

---

# Uso de Kinect para el entrenamiento de actividades físicas

---



TFG

Víctor Tobes Pérez  
Raúl Fernández Pérez

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

Junio 2017

Documento maquetado con T<sub>E</sub>X<sup>S</sup> v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

# Uso de Kinect para el entrenamiento de actividades físicas

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

*Versión 1.0+*

**Departamento de Ingeniería del Software e Inteligencia  
Artificial**  
**Facultad de Informática**  
**Universidad Complutense de Madrid**  
  
**Junio 2017**

Copyright © Víctor Tobes Pérez y Raúl Fernández Pérez

ISBN 978-84-692-7109-4

# Agradecimientos



# Resumen





# Índice

<b>Agradecimientos</b>	<b>v</b>
<b>Resumen</b>	<b>vii</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Introducción . . . . .	1
<b>2. Estado del arte</b>	<b>3</b>
2.1. Captura de movimiento . . . . .	3
2.2. Historia de la captura de movimiento . . . . .	3
2.2.1. Precursores . . . . .	3
2.2.2. Nacimiento de la captura de movimiento . . . . .	4
2.3. Tecnología captura de movimiento . . . . .	5
<b>3. Sensor Kinect</b>	<b>7</b>
3.1. Versiones de Kinect . . . . .	7
3.1.1. Kinect V1 . . . . .	7
3.1.2. Kinect V2 . . . . .	7
<b>4. Materiales y Métodos</b>	<b>9</b>
4.1. Software y hardware empleados . . . . .	9
4.1.1. Entorno de desarrollo : Unity 3D . . . . .	9
4.1.2. Kinect V2 . . . . .	10
4.1.3. MakeHuman . . . . .	10
4.1.4. Fuse Character Creator . . . . .	11
4.1.5. Mixamo . . . . .	12
4.1.6. Marvelous Designer . . . . .	12
4.1.7. Blender . . . . .	12
<b>5. Desarrollo Del Proyecto</b>	<b>13</b>
5.1. Paquetes para Unity . . . . .	13
5.1.1. Grabar y reproducir movimientos de Usuario . . . . .	15

5.1.2.	Comparación de los movimientos de Usuario . . . . .	18
5.1.3.	Avatares y animaciones . . . . .	19
5.1.4.	Análisis de los movimientos de Usuario . . . . .	24
5.1.5.	Creación de los escenarios 3D . . . . .	25
5.1.6.	Grabación de movimientos con gente experta . . . . .	25
<b>6.</b>	<b>Métodos</b>	<b>27</b>
6.1.	Escenas . . . . .	27
6.1.1.	Escena 1: Inicio . . . . .	27
6.1.2.	Escena 2: Menú Principal . . . . .	28
6.1.3.	Escena 3: Ayuda . . . . .	29
6.1.4.	Escena 4: Créditos . . . . .	29
6.1.5.	Escena 5: Entrenar . . . . .	29
<b>I</b>	<b>Apéndices</b>	<b>31</b>
<b>A.</b>	<b>Así se hizo...</b>	<b>33</b>
A.1.	Introducción . . . . .	33

# Índice de figuras

4.1. Logo MakeHuman . . . . .	11
4.2. Aspecto del humano estándar de MakeHuman . . . . .	11
5.1. Cubeman con la lista de todos los Joints . . . . .	16
5.2. Patrón de Singleton . . . . .	16
5.3. Avatar inicial . . . . .	20
5.4. Avatar con 31 huesos . . . . .	20
5.5. Avatar con 163 huesos . . . . .	21
5.6. Avatar con 137 huesos . . . . .	22
5.7. Avatar con 53 huesos . . . . .	22
5.8. Avatar inicial con ropa . . . . .	23
6.1. Diagrama que ilustra la lógica del sistema . . . . .	28



# Índice de Tablas



# Capítulo 1

## Introducción

### 1.1. Introducción





## 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. ? ?.

### 2.2. Historia de la captura de movimiento

#### 2.2.1. Precursores

Aristoteles (384-322 AC) podría ser considerado el primer biomecánico, 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.

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...

Eadweard Muybridge (seudónimo de Edward James Muggeridge) fué el primer fotógrafo capaz de diseccionar el movimiento humano y animal, a través de multiples cámaras tomando varias fotografías para captar instantes seguidos en el tiempo. Este experimento llamado ".<sup>el</sup> caballo en movimiento",

mostrado en la figura "tal", utilizó esta técnica de fotografía.

COMENTARIO: meter figura

### 2.2.2. Nacimiento de la captura de movimiento

Con la aparición de la técnica de la rotoscopia se consiguió naturalizar los movimientos de los personajes animados. Los estudios Walt Disney Pictures utilizaron la rotoscopia en 1937 en "Blancanieves y los siete enanitos" para la animación de los personajes del príncipe y de Blancanieves. Esta técnica consiste en reemplazar los fotogramas de una grabación real por dibujos calcados sobre cada fotograma (Figura 2.2)

En paralelo, los laboratorios de biomecánica comenzaban a utilizar ordenadores para analizar el movimiento humano. En la década de los 80, Tom Calvert, profesor de kinesiólogía y ciencias de la computación en la universidad Simon Fraser, Canada, incorporó potenciómetros a un cuerpo y la salida la usó para mover personajes animados por ordenador para estudios coreográficos y asistencia clínica para pacientes con problemas de locomoción.

