Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Animaciones Faciales a Tiempo Real de Personajes 3D en Entornos Virtuales

Real-time Facial Animations of 3D Characters in Virtual Environments

Daniel Oria Martín

La Laguna, 14 de julio de 2023

D. **Isabel Sánchez Berriel**, con N.I.F. 42.885.838-S profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

D. **Fernando Andrés Pérez Nava**, con N.I.F. 42.091.420-V profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Animaciones Faciales a Tiempo Real de Personajes 3D en Entornos Virtuales"

ha sido realizada bajo su dirección por D. **Daniel Oria Martín**,

con N.I.F. 51.167.060-H.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 14 de julio de 2023

Resumen

El trabajo de fin de grado se centra en desarrollar animaciones faciales en 3D para personajes en una aplicación de realidad virtual de San Cristóbal de La Laguna. El objetivo principal de estas animaciones faciales es mejorar la inmersión de la aplicación, ya que la realidad virtual busca proporcionar experiencias envolventes y realistas para los usuarios.

Para lograr este propósito, se han empleado diversas tecnologías como Unity, Blender o Meta Quest (Oculus)

El enfoque principal del trabajo ha sido perfeccionar las expresiones faciales de los personajes virtuales, ya que la comunicación no verbal juega un papel fundamental en la inmersión y la interacción natural dentro de la realidad virtual. Mediante el uso de las tecnologías mencionadas, se ha logrado crear animaciones faciales detalladas y realistas que reflejan las emociones y expresiones de los personajes de manera convincente.

Palabras Clave: Realidad Virtual, Diseño 3D

Abstract

This Final Degree Project focuses on the development of 3D facial animations for different characters within a virtual reality application in San Cristóbal de La Laguna. The main objective of these facial animations is to enhance the immersion of the application, as virtual reality aims to provide immersive and realistic experiences for users.

To achieve this purpose, various technologies such as Unity, Blender and Meta Quest (Oculus) have been used.

The main focus of the project has been to perfect the facial expressions of virtual characters, as non-verbal communication plays a fundamental role in immersion and natural interaction within virtual reality. By using the mentioned technologies, detailed and realistic facial animations have been created that convincingly reflect the emotions and expressions of the characters.

Keywords: Virtual Reality, 3D Design

Índice

Índice de figuras

1. Introducción

1.1. Contexto y justificación

A finales del siglo pasado, la UNESCO [1] declaró el casco antiguo de San Cristóbal de La Laguna Patrimonio de la Humanidad [2]. El centro histórico de La Laguna es un ejemplo de planificación urbana innovador que tuvo una gran importancia para el desarrollo de ciudades coloniales españolas en América. En el siglo XVI, el casco histórico de la ciudad quedó configurado tal y como lo conocemos y, aunque hoy se transformen algunas edificaciones, sus calles y su estructura permanecen. Podemos observar esto en el primer plano que se conserva de la ciudad, realizado en 1588 por el ingeniero militar Leonardo Torriani [3].

Por otro lado, el uso de realidad virtual es algo que vemos cada vez más presente en nuestro día a día en distintas aplicaciones, desde unas para la venta de inmuebles hasta juegos o películas realizadas enteramente para realidad virtual.

La atención es algo importante a la hora de captar y retener información. Si se tiene cautivada la atención de una persona, es mucho más probable que se consiga transmitir una información de manera efectiva. El área de las expresiones faciales siempre ha sido y sigue siendo un gran desafío para animadores y programadores gráficos. Recrear movimientos faciales en realidades virtuales puede ser un gran desafío, pero también una manera de captar mejor la atención de un usuario.

1.2. Antecedentes

El trabajo de Fin de Grado se encuentra dentro del proyecto de “Reconstrucción histórica Virtual de San Cristóbal de La Laguna” [4], en esta línea se han realizado otros trabajos de Fin de Grado años atrás. En dichos proyectos se han diseñado y modelado entornos 3D para la arquitectura, vestimenta y costumbres de la sociedad lagunera del siglo XV, basadas en el plano del militar Leonardo Torriani [3].

La aplicación que se seguirá desarrollando inicialmente es una visita virtual a San Cristóbal de La Laguna. Esta sitúa al usuario en el corazón de la ciudad con la finalidad de que este aprenda al mismo tiempo que explora el escenario e interactúa con los pesonajes.

Como la aplicación se enfoca a la educación, cuenta con un sistema de guía que, al apuntar a algún lugar de interés, muestra información tanto por texto como audio.

