Joints*. La segunda pieza del *string* está formada por la elección de la dirección, y esta se comprueba, realizando la diferencia de la coordenadas X del *Joint* perteneciente al 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. Seguidamente para la tercera pieza del *string*, se realiza la diferencia con la coordenada Z definiendo así la profundidad. Si la diferencia es positiva la profundidad será hacia atrás , en caso contrario hacia delante. Y por último para la cuarta pieza del *string*, se efectúa la distinción con la coordenada Y, definiendo así la altura. Si la diferencia es positiva la altura será definida de forma descendente, en caso contrario ascendente. Esta lista de correcciones se mostrará en un panel una vez completada evitando los *strings* repetidos, a su vez, se añaden colores a cada una de las piezas del *string*, ofreciendo de esta manera un texto menos plano y mas legible (ver Figura 4.11)

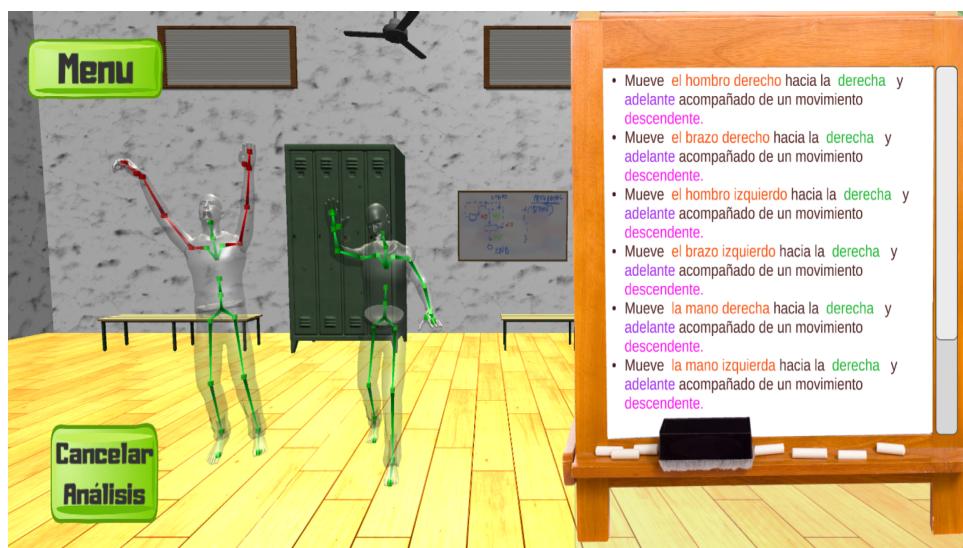


Figura 4.11: Análisis de movimiento

## 4.6. Creación de los escenarios 3D

Con el objetivo de conseguir un ambiente más realista y dinámico, se elaboraron dos escenarios diferentes, ambientados cada uno de ellos según su propósito. El primero de ellos escenifica un entorno para el alumno donde realizará el entrenamiento, como es el caso de un gimnasio. El segundo, está orientado hacia el profesor y los animadores, representando una playa Brasileña.

COMENTARIO: Meter foto escenario alumno y profesor

### 4.6.1. Escenario del gimnasio

La creación del escenario del 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**<sup>10</sup>, para simular un suelo de madera. En la construcción de las paredes del escenario se crearon tres paneles, colocándolos de forma vertical formando un cubo abierto, y además se añadió a todos los paneles, un material del paquete **18 High Resolution Wall Textures**<sup>11</sup> 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**<sup>12</sup>. 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**<sup>13</sup>, 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.

### 4.6.2. Escenario de la isla

Para la creación de este escenario, y con la misión de realizar un entorno ambientado en las playas brasileñas, se realizó una investigación previa sobre los paquetes existentes que cumplían estas características. Se descartaron varias opciones de pago, ya que algunos no cumplían con las particularidades buscadas y otros tenían un coste demasiado elevado. Por lo tanto, tras esta decisión, se optó por usar el paquete gratuito **Island Assets**<sup>14</sup> obtenido en el **Asset Store** de **Unity**, de modo que junto al profesor y los animadores,

<sup>10</sup><https://www.assetstore.unity3d.com/en/#!/content/13213>

<sup>11</sup><https://www.assetstore.unity3d.com/en/#!/content/12567>

<sup>12</sup><https://www.assetstore.unity3d.com/en/#!/content/3355>

<sup>13</sup><https://www.assetstore.unity3d.com/en/#!/content/5977>

<sup>14</sup><https://www.assetstore.unity3d.com/en/#!/content/56989>



Figura 4.12: Escenario de gimnasio

este escenario simulase la sensación de estar en una playa brasileña.