En la década de los 70, cuando empezaba a surgir la posibilidad de realizar animaciones de personajes por ordenador, se conseguía naturalizar los movimientos mediante técnicas clásicas de diseño, como la técnica de rotoscopia. Esta técnica consiste en reemplazar los frames de una grabación real por dibujos calcados cada frame.

Pero mientras, los laboratorios de biomecánica empezaban a usar los ordenadores como medio para analizar el movimiento humano. En la década de los 80, un profesor de kinesiólogía 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.

A finales de los años 70, cuando se empezaba a hacer posible la animación de personajes por ordenador, los diseñadores comenzaron a usar las técnicas clásicas de diseño (como el rotoscopia). El rotoscopia era una técnica en la cual se utilizaban frames reales que se utilizaban como base para diseñar algo por encima, similar a calcar un folio por encima de otro que contiene lo que queremos copiar. Pero mientras, se empezaron a usar los ordenadores para analizar el movimiento humano, en estudios de biomecánica. Las técnicas y dispositivos usados en éstos empezaron a adoptarse en la comunidad de GC.

Al principio de los años 80, Tom Calvert, un profesor de kinesiólogía y ciencias de la computación en la Simon Fraser University, adhirió potenciómetros a un cuerpo y usó la salida para generar personajes animados por ordenador con objeto de ser usado en estudios de coreografía y asistencia clí-

nica para pacientes con problemas de locomoción. Por ejemplo, para analizar la flexión de rodilla, creó una especie de exoesqueleto para cada pierna, cada uno de los cuales tenía adherido un potenciómetro para analizar el grado de flexión. La señal analógica era digitalizada e introducida en un programa que hacía una simulación mediante una animación en el ordenador.

Poco después, comienzan a salir los primeros sistemas de seguimiento visual como el Op-Eye y el SelSpot. A principios de los 80, tanto el MIT como el CGL del NYTC experimentaron con dispositivos de seguimiento visual aplicados en el cuerpo humano.

Estos sistemas normalmente usan pequeños marcadores adheridos al cuerpo (tanto LEDs parpadeantes como pequeños puntos reflectantes) y una serie de cámaras alrededor del espacio de maniobras. Una combinación de hardware especial y software distinguen los marcadores en el campo visual de cada cámara, y mediante comparación, calculan la posición tridimensional de cada marcador en cada instante.

La tecnología está limitada por la velocidad a la que los marcadores pueden ser rastreados (esto afecta al número de posiciones por segundo que pueden ser capturadas), por la oclusión de los marcadores por el cuerpo y por la resolución de las cámaras (específicamente por su capacidad para diferenciar distintos marcadores próximos). Los primeros sistemas podían rastrear sólo una docena de marcadores al mismo tiempo. Los sistemas más recientes pueden distinguir varias docenas. Los problemas de oclusión se pueden superar con el uso de más cámaras, pero incluso con eso, los sistemas ópticos más modernos suelen requerir un post-procesamiento manual para recuperar trayectorias cuando un marcador se pierde de vista. Esto cambiará según los sistemas se vuelvan más sofisticados. El problema de la resolución está relacionado con varias variables, como el precio de la cámara, el campo de visión, y el espacio de movimientos. A mayor resolución requerida, mayor el precio de la cámara. La misma cámara puede dar una mejor resolución de movimiento si está enfocando un menor campo de visión, pero esto limita la capacidad de los movimientos a realizar. Por ello, casi todos los resultados de los sistemas de captura ópticos necesitan una post-producción para analizar, procesar y limpiar la información antes de ser utilizados.

## 2.3. Tecnología captura de movimiento

Captura de movimientos óptica

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

Captura de movimientos inercial



## Capítulo 3

# Sensor Kinect

### 3.1. Versiones de Kinect

#### 3.1.1. Kinect V1

#### Características

Video: 640x480 @30 fps

#### 3.1.2. Kinect V2

COMENTARIO: Enlaces sobre las características de Kinect
---

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



## Capítulo 4

# Materiales y Métodos

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

### 4.1. Software y hardware empleados

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

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

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

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

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

Los entornos a elegir serían Visual Studio 2015, Unity 3D y Unreal Engine 4 (referencia). El primero descartado es desarrollar directamente en Visual Studio 2015 porque se busca crear también un escenario 3D y, tanto Unity como Unreal ,facilitan la creación de estos escenarios.

En cuanto a la decisión de elegir entre Unity 3D o Unreal Engine 4 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 a pequeñas y grandes aplicaciones siendo su aprendizaje de este mas intuitivo que Unreal.