1.3. Objetivo del Trabajo

El objetivo de este trabajo busca mejorar la animación facial de los personajes de este proyecto educativo en 3D y realidad aumentada de San Cristóbal de La Laguna, para conservar, consolidar y valorizar el patrimonio histórico gracias al uso de tecnologías en auge estos últimos años.

La finalidad es conseguir que un personaje guía, no solo muestre esta información en un texto mientras se reproduce un audio, sino que también tenga una animación en su rostro a medida que se reproduce dicho audio. Esto permite una mayor sensación de inmersión en la aplicación y consigue captar mejor la atención del usuario.

2. Fundamentos teóricos

2.1. Realidad virtual y su inmersión

La realidad virtual ha despertado interés durante décadas y, a pesar de estar en sus etapas iniciales de desarrollo, su potencial futuro es prometedor. Una de las características más destacadas de la realidad virtual es su capacidad para sumergir a los usuarios en entornos virtuales, brindando experiencias sensoriales inmersivas y únicas [5].

La inmersión en la realidad virtual se logra mediante tecnologías avanzadas que estimulan los sentidos y crean una sensación de presencia virtual. Los visores o gafas de realidad virtual permiten a los usuarios explorar y visualizar entornos tridimensionales, sumergiéndolos por completo en un mundo digital. Además, los dispositivos de realidad virtual pueden incluir auriculares con sonido envolvente, lo que contribuye aún más a la sensación de estar presente en el entorno virtual.

Esta inmersión en la realidad virtual tiene un impacto significativo en diversos campos y sectores. En el ámbito del entretenimiento, por ejemplo, los usuarios pueden sumergirse en videojuegos y experiencias interactivas que superan los límites de los métodos tradicionales [6]. Los jugadores pueden interactuar con personajes y entornos de manera más natural, experimentando emociones intensas. Esto abre nuevas posibilidades creativas para desarrolladores y diseñadores, y ofrece a los jugadores una forma más inmersiva de disfrutar de estas actividades [7].

Además del entretenimiento, la inmersión en la realidad virtual tiene aplicaciones prácticas en campos como la medicina, la arquitectura y la educación. Los estudiantes pueden realizar

visitas virtuales a lugares históricos, explorar el interior del cuerpo humano o participar en simulaciones interactivas para adquirir conocimientos de manera práctica y envolvente.

La inmersión proporcionada por la realidad virtual también tiene implicaciones en la interacción y comunicación entre personas. Con el avance de la tecnología, se espera que la realidad virtual permita experiencias sociales más realistas y la conectividad entre usuarios de todo el mundo. Esto podría revolucionar la forma en que nos relacionamos y colaboramos, creando un sentido compartido de presencia a pesar de la distancia física, eliminando las barreras geográficas.

2.2. Animaciones faciales y su importancia

La importancia de lograr animaciones faciales realistas en los personajes de una aplicación de realidad virtual educativa no puede subestimarse. Estas animaciones faciales permiten una comunicación más efectiva y una conexión emocional más profunda entre los usuarios y los personajes virtuales, lo que en última instancia mejora la experiencia de aprendizaje.

Los rostros humanos son una fuente rica de información no verbal, ya que transmiten una amplia gama de emociones y expresiones. Al replicar estas expresiones faciales de manera realista en los personajes de realidad virtual, se les dota de una mayor capacidad para comunicarse y transmitir información de manera efectiva. Esto es relevante en el contexto educativo, donde la comunicación verbal y no verbal es fundamental en la transmisión de conceptos y conocimientos [8].

2.3. Tecnologías utilizadas

2.3.1. Unity

Unity [9] es una plataforma de desarrollo de Software líder en la industria de los videojuegos y la realidad virtual. Es un motor de juego versátil, lo que permite a desarrolladores crear experiencias interactivas de alta calidad para una amplia variedad de plataformas.

Con Unity se pueden crear potentes gráficos, la plataforma ofrece una amplia gama de herramientas integradas para ayudar en la creación de aplicaciones. Incluye un editor intuitivo y potente, bibliotecas de assets predefinidos y C# como lenguaje de programación para la lógica de las aplicaciones.