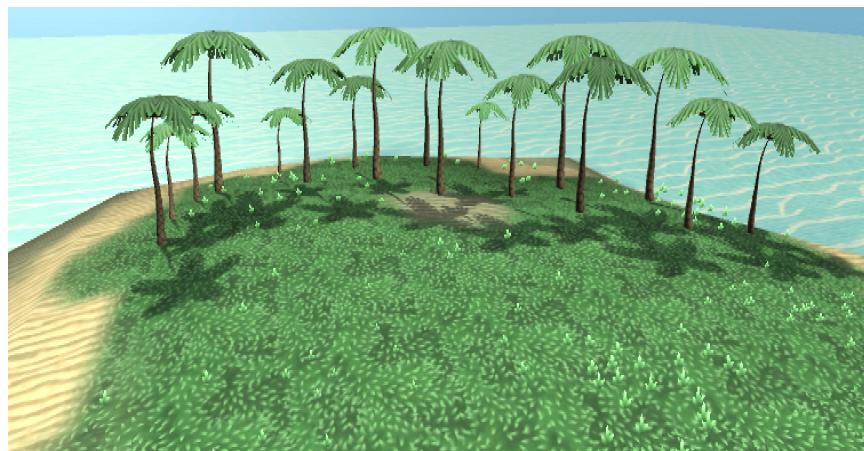


Figura 4.13: Escenario para el profesor y los animadores

Posteriormente, se hicieron una serie de modificaciones para que el paquete **Island Assets** tuviera un aspecto acorde con el proyecto a desarrollar. Lo primero que se hizo fue quitar el componente *Treasure chest*(cofre) ya que desentonaba demasiado con la temática del entorno deseado. En segundo lugar se desactivó el componente *Dock*(Embarcadero) puesto que no sería necesaria su utilización debido a la falta de barcos navegando. Por último, el

paquete de **Unity** venía predefinido con tres palmeras, dando una sensación de isla desierta. De modo que, se añadieron como componente del terreno, más de una decena de palmeras en tamaños aleatorios, intentando dar un efecto de isla habitada (véase Figura 4.13). Con todo ello, el paquete **Island Assets** de **Unity** ha servido como escenario para simular una playa brasileña.

## 4.7. Grabacion de movimientos con gente experta

En esta etapa del proyecto se visitó la Asociación Cultural Deportiva Rio<sup>15</sup>, donde se reúnen los practicantes expertos en capoeira, para realizar una serie de grabaciones y pruebas.

### 4.7.1. 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 se ejecuta con una patada frontal básica, realizada contra el abdomen o el pecho del oponente.
- *Chapa-Pisão*, movimiento similar al Bêncão 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.

<sup>15</sup><https://www.asociacionrio.com/>

- *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 180° 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.
- *Skada*, movimiento que consisten en lanzar golpes consecutivos hacia delante con la mano abierta.
- *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.

#### 4.7.2. Elección y ajuste de los 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-Pisão 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.