Y por ultimo la elección de lenguaje de programacion siendo el lenguaje de C# una opción más sencilla para el desarrollo de este proyecto.(buscar diferencias claras)

## Comunidad de Unity

Video: 640x480 @30 fps

### 4.1.2. Kinect V2

COMENTARIO: Enlaces sobre las características de Kinect
---

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

### 4.1.3. MakeHuman

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

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

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





Figura 4.1: Logo MakeHuman

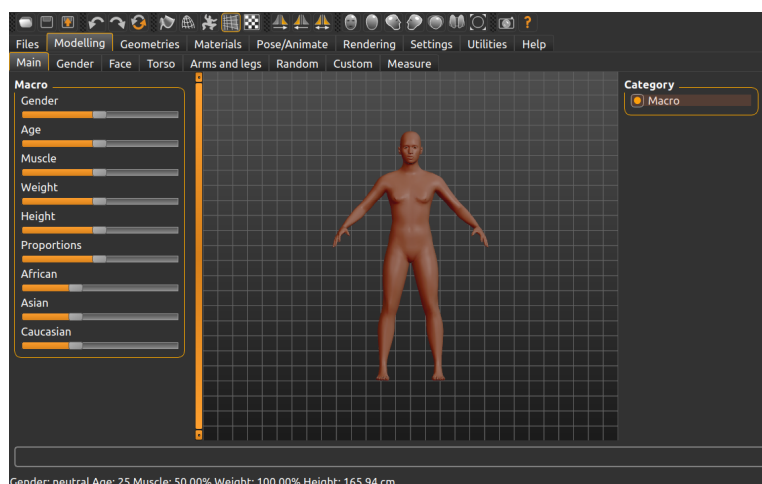


Figura 4.2: Aspecto del humano estándar de MakeHuman

#### 4.1.4. Fuse Character Creator

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

un scrool o desplazamiento 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 y completar la configuración de los huesos, a demás de incluir una posible animación.

#### **4.1.5. Mixamo**

#### **4.1.6. Marvelous Designer**

#### **4.1.7. Blender**

## Capítulo 5

# Desarrollo Del Proyecto

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

### 5.1. Paquetes para Unity

El primer paquete que se prueba es el que nos ofrece microsoft <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 esta especificado para Unity Pro. Aun así, se corrigen errores de comandos y llamadas a funciones obsoletas para ver si se puede aprovechar este paquete o no.

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

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

---

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

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

paquete `Kinect v2 Examples with MS-SDK`(?Referencia o imagen?) que lo elegimos por su valoración positiva y también porque hay poca variedad de paquetes relacionados con kinect v2.

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

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

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

- **PhysicsDemo**, muestra una 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 explicará con más detalle que se utiliza de estos ejemplos.

#### 5.1.1. Grabar y reproducir movimientos de Usuario

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

### Investigación base

En una primera vista nos muestra como representa al usuario captado por la kinect, esta representación se hace mediante un cubeman<sup>3</sup> que simula todo el movimiento que realiza el usuario.

Esta representación de cubeman en el escenario de Unity es creada gracias al script de **Cubeman Controller**. Este script se inicializa con el número del cuerpo que quieres que muestre, kinect v2 puede detectar hasta 6 personas, y también si se quiere que el cuerpo este representado en modo espejo. Los datos del cuerpo del usuario se los pide al **Kinect Manager** (este script se

---

<sup>3</sup>Esqueleto verde que emula el movimiento de la persona captada.

explica más adelante). Los datos del cuerpo están definidos con 25 `GameObjects` llamados `Joints` que hacen una representación de la articulaciones del cuerpo humano en unas coordenadas (x,y,z) y su rotación. Los 25 `Joints` serían : cadera central y laterales , pecho, clavícula, cuello, cabeza, hombros, codos, muñecas, pulgares, manos centrales, rodillas, tobillos y pies.

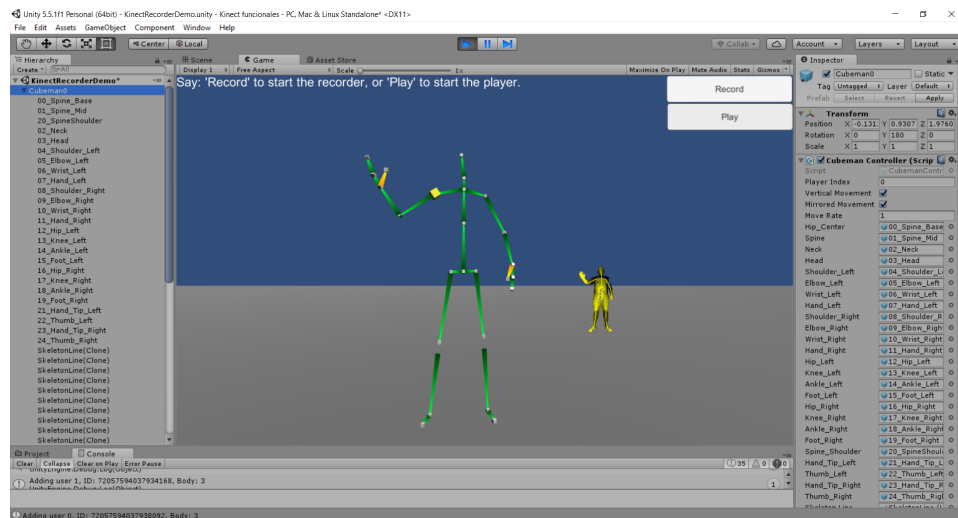


Figura 5.1: Cubeman con la lista de todos los Joints

Para hacer una representación correcta del esqueleto y se mueva como una entidad, el script `Cubeman Controller` pone la posición del Joint de la cadera base como la posición y rotación del objeto padre y todos los demás joint serán hijos de este `GameObject`. 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>4</sup> encargado de la comunicación entre el sensor de kinect y las aplicaciones de Unity.