Esta herramienta se usa no solo en el campo de los videojuegos, también está presente en muchos otros sectores como el de la arquitectura, la medicina o la enseñanza, en los que se desarrollan aplicaciones interactivas, experiencias didácticas o simulaciones. [13], [14]

2.3.2. Blender

Blender [10] es un software de modelado y animación 3D de código abierto [11]. Es una herramienta versátil que permite crear distinto contenido visual, desde gráficos en 3D hasta animaciones o efectos especiales.

Cuenta con un conjunto de herramientas robustas y una interfaz intuitiva que lo hacen accesible tanto para principiantes como para usuarios más avanzados. Además, el hecho de ser código abierto fomenta la colaboración y el intercambio de conocimiento de sus usuarios.

2.3.3. Oculus

Oculus [12] es una compañía líder en el campo de la realidad virtual que fue adquirida por Meta [13] en 2014. Meta, anteriormente conocida como Facebook, viendo el potencial de la realidad virtual como una plataforma transformadora, adquirió Oculus con la visión de construir el Metaverso [14]. Meta ha estado invirtiendo significativamente en el desarrollo del metaverso, buscando crear una plataforma que permita a las personas trabajar, socializar, aprender y entretenerse en un espacio virtual tridimensional.

2.3.4. Oculus Lipsync

Oculus Lipsync es un plugin de Unity que se usa para sincronizar el movimiento de los labios con los sonidos del habla. Lipsync analiza la entrada de un micrófono o de un audio y predice un conjunto de valores llamados visemas [15]. Los visemas son gestos o expresiones de los labios y la cara que se corresponden con un sonido en concreto, son la representación visual de los fonemas en la animación 3D. Cada uno de los visemas apunta a una transformación (o deformación) geométrica específica de un avatar para influir en la cantidad de transformación que se expresará en el modelo, por ejemplo, al visema asociado con la letra P, cuando se detecte la palabra “pop”, se le asociará un valor decimal entre 0 y 1 que corresponde al nivel de transformación que expresará el modelo en la cara. Los visemas se interpolan entre ellos para simular así el movimiento natural de la boca en el habla.

3. Estado del arte

3.1. Investigaciones previas sobre animaciones faciales en realidad virtual

Al comenzar este trabajo hubo una búsqueda de información acerca de distintas aplicaciones relacionadas con las animaciones, en concreto animaciones faciales orientadas a realidad virtual. La mayor parte de los estudios que se encontraron tratan de cómo, mediante técnicas de Inteligencia Artificial y Machine Learning, se estudian y se hace un seguimiento de los movimientos faciales para aplicarlos a un avatar virtual en 3D. [16], [17]

Dado que se trata de un campo puntero de investigación, muchas de estas tecnologías son privadas e inaccesibles, pero existen varias públicas y gratuitas, con las que se abordará este proyecto.

3.2. Aplicaciones existentes y sus características

Existen muchas aplicaciones interesantes de realidad virtual, brevemente se detallarán dos juegos de VR que han servido de inspiración para pensar el desarrollo de este trabajo.

El primero de ellos se trata del Half-Life: Alyx [6], un juego mucha gente considera revolucionario y que personalmente cambió mi modo de ver la realidad virtual. Lo que más me impresionó fue que es un juego que permite una interacción muy orgánica con cualquier material del entorno, puedes desde rebuscar cosas en una estantería moviendo cada uno de los objetos que hay en ella hasta asomarte por el borde de una pared mínimamente rota, se siente realmente como que formas parte del entorno virtual en el que te encuentras. Con este juego me di cuenta del potencial que realmente tiene la realidad virtual y una gran parte de esto se debe a lo logradas que están las interacciones y las animaciones de los personajes.

El segundo se llama VR-Chat, es un juego que se puede categorizar como “Hub Social Virtual”. Es como una fase temprana de lo que se conoce como el Metaverso. En este juego se interactúa con otras personas y cada persona puede crear su propio avatar e importarlo al juego. El juego tiene un motor de creación de avatares, pero también te permite importar los tuyos propios. Estos avatares que se importan tienen sus propias animaciones que son creadas mediante controles digitales (Rigs), de aquí nace la idea inicial de utilizar Rigging para el proyecto. Más tarde al descubrir los Blendshapes también descubrí que estos están implementados en el juego y se usan sobre todo para las animaciones faciales.

4. Diseño y desarrollo de la aplicación

4.1. Ideas iniciales de desarrollo

Cuando se habló por primera vez acerca del proyecto y se investigó sobre animaciones faciales, la idea inicial que se tuvo fue la de grabar un vídeo pronunciando todas las letras del abecedario y replicar dicho vídeo en nuestros modelos 3D. Una vez conseguido este paso, lo siguiente sería desarrollar un programa que fuese capaz de analizar el sonido, categorizando así cada uno de los fonemas por separado y asociándolos con las distintas animaciones de nuestro personaje.