Figura 4.14: Realizando las grabaciones de movimiento

#### 4.8. Transición de las 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 4.15

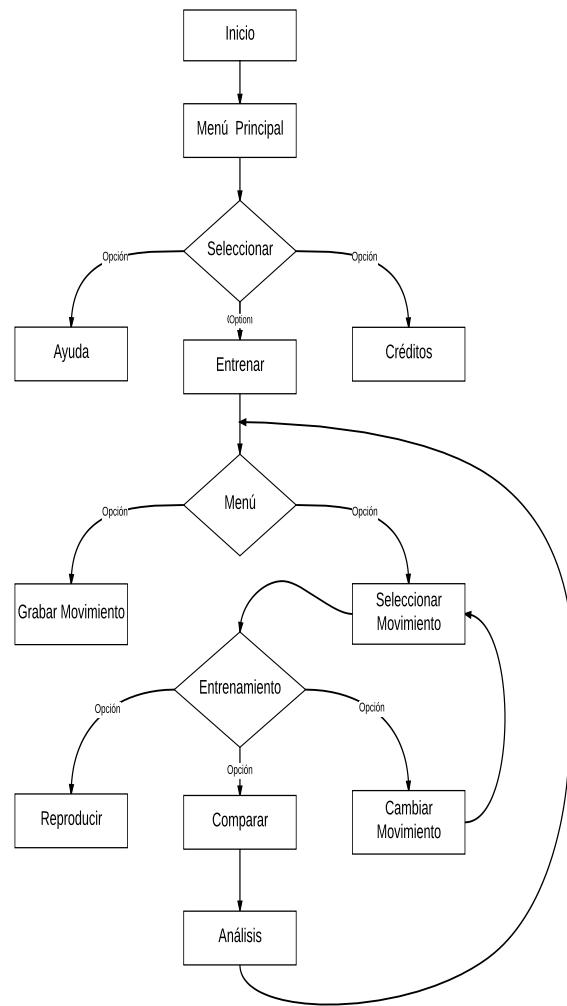


Figura 4.15: Diagrama que ilustra la lógica del sistema

#### **4.8.1. Escena de 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.

#### **4.8.2. Escena de 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.

#### **4.8.3. Escena de 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.

#### **4.8.4. Escena de 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.

#### **4.8.5. Escena de 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 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.<sup>16</sup>

#### 4.8.6. Escena de 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.

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*).

---

<sup>16</sup><http://www.abadacapoeira.com.br/>

## 4.9. Conclusiones del desarrollo del proyecto

A lo largo de este capítulo se ha logrado explicar los estudios y las implementaciones necesarias para este proyecto. La primera decisión fue elegir el paquete para **Unity Kinect v2 Examples with MS-SDK**, siendo este, un pilar importante para la compresión y entendimiento del funcionamiento de **Kinect**.

Los datos obtenidos del sensor de **kinect**, son extraídos para interpretarlos y mostrarlos a través de los *cubeman*. Gracias a ello, se consiguió guardar y catalogar los movimientos del *cubeman* para realizar comparaciones. El funcionamiento de los gestos que reconoce **Kinect** fueron una inspiración para la implementación de la comparación de movimientos.

Posteriormente, se diseñaron avatares con el programa **Fuse Character Creator** para sustituir a los *cubeman* y dar una apariencia más realista a la aplicación. Con el objetivo de dar un toque más dinámico, se añadieron a los avatares unas animaciones de movimientos de capoeira, de modo que, se reproduzcan cuando no se detecte al usuario. Adicionalmente, para el *feedback* de la comparación del movimiento, se añadieron también dos animaciones de victoria y derrota.

Para separar a los avatares se lograron diseñar dos escenarios. El primero simula un gimnasio dando la sensación al usuario de que está practicando, y el segundo escenario simula una isla tropical típica donde los expertos de capoeira practican. En el segundo escenario se mostrarán los movimiento que el usuario debe realizar.

Con el objetivo de completar la comparación de los movimientos, se logró implementar un proceso que realiza un análisis de cada parte del cuerpo. De esta manera, se contrasta con el movimiento a realizar y mostrará las correcciones oportunas que el usuario debe ejecutar.

Y por último, se consiguieron grabar movimientos de gente experta en capoeira, con el fin de catalogarlos y prepararlos para su ejecución.

## Capítulo 5

# Discusión, Conclusiones y Trabajo futuro

### 5.1. Discusión

Como se mencionó en la introducción, uno de los objetivos al desarrollar este sistema era que pudiese funcionar con tecnología de uso doméstico, lo cual restringe notablemente el abanico de posibilidades a considerar. La configuración elegida para el sistema utiliza una única cámara para la captación del movimiento del usuario, proporcionada por **Kinect**. Esto limita la capacidad del sistema de percibir los movimientos más complejos, como aquellos casos en que hay rotación del torso, lo cual puede generar confusiones entre distintos miembros del avatar que tengan simetría axial (tras el giro, el sistema confunde un brazo con otro, y piensa que el usuario sigue de frente cuando en realidad está de espaldas). Lo mismo sucede a la hora de analizar la profundidad de algunos de los movimientos realizados. Se espera en el futuro poder ampliar este trabajo con la utilización de mayor número de cámaras.

