Capítulo 5

Desarrollo Del Proyecto

Una vez elegida la tecnología y 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 que información obtenida del sensor de movimiento podemos aprovechar para el proyecto.

5.1. Paquetes para Unity

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

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

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

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

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

5.1.1. Grabar y reproducir movimientos de Usuario

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

Investigación base

En una primer vista nos muestra como representa al usuario captado por la kinect, esta representación se hace mediante un cubeman[3] 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 persona, y también si se quiere que el cuerpo este representado en modo espejo. Los datos del cuerpo del usuario se los a pide al `Kinect Manager` (este script se

[3]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 representacion 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.

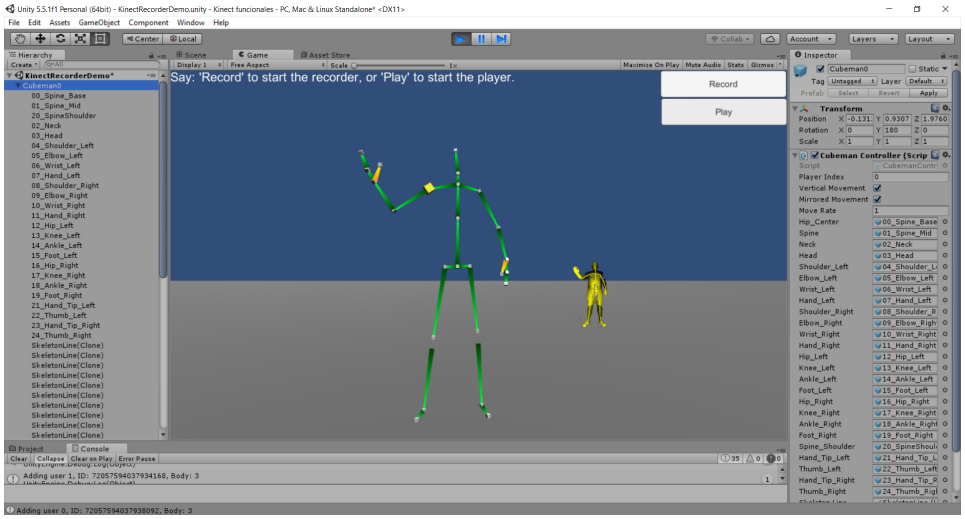

Figura 5.1: Cubeman con la lista de todos los Joints

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 poscion y rotación del objeto padre y todos los demás joint serán hijos de este GameObject. Para calcular la posión 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 Singletone[4] encargado de la comunicación entre el sensor de kinect y las aplicaciones de Unity.

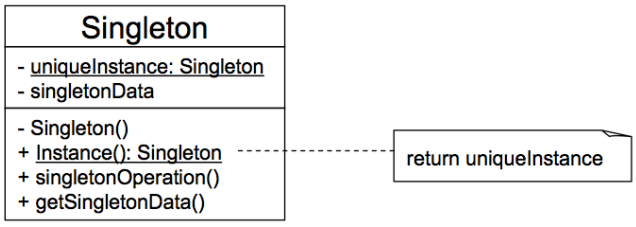

Figura 5.2: Patrón de Singleton

[4]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`[5] donde aparece la información de la imagen, pronfundidad, color y datos de los usuarios captado con el sensor. Los datos de los usuarios captados se encapsulan en el `BodyDataFrame`, esta estructura contiene la información del número de usuarios captados, su identificador ,la información de los BodyData[6], etc.

Una vez comprendido el SensorDatar se observa como el Kinect Manager ,analizando el BodyDataFrame, va actualizando y asignando a cada usuario el Id correspodiente 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[7] que tiene la información del instante de tiempo, los BodyData captados y las coordenadas (x,y,z) respecto al mundo de todos sus joinst. Los BodyData captados aparece primero su Id y después sus 25 joints, los BodyData no captados aparecerá un cero. `KinecRecorderPlayer` también discierne entre grabación y reproducción activando solo una de la 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.

[5]Script encargado de tener comunicación directa con el sensor de la kinect
[6]Información de los joint de un cuerpo especifico
[7]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 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 especifica y laboriosa para cada movimiento, por lo que se intento tener otro camino para hacer la comparación de forma más genérica. Con esta idea en mente 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 especifica. 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 implemenentar 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 siguiente estados al estado cero. El número del estado será utilizado para acceder y obtener el *frame* de la lista del movimiento. En este momento 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.