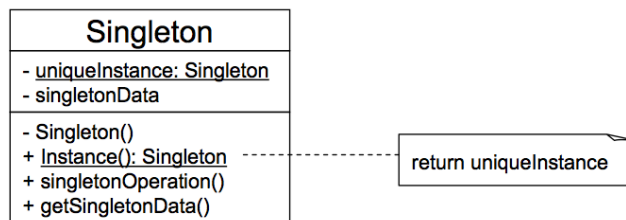


Figura 5.2: Patrón de Singleton

<sup>4</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

Este script al ser común a todos los ejemplos nombrados anteriormente implementa toda la funciones y comunicaciones necesarias para su funcionamiento, pero en nuestro caso se investiga como obtiene la información que le envía al **cubeman Controller**. Kinect Manager se encarga de analizar la información del **SensorData**, este **SensorData** es una estructura de datos ofrecida por el script **KinectInterop**<sup>5</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 contiene la información del número de usuarios captados, su identificador ,la información de los **BodyData**<sup>6</sup>, etc.

Una vez comprendido el **SensorData** se observa como el **Kinect Manager** ,analizando el **BodyDataFrame**, va actualizando y asignando a cada usuario el Id correspondiente de un **BodyData** en todo momento y 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 txt. La información se le pide al **Kinect Manager** y este a su vez al **KinectInterop** devolviéndola en una cadena de caracteres. Esta cadena hace referencia a un frame<sup>7</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 Id y después sus 25 joints, los **BodyData** no captados aparecerá un cero. **KinectRecorderPlayer** también discierne entre grabación y reproducción activando solo una de las dos a la vez.

COMENTARIO: foto de txt
-------------------------

## Cambios e implementaciones

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

En primer lugar debemos implementar la forma de poder reproducir un movimiento grabado a la vez que muestra al usuario captado en tiempo real. Para ello implementamos una función llamada **SetBodyFrameFromCsvAndPoll** en el **KinectInterop** ,para así 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 rellenarlo con los datos del fichero txt.

---

<sup>5</sup>Script encargado de tener comunicación directa con el sensor de la kinect

<sup>6</sup>Información de los joint de un cuerpo específico

<sup>7</sup>Es un fotograma, Unity por defecto reproduce a 60 frames por segundo

También se añade un campo más a la estructura de `BodyData` para saber si los datos son procedentes del sensor o de un fichero. Posteriormente de hacer estos cambios, se realiza una modificación en el script `CubemanController` para poder escoger de que usuario se quiere mostrar el movimientos si proceden del sensor o de un fichero. 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.

COMENTARIO: foto esqueletos verdes con las dos camaras
--

### 5.1.2. Comparación de los movimientos de Usuario

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

## Investigación base

La primera forma de comparar un movimiento en la que se debatió fue la de estudiar las trayectorias de cada movimiento, calculando 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 intento tener otro camino para hacer la comparación de forma más genérica. Con esta idea en mente nos fijamos en el ejemplo de `GestureDemo` que reconoce gestos del usuario para rotar y hacer zoom sobre un cubo. Para que reconozca estos gestos previamente hay que añadirlos a las lista de gestos del script `KinectGesture` e implementar su funcionalidad en el método `CheckForGesture`. Posteriormente hay que notificar al `KinectManger` que gesto de los registrados queremos que se detecte. 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.

## Cambios e implementaciones

Una vez analizado la funcionalidad de los gestos de kinect nos apoyaremos 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* e implementamos 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*, esta información será accedida en el estado. 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 decidimos que para sea más eficiente el salto al siguiente estado no incrementaría en uno sino en más unidades, ya que, la comparación de dos *frames* consecutivos es muy similar y así se ahorrarían comparaciones innecesarias. Para pasar al siguiente estado habrá una comparación de cada uno de los 25 *joints* del usuario con los *joints* del *frame* correspondiente. La comparación consiste en igualar las coordenadas x e y con un margen de error modificable por el usuario, si cumple este requisito pasará al siguiente estado. La comparación tiene un tiempo límite que sería 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. El resultado del gesto se muestra por pantalla para dar un *feedback* inmediato al usuario.

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

COMENTARIO: Grafo de comparar movimiento
--

### 5.1.3. Avatares y animaciones

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

COMENTARIO: Foto de la configuración de rig en unity
--

La primera aplicación que se utilizó para crear los personajes que darían vida al TFG fue MakeHuman, ya que mostraba una interfaz amigable e intuitiva. El primer avatar utilizado fue simple, ya que al principio interesaba ver el resultado que proporcionaba el personaje junto con el *cubeman* y de esta forma determinar el número de huesos utilizados que mas se ajustaba a las necesidades dadas. Para comprobar que esqueleto se debía utilizar, se

tuvieron que crear cuatro avatares diferentes, con 31, 163, 137 y 53 huesos respectivamente.