Por otro lado, en algunos de los trabajos estudiados (e.g. Keerthy (2012); Kyan et al. (2015)) se utilizan técnicas más sofisticadas para analizar los movimientos de los usuarios que las usadas en este trabajo. Si bien tales técnicas pueden ser útiles en niveles de aprendizaje avanzados, cuando se trata de principiantes en capoeira, no existen grandes diferencias respecto a técnicas más sencillas. Esto es así porque los movimientos de un aprendiz son más simples y, por tanto, la comparación con el entrenador también lo es. Por ello, un método más sencillo de comparación de movimientos se traduce en dos claras ventajas: primero, es menos complejo y más rápido de calcular; y, segundo, el resultado de la comparación es más fácilmente traducible a texto para proporcionar al aprendiz explicaciones útiles acerca de cómo mejorar la realización de las técnicas.

## 5.2. Conclusiones

A lo largo del presente trabajo se ha descrito el proceso por el cual se ha desarrollado un entrenador virtual de capoeira a partir de la grabación previa de los movimientos de un experto realizada con el dispositivo **Kinect**. Por otro lado, el usuario, haciendo uso del mismo dispositivo, hará una captura de sus movimientos que servirán para su posterior comparación con el entrenador virtual. Una vez comparados, se ofrece la explicación oportuna para conseguir realizar correctamente los movimientos que fueron ejecutados de manera errónea. En este proceso se debe enfatizar la capacidad del dispositivo **Kinect** para capturar los movimientos de forma correcta, un hecho relevante si además tenemos en cuenta su bajo precio de mercado.

Hay que destacar los beneficios que puede conllevar utilizar un entrenador virtual, que engloba, desde la comodidad de trabajar desde casa, hasta tener un *feedback* dedicado, personal e inmediato. Del mismo modo, se trata de un sistema de entrenamiento mediante juego, lo cual potencia el aprendizaje y la motivación, como se aprecia en la escena de alumno, en la que se debe de realizar el movimiento correctamente. Permite establecer la dificultad deseada para comparar los movimientos, lo cual hace que se pueda adaptar a las necesidades y capacidades de los distintos alumnos. Se trata de un entorno intuitivo tanto para el profesor como para el alumno, pudiéndose complementar con la sección de ayuda que ofrece la aplicación. Además, su uso doméstico a bajo coste posibilita un mayor alcance entre el público.

Aunque aún no se ha realizado una evaluación extensa con usuarios novatos y expertos en capoeira, sí se han llevado a cabo pruebas preliminares para comprobar el correcto funcionamiento de la aplicación y la fiabilidad de los comentarios para corregir sus defectos. Dichas pruebas se han realizado con una pequeña muestra 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.

## 5.3. Trabajo futuro

Este proyecto tiene potencial en el campo de entrenamiento de actividades físicas, ya que comparando el movimiento del usuario con el de un experto se ofrece un feedback visual. De este modo, el usuario podrá corregir su técnica en caso de que cometa algún tipo de error.

Una funcionalidad que se puede implementar en un futuro sería la gestión de más usuarios y el registro de la progresión. Cuando el usuario acceda con su cuenta, se mostrará el porcentaje con el avance conseguido de cada movimiento.

Otra futura ampliación trata de añadir más funcionalidad al experto que graba los movimientos. Con ello, tendrá la posibilidad de manejar y modificar los movimientos grabados de una forma sencilla.

Por otro lado, se plantea la necesidad de incluir métodos más sofisticados para el análisis de los movimientos, similares a los usados en Keerthy (2012); Kyan et al. (2015) para permitir un análisis más fino de los movimientos que proporcione 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 diferentes tipos de rehabilitación. En este caso serviría para comunicarle a un paciente que tenga, por ejemplo, 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. Una 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 esta manera, se podrían 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.



## Capítulo 6

# Distribución de trabajo

En el presente TFG la distribución de trabajo por parte de los miembros del equipo de desarrollo ha sido bastante ecuánime.

En una primera instancia fue necesario realizar una investigación conjunta sobre el campo de trabajo. El cual comprende el uso de la informática y nuevas tecnologías aplicadas a la captura de movimiento, que al ser un tema totalmente novedoso para los desarrolladores, implicó un amplio estudio del estado del arte actual.