Por problemas de profundidad de *kinect* se ha obviado la comparación de la coordenada z 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 numero 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.

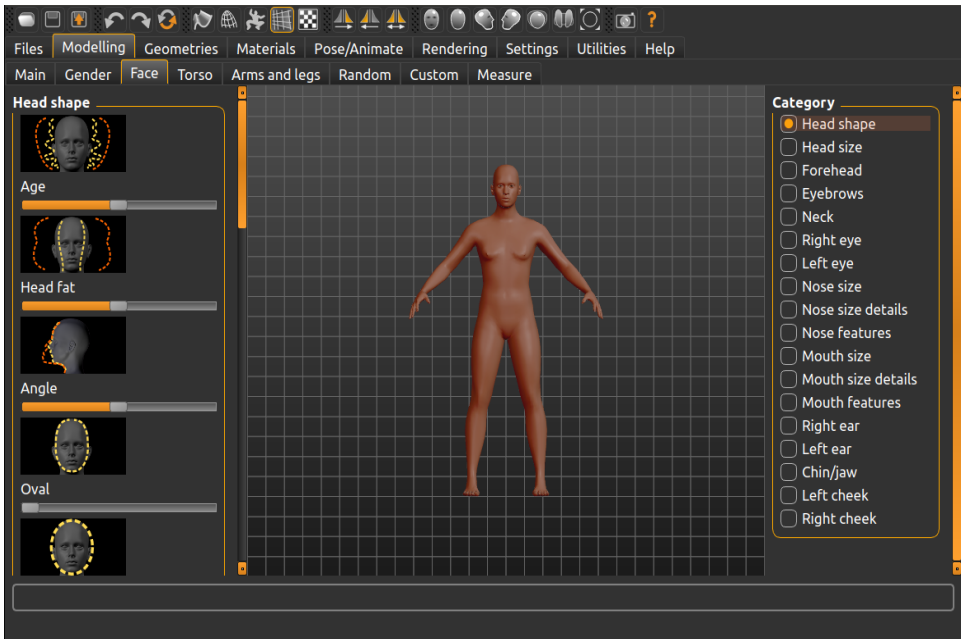

Figura 5.3: Avatar inicial