La idea inicial fue acompañada de una búsqueda de distintos posibles Softwares para ayudar todo este proceso. Después de una exhaustiva búsqueda nos quedamos con dos posibles opciones, SAPI Lipsync by Annosoft [18] y Oculus Lipsync by Meta Quest. En un primer momento, interesó más el Software de Annosoft pero rápidamente en el transcurso del proyecto se cambió a Oculus Lipsync por distintos motivos, siendo los principales su simple integración con Unity y su sencillez.

4.2. Diseño de los personajes

Los personajes usados en la aplicación fueron creados en el marco del proyecto, por una investigadora de la Facultad de Bellas Artes y un desarrollador. Estos tienen ropas y animaciones distintas cada uno. Los personajes tenían animaciones propias, pero ninguna de ellas era facial, todas de movimiento de tronco hacia abajo. Es por eso que se decidió crear una primera aproximación con un Rig para la cara.

Figura 1: Modelos Caballero hablando y mujer hablando respectivamente

4.2.1. Rigging

La primera idea para llevar a cabo la tarea de las animaciones faciales fue le uso de un esqueleto (Rig), mediante la herramienta de Unity Rigify [19]. Un rig consiste en un sistema de control de animaciones que se utiliza en el campo de la animación en 3D [Figura 2]. El proceso implica asignar y conectar huesos virtuales en un modelo de manera similar a, en nuestro caso, el cuerpo humano. Este proceso permite animar personajes de menara eficiente y realista, generando una jerarquía de objetos para las que la animación de uno afecta a las animaciones de sus hijos.

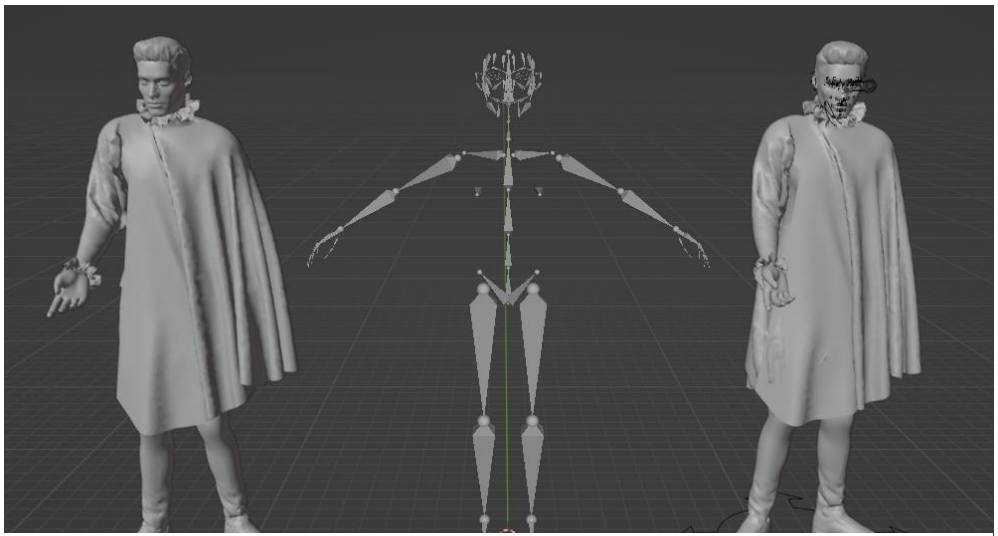

Figura 2: Las tres fases del proceso de creación del esqueleto de un modelo 3D (de izquierda a derecha), el modelo limpio, el esqueleto de humanoide y el modelo con el esqueleto aplicado.

El proceso de implementación del rigging comenzó con la importación de nuestro modelo 3D en Blender. En esta etapa, se aprovechó la capacidad de Blender para importar esqueletos humanos predefinidos, los cuales pudieron ser ajustados y adaptados al modelo en cuestión. Este paso consistió en realizar ajustes precisos para asegurar que cada hueso se alineara adecuadamente con el modelo deseado, requiriendo un trabajo manual detallado, moviendo y posicionando cada uno de los huesos de manera individual. Esto nos brindó una primera base sólida con la que trabajar.

Con todos los huesos correctamente