Debido a las características del trabajo y los dispositivos necesarios para su desarrollo, al contar con recursos limitados, es decir, con un único dispositivo **Kinect** y un equipo compatible con las características del mismo, resultaba complicado el desarrollo en paralelo. Por ello, se adoptó la política de establecer reuniones diarias, en las que se trabajaba de manera intensiva en tareas conjuntas que permitieran la integración en el sistema de los dispositivos mencionados. Estas reuniones tuvieron lugar en la Facultad de Informática de la Universidad Complutense de Madrid, específicamente en la biblioteca de este centro.

El objetivo general del proyecto era el desarrollo de un entrenador personal virtual basado en el arte marcial afro-brasileño capoeira. Para ello, la intención consistía en mostrar, mediante un escenario 3D, el entrenamiento realizado por un usuario para el aprendizaje de capoeira, siendo necesario exponer los movimientos del entrenador virtual con el fin de ser imitando. Inicialmente, se desconocía como se iba a llevar a cabo este objetivo global y abstracto, ya que requería la comprensión profunda del dispositivo **Kinect** y sus posibilidades. Uno de los aspectos más importantes del proyecto ha sido todo lo relacionado con la captura de movimientos. Debido a los problemas de conectividad de **Kinect** con los ordenadores y que solo funcionaba en un equipo, se decidió de manera conjunta, estudiar y extraer los datos transmitidos por el sensor de **kinect**.

Tras un breve periodo inicial de aprendizaje en Unity, ambos desarrolladores contaban con los conocimientos necesarios para abordar el escenario

3D. Una vez entendido el funcionamiento de la simulación del *cubeman*, de forma conjunta, se investigó la manera de guardar la información de la captura de movimiento, empleando así, un fichero de texto como registro de cada articulación del *cubeman*. Los datos se archivan guardando la identificación del *cubeman* y cada una de las coordenadas(X,Y,Z) asociadas a las articulaciones. Gracias a la investigación anterior, se implementó la forma de reproducir los movimiento pregrabados, realizados por los desarrolladores, a la vez que se simulaba el movimiento en tiempo real de un usuario. Esto consiste en generar los datos del *BodyData* registrados en el *SensorData* de diferentes fuentes. La primera de ellas, trata de datos obtenidos a partir de la cámara de *Kinect* y la segunda, genera el *BodyData* con el fichero de texto. Aunque antes de añadir la segunda fuente, se busca un espacio libre en el *SensorData* de *Kinect* para no pisar el *BodyData* del usuario que está captando la cámara, y de esta manera, mostrar los dos *cubeman* a la vez.

Llegados a este punto, se debatió entre varias ideas para desempeñar una de las funcionalidades del entrenador virtual, la comparación del movimiento. El planteamiento exitoso, consistió en crear un gesto para la comparación del movimiento, y este a su vez, desempeña un comportamiento de máquina de estados, sirviendo el primer estado para calibrar la posición inicial del movimiento y los siguientes estados se encargan de la comparación del movimiento a superar. Esta contribución se realizó por los dos miembros del grupo de forma equitativa.

A partir de esta etapa del desarrollo se pudo realizar cierto reparto de tareas, pero siempre atado a las limitaciones de tener un único dispositivo *Kinect*.

El integrante del grupo **Víctor**, contribuyó a conectar los huesos de los avatares con la información de los *Joints* ofrecidos por el *Sensordata*, de esta manera, los avatares son animados por los movimientos de los usuarios. Después acopló tres animaciones a los avatares, siendo la primera el movimiento de la *ginga* que se activa cuando no se detecta al usuario, ofreciendo así, un aspecto mas dinámico al avatar. Las dos animaciones siguientes son la pose de victoria y derrota, que son incorporadas para dar un *feedback* visual al usuario en respuesta del movimiento realizado. Después, se observó un conflicto generado por el componente de **animator** y el componente que conectaba el avatar con *Kinect*, haciendo imposible el control del avatar cuando los dos componentes estaban activos. Por ese motivo, desarrolló un método que desactiva por completo el componente **animator** cuando no se detectaba un usuario, haciendo solo que el controlador de *Kinect* asumiera pleno control sobre el avatar.