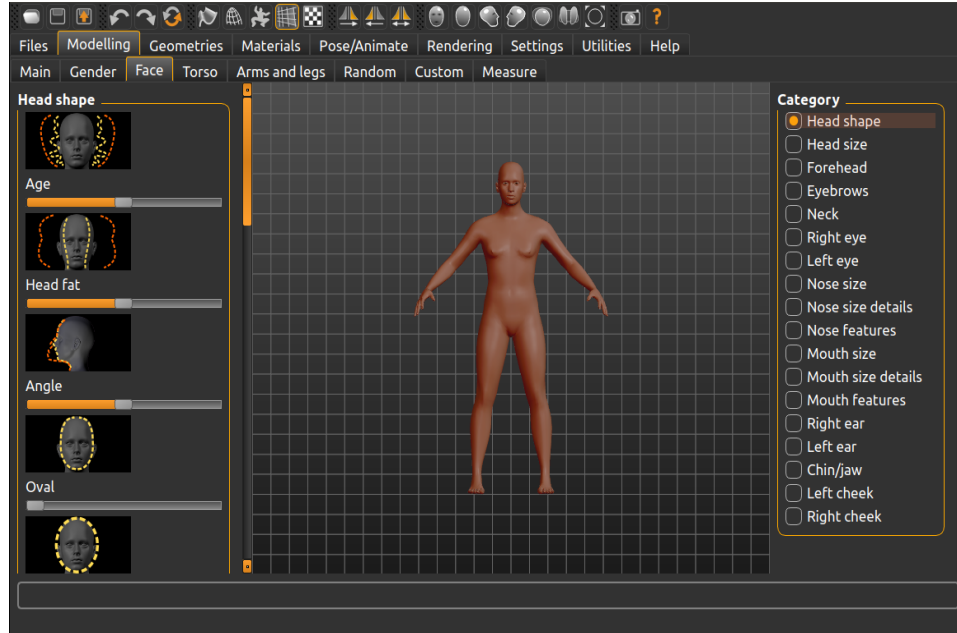


Figura 5.3: Avatar inicial

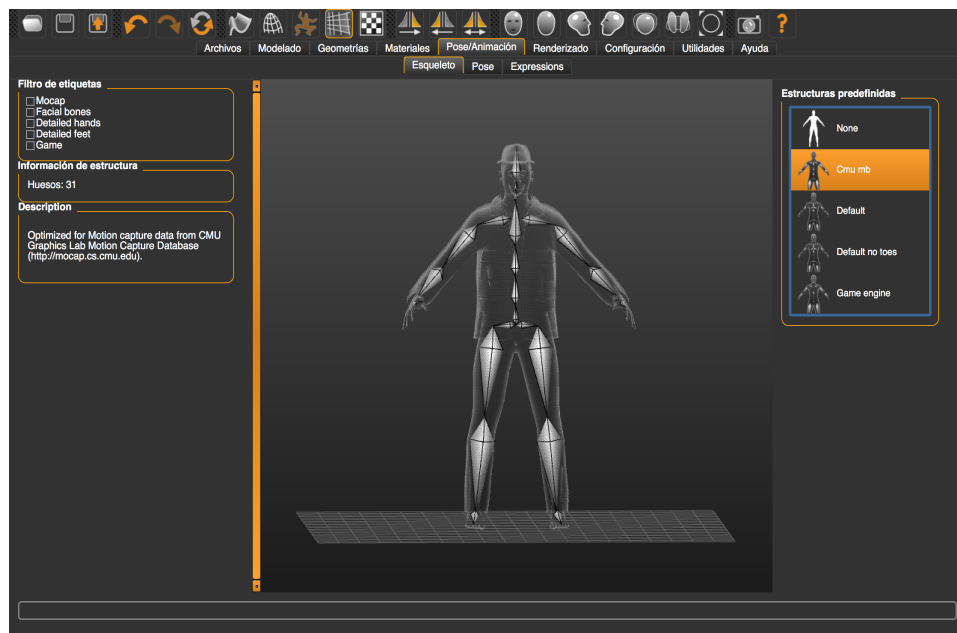


Figura 5.4: Avatar con 31 huesos

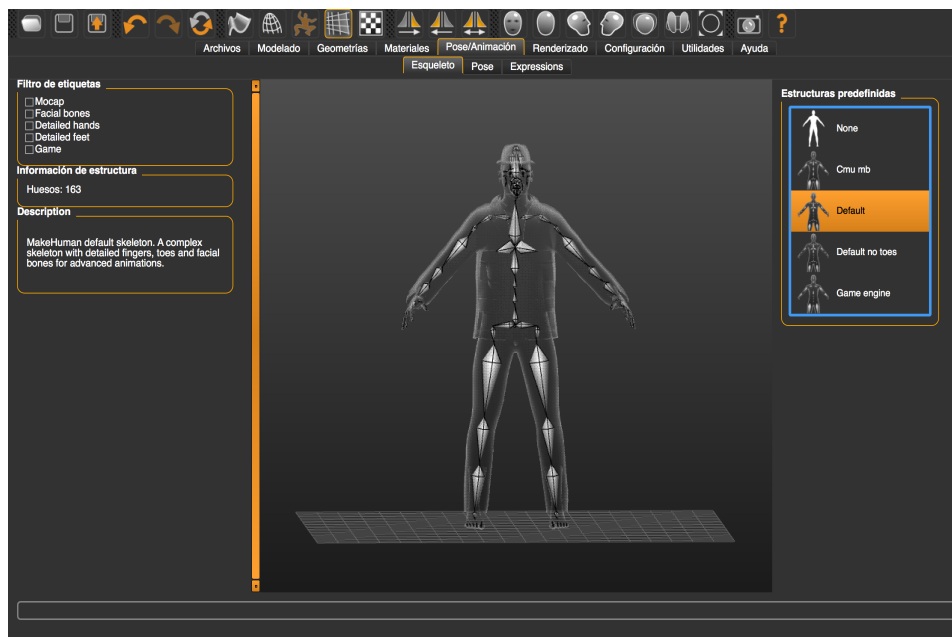


Figura 5.5: Avatar con 163 huesos

Después de realizar varias pruebas, se determinó que el avatar que mejor se adaptaba a los movimientos capturados por *kinect* era *Default no toes*. 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. 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.