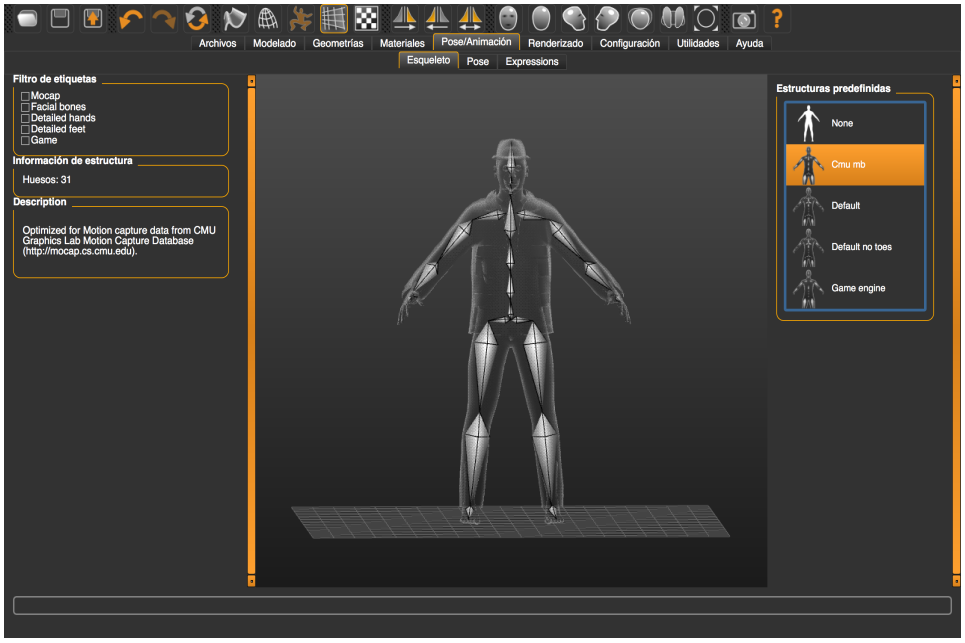

Figura 5.4: Avatar con 31 huesos

Después de realizar varias pruebas, se determinó que el avatar que mejor

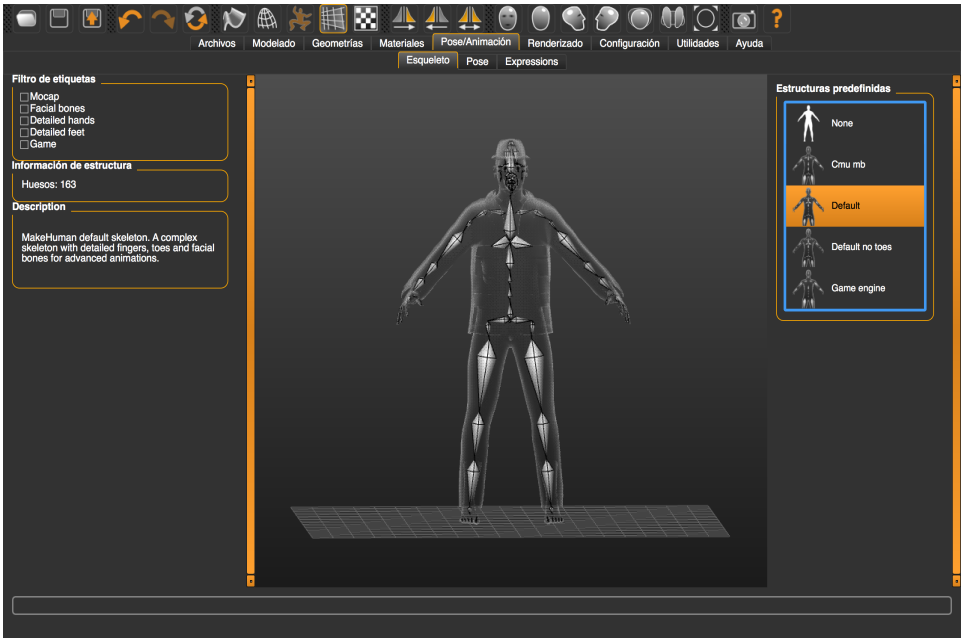

Figura 5.5: Avatar con 163 huesos

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*.