Siguiendo con la contribución de Víctor, implementó la forma de mostrar el análisis del movimiento comparado anteriormente. Para lograr esta funcionalidad, registró el movimiento del usuario y el del experto hasta el momento que comete el error, de esta manera, se tiene la información de la

---

posición de cada uno de los **Joints** perteneciente a los dos avatares. A continuación, modificó el aspecto de los avatares para hacerlos transparentes y generar un esqueleto verde, simulando de este modo las partes del cuerpo. A partir de este estado, los avatares reproducen el movimiento registrado hasta el momento en que el usuario falla, coloreando de rojo los huesos o partes del cuerpo donde el usuario a errado. Adicionalmente se incorporó un panel, indicando las medidas que debe tomar para la corrección de las partes del cuerpo donde ha cometido algún fallo.

Paralelamente, Raúl, el otro integrante del grupo, investigó como darle vida a los personajes del proyecto, con el objetivo de desarrollar diferentes avatares con una estética humana. La primera aplicación que utilizó para ese propósito fue **MakeHuman**, ya que mostraba una interfaz amigable e intuitiva. Creó varios avatares, pero el inicial fue muy sencillo (véase Figura 4.6), ya que de momento le interesaba ver el resultado que proporcionaba esta aplicación junto con el *cubeman*. De los diferentes avatares creados, uno de ellos fue el que se utilizó en un principio (véase Figura 4.7). Al observar que este avatar no plasmaba el realismo esperado, decidió buscar otro software que pudiera realizar esta tarea. En la búsqueda, encontró varias aplicaciones como **Blender** y **MarvelousDesigner**, las cuales al final no fueron utilizadas. Una de ellas, debido a su alta curva de aprendizaje para las necesidades dadas y la otra, no conseguía ser compatible de forma correcta con los formatos de las otras aplicaciones utilizadas. Por esta razón, siguió buscando un *software* que pudiera modelar los personajes de una forma correcta y en el camino halló **Mixamo**. Una compañía que ofrece una aplicación llamada **Fuse Character Creator** de Mixamo que sirve para el desarrollo de personajes 3D, y a su vez, un entorno dedicado a la animación de personajes. De modo que decidió cambiar a esta nueva aplicación, ya que se amoldaba perfectamente a las necesidades de este proyecto.

Como base para realizar los personajes, los dos integrantes del grupo, asistieron a una de las clase semanales que imparten en las instalaciones de la Universidad Complutense de Madrid. De este modo, observaron las características de los integrantes de la escuela **Abadá-Capoeira** y la vestimenta que portaban para realizar los entrenamientos.

Con toda esta información, el integrante Raúl, creó los avatares del proyecto, como son el alumno, profesor y los dos animadores que se encuentran en el fondo del escenario de la isla. Además diseñó la ropa típica de capoeira para los personajes, como son el pantalón largo y la camiseta de tirantes de color blanco. Cuando terminó el modelado de los avatares, inició el proceso para incorporar animaciones a los personajes del fondo de la escena de la isla, como son los animadores.

Antes de unir el trabajo realizado con el integrante Víctor, Raúl diseño la interfaz de tipo UI, pudiendo realizar diferentes transiciones en la aplicación VIC, empezando en la escena de inicio, hasta llegar al entrenamiento y su

posterior análisis.

Aunque se repartieron las tareas era necesario seguir la política de reuniones constantes para compactar las partes y comprobar su correcto funcionamiento.

Posteriormente, después de conectar las tareas anteriores, cada desarrollador diseña un escenario para colocar a los diferentes avatares que aparecen en el proyecto, y de esta manera, poder diferenciar la característica que les identifica.

Por último, con la actuación de ambos componentes del grupo, se hicieron grabaciones con la gente experta de la escuela **Abadá-Capoeira**. Las grabaciones tuvieron una duración de dos días, y gracias a esto, se obtuvieron todas las muestras posibles que la tecnología de captura de movimiento de **Kinect** permitió grabar de una manera aceptable. Después se ajustaron y catalogaron los movimientos para poder incorporarlos al proyecto.

# Parte I

## Apéndices



## Apéndice A

### Así se hizo...

...

...

**R**ESUMEN: ...

#### 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

ALEMÁN SOLER, N. M. *Solución de bajo coste de captura de movimiento basada en Kinect*. Tfg, 2014.