Tras la decisión tomada de cambiar a Fuse Character Creator como aplicación para el desarrollo de los personajes

## Investigación base

Para saber como animar un avatar nos fijamos en el ejemplo **AvatarsDemo**, para ello utiliza un script **AvatarController** que realiza una conexión entre los joints y los huesos incorporados en el avatar. RAUL TU SIGUES CON LOS PROGRAMAS Y ANIMACIONES.

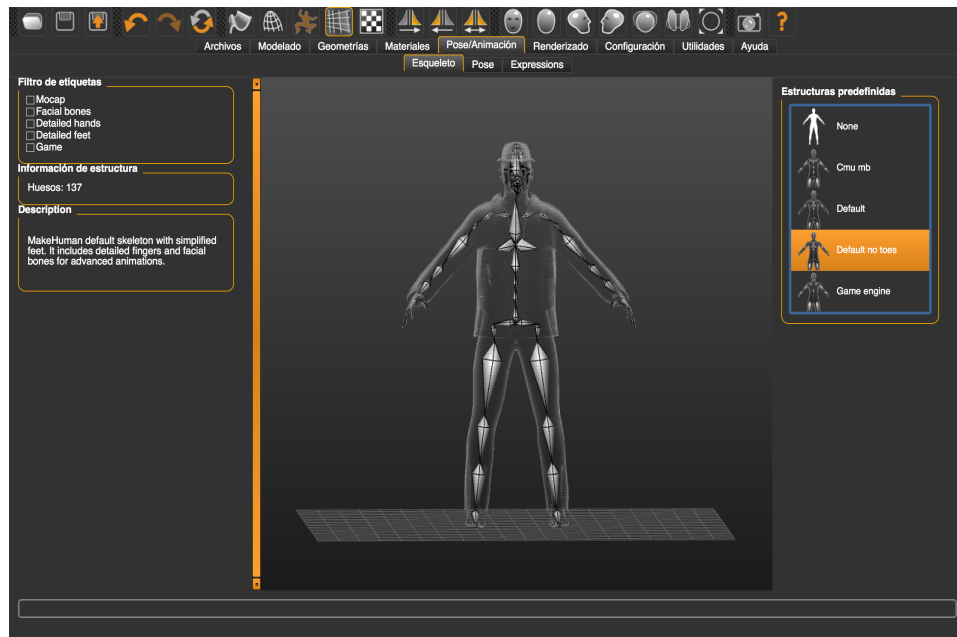


Figura 5.6: Avatar con 137 huesos

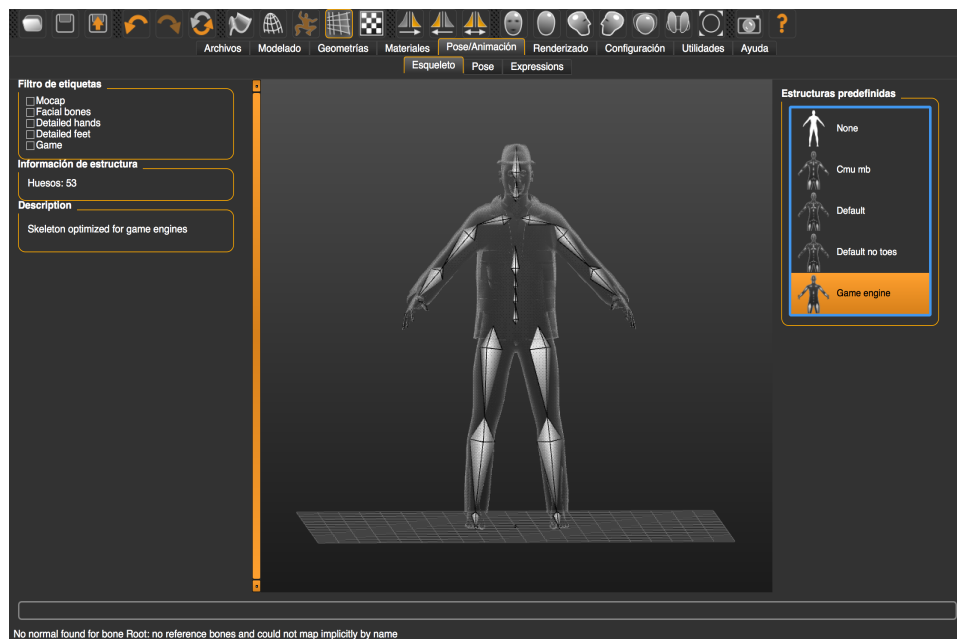


Figura 5.7: Avatar con 53 huesos

## Cambios e implementaciones

Una vez comprendido la generación de avatares, se realizan una prueba con cuatro avatares que se diferencian en la cantidad de huesos que llevan