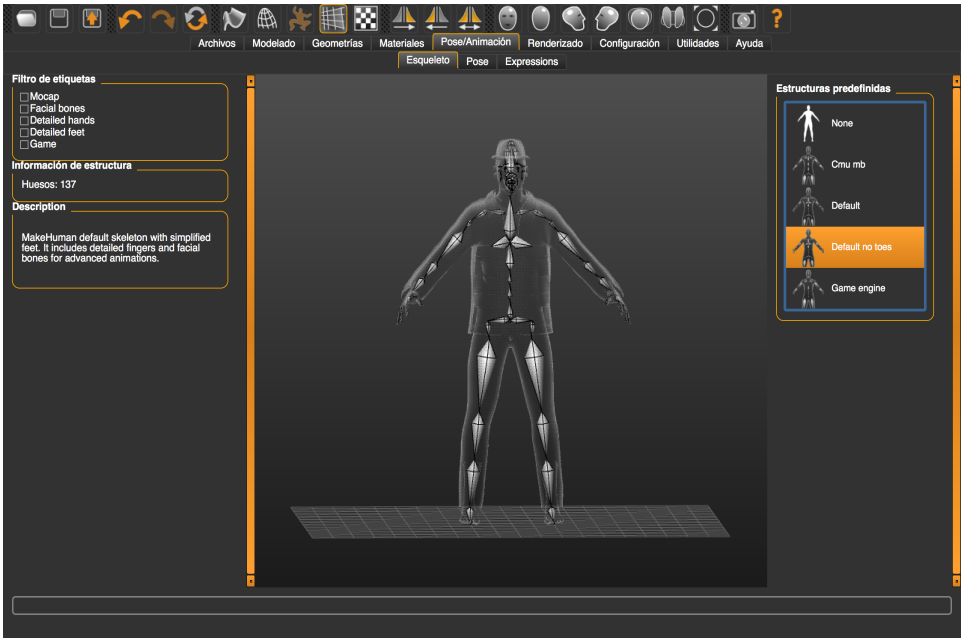

Figura 5.6: Avatar con 137 huesos

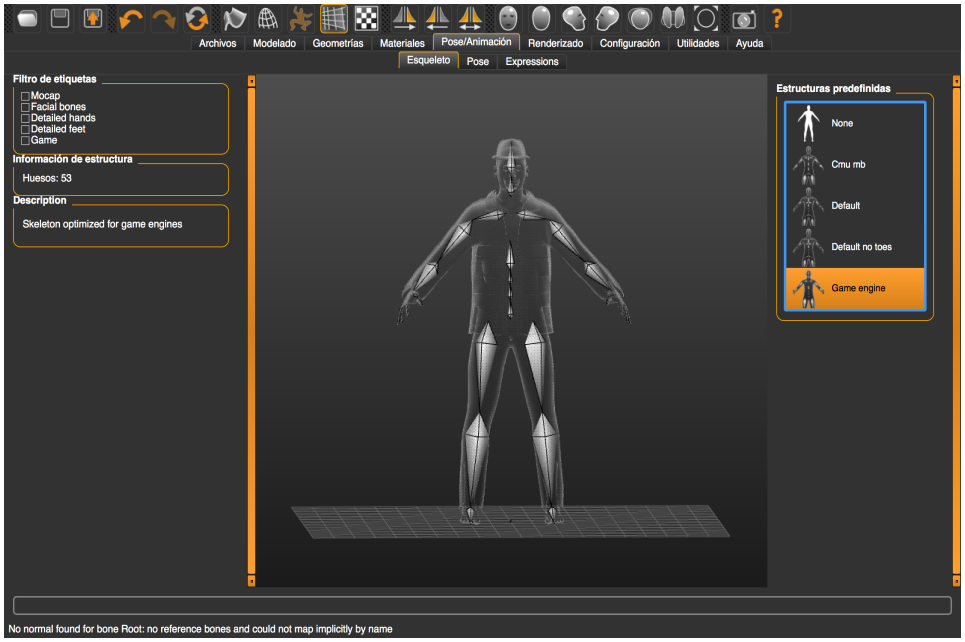

Figura 5.7: Avatar con 53 huesos

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.

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 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

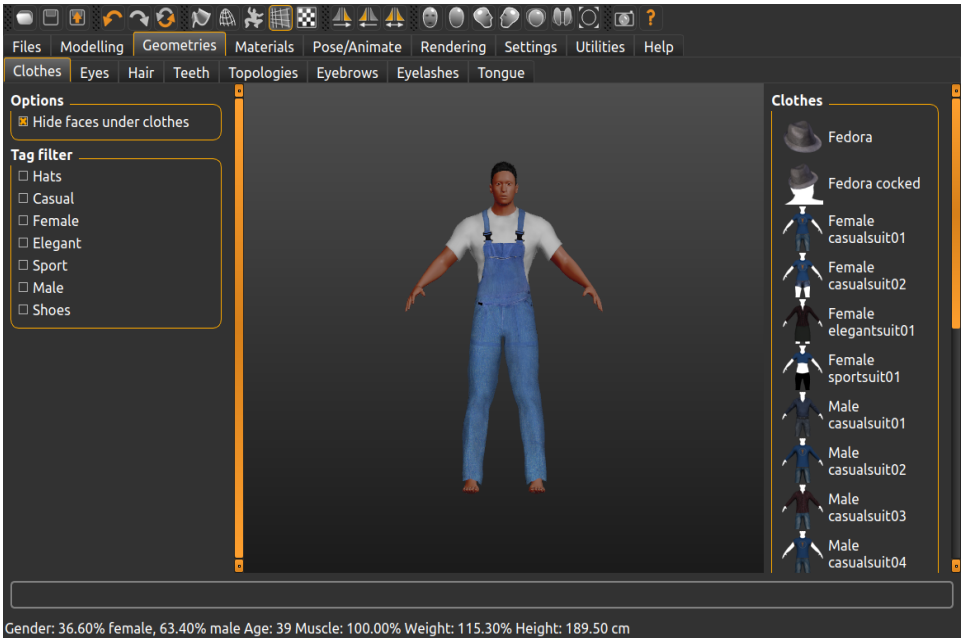

Figura 5.8: Avatar inicial con ropa

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 desactivar 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

5.1.5. Creación de los escenarios 3D

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

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