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

CHUA, P. T., CRIVELLA, R., DALY, B., HU, N., SCHAAF, R., VENTURA, D., CAMILL, T., HODGINS, J. y PAUSCH, R. *Training for physical tasks in virtual environments: Tai Chi*. 2003.

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

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.

KYAN, M., SUN, G., LI, H., ZHONG, L., MUNEESAWANG, P., DONG, N., ELDER, B. y GUAN, L. An Approach to Ballet Dance Training Through MS Kinect and Visualization in a CAVE Virtual Reality Environment. *ACM Trans. Intell. Syst. Technol.*, vol. 6(2), páginas 23:1–23:37, 2015. ISSN 2157-6904.

LI, B., MAXWELL, M., LEIGHTLEY, D., LINDSAY, A., JOHNSON, W. y RUCK, A. Development of exergame-based virtual trainer for physical therapy using kinect. En *Games for Health 2014*, páginas 79–88. Springer, 2014.

MEJÍAS, S. M. *Rotoscopia y captura de movimiento. Una aproximación general a través de sus técnicas y procesos en la postproducción*. Tfg, 2014.

- MENÉNDEZ MENDOZA, S. y RODRÍGUEZ MARANTE, J. *Protocolo de trabajo y banco de pruebas para el uso del equipo de captura de movimientos MOCAP-Optitrack*. Tfg, 2015.
- MICROSOFT. *Kinect para Xbox One / Xbox*. <http://www.xbox.com/es-ES/xbox-one/accessories/kinect>, 2010.
- MICROSOFT. *Buy Kinect Adapter for Xbox One S and Windows PC - Microsoft Store*. <https://www.microsoft.com/en-us/store/d/Kinect-Adapter-for-Xbox-One-S-and-Windows-PC/933Z2VKMTHCS>, 2014a.
- MICROSOFT. *Hardware de Kinect*. <https://developer.microsoft.com/es-es/windows/kinect/hardware>, 2014b.
- MICROSOFT. *Kinect: desarrollo de aplicaciones de Windows*, 2014c.
- MICROSOFT. *Microsoft Visual Studio*. [online]. [Accessed 07 June 2017]. <https://www.microsoft.com>, 2017.
- NINTENDO. *Wii / Nintendo*. <https://www.nintendo.es/Wii/Wii-94559.html>, 2006.
- PAREDES, H., CASSOLA, F., MORGADO, L., DE CARVALHO, F., ALA, S., CARDOSO, F., FONSECA, B. y MARTINS, P. *Exploring the Usage of 3D Virtual Worlds and Kinect Interaction in Exergames with Elderly*, páginas 297–300. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08596-8.
- RUTTKAY, Z. y VAN WELBERGEN, H. *Elbows Higher! Performing, Observing and Correcting Exercises by a Virtual Trainer*, páginas 409–416. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-85483-8.
- SIN, H. y LEE, G. Additional virtual reality training using xbox kinect in stroke survivors with hemiplegia. *American Journal of Physical Medicine & Rehabilitation*, vol. 92(10), páginas 871–880, 2013.
- SONY. *Mando de movimiento PlayStation Move / Más formas de jugar / PlayStation*. <https://www.playstation.com/es-es/explore/accessories/playstation-move-motion-controller/>, 2010.
- SONY. *Cámara PlayStation® Eye - PlayStation®3 Accesorios - PS3? Accesorios para PlayStation®*. <http://latam.playstation.com/ps3/accesorios/scph-98047.html>, 2013.
- STAAB, R. *Recognizing specific errors in human physical exercise performance with Microsoft Kinect*. Proyecto Fin de Carrera, California Polytechnic State University, CA, USA, 2014.

TECHNOLOGIES, U. *Unity - Game engine, tools and multiplatform. [online]* . [Accessed 07 June 2017].. <https://unity3d.com/es/public-relations>, 2017a.

TECHNOLOGIES, U. *Unity - Manual: Unity Manual. [online]* . [Accessed 07 June 2017].. <http://docs.unity3d.com/Manual/index.html>, 2017b.

UNITY-TECHNOLOGIES. *Unity - Game engine, tools and multiplatform. [online]* . [Accessed 07 June 2017].. <https://unity3d.com/es/unity>, 2017.

WIKIPEDIA. Captura de movimiento - wikipedia, la enciclopedia libre. 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 —; firmela 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*