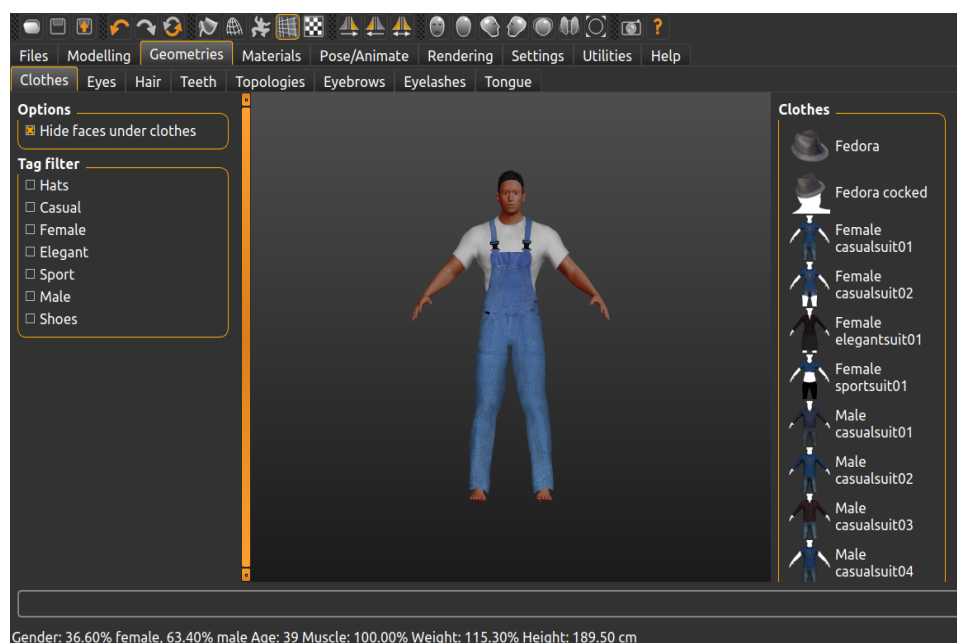


Figura 5.8: Avatar inicial con ropa

incorporados. El avatar elegido es el que contiene 52 huesos porque demuestra una movilidad mas semejante al *cubeman* inicial.

COMENTARIO: captura con los avatar integrando de makehuman con huesos en unity

Posteriormente para que los avatares no se queden en una posición estática se incorporan la animaciones como el movimiento de la ginga y las animaciones de victoria y derrota para el feedback de la comparación de movimiento.

COMENTARIO: tres fotos ginga victoria y derrota

Para incorporar estas animaciones se crea un **Animator Controller** que es un componente de Unity encargado de añadir y gestionar animaciones a un avatar. Este componente tiene una máquina de estados para gestionar que animación se ejecuta en cada momento. En nuestro caso el estado inicial será la animación de la ginga y realizara una transición a la animación de victoria o derrota si recibe un disparador de la comparación de movimiento y una vez terminada volverá a la animación de ginga.

COMENTARIO: imagen Grafo animator

Hay que constatar que hay un conflicto entre el **Animator Controller** y el sensor de movimiento por quien tiene el control del avatar, por ello, se ha implementado un script que cuando el sensor de kinect detecta al usuario automáticamente se desactiva el componente **Animator Controller** y

cuando el usuario se sale de escena el componente vuelve a activarse.

#### 5.1.4. Analisis de los movimientos de Usuario

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

##### Investigación base

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

El planteamiento para el análisis comienza reproduciendo el movimiento del avatar del usuario junto al avatar de la grabación, deteniéndose 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 parte de cuerpo, dirección y altura que servirá de pautas para que el usuario corrija su posición.

##### Cambios e implementaciones

Una vez planteada las especificaciones del análisis se empieza con su implementación. En primer lugar se creó un script llamado **Registrar\_movimientos**, el cual, guarda toda la información del movimientos hasta donde el usuario comete el error. Esta información contiene el estado<sup>8</sup> donde se produce el error y los datos de los *joints* de los avatares. Posteriormente se necesita hacer una modificación en **KinectGesture** en la parte de la funcionalidad del gesto *Move*, para poder rellenar esa información. Esta modificación consiste en enviar la información del estado y los *joints* de los avatares al **Registrar\_movimientos**, una vez superada la comparación de ese *frame*. Con esta lógica se quedará registrado el movimiento hasta que el usuario cometa el error.

A partir de este momento, para poder mostrar la información recogida se implementa una funcionalidad nueva en el script **Registrar\_movimientos**, que consiste en desplazar los avatares a primera escena, modificando los materiales, para que se vean transparentes y generar un esqueleto verde parecido a los *cubeman* del principio. Estos avatares simularán el movimiento hasta el *frame* donde se equivocó el usuario y realizará una comprobación

---

<sup>8</sup>Corresponde al frame que se esta comparando.

de cada *joint* para cambiar el hueso de color de verde a rojo. A la vez, se va generando un *string* con una corrección de posicionamiento compuesto por tres partes. Para la primera parte se crea un diccionario que, a partir del número del *joint*, devuelve la parte del cuerpo correspondiente. En la implementación de este diccionario se han agrupado *joints*, ya que varios forman una misma parte del cuerpo, como por ejemplo, las manos que lo forman tres *joints* diferentes. La segunda parte del *string* esta formada por la elección de la dirección, que se comprueba, realizando la diferencia de la coordenadas X del *joint* del usuario con la del *joint* de la grabación. Si la diferencia es positiva la dirección será hacia derecha, en caso contrario, será hacia la izquierda. Y para la tercera parte del *string* se realiza la diferencia con la coordenada Y definiendo así la altura. Si la diferencia es positiva la altura sera hacia abajo, en caso contrario hacia arriba. Esta lista de correcciones se mostrará en un panel una vez completada evitando frases repetidas.

COMENTARIO: Imagen completa de analisis con panel
---

#### 5.1.5. Creación de los escenarios 3D

#### 5.1.6. Grabacion de movimientos con gente experta





## Capítulo 6

# Métodos

### 6.1. Escenas

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

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

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

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

#### 6.1.1. Escena 1: Inicio

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

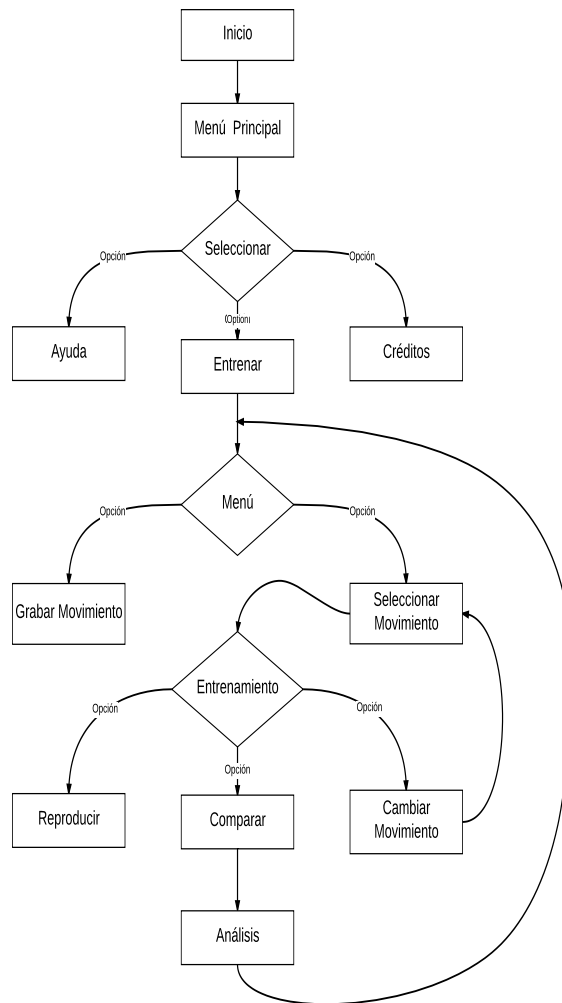


Figura 6.1: Diagrama que ilustra la lógica del sistema

### 6.1.2. Escena 2: Menú Principal

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

### 6.1.3. Escena 3: Ayuda

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

### 6.1.4. Escena 4: Créditos

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

### 6.1.5. Escena 5: Entrenar

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

#### 6.1.5.1. Profesor

En el caso de que se trate del profesor, se podrá acceder a la funcionalidad añadida de la grabación de movimientos. Tras seleccionar esta opción, será necesario introducir el nombre deseado para el nuevo movimiento y colocarse a unos tres metros de Kinect para ser capturado correctamente como ya se explicó anteriormente. Para una mayor inmersión, en la aplicación se utilizará el modo espejo como efecto para reproducir los movimientos capturados.

La aplicación viene definida con ocho movimientos capturados con la escuela Abadía-Capoeira. De este modo, el profesor también podrá utilizar la funcionalidad del alumno, ya que será necesario revisar y reproducir el correcto funcionamiento de los nuevos movimientos capturados con Kinect.

Por lo tanto, el profesor consultará los movimientos que el alumno ha conseguido completar en los diferentes niveles de dificultad y por ello, podrá añadir los movimientos que crea oportunos para la correcta evolución del alumno.

#### 6.1.5.2. Alumno

Si se trata del alumno, se entrará en la escena principal del TFG, 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.

Para iniciar el entrenamiento 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.

En la siguiente escena, considerando que al comenzar a formarse en un arte marcial se desconocen los nombres de los movimientos asociados a las transiciones del mismo, el alumno podrá visualizar previamente el movimiento antes de empezar a practicar.

Posteriormente a la selección del movimiento, el alumno deberá iniciar la comparación mediante el botón correspondiente. Al pulsar sobre el botón, el entrenador virtual tomará una pose inicial correspondiente al movimiento, hasta que el alumno no tome esa misma pose, el sistema estará esperando la calibración. En el instante que la pose del alumno sea la misma que la del entrenador virtual, el sistema estará calibrado y se empezará una cuenta atrás de tres segundos, la cual tras finalizar empezará a reproducir el movimiento capturado. En el momento que el entrenador virtual inicie una transición de movimientos, el alumno deberá imitarlo lo mas perfectamente posible. Mientras se realizan las transiciones, el sistema comparará las posición de los 25 joints que anterior mente se han explicado 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, o una animación de fracaso en el caso de que se produzca algún error en la ejecución del movimiento. Tras de sí, el análisis presenta un esqueleto de las partes del cuerpo que tienen que ser corregidas en color rojo y en color verde las que deben permanecer invariantes, 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.

Parte I

Apéndices



# Apéndice A

## Así se hizo...

...

...

**RESUMEN:** ...

### A.1. Introducción

...





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

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

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

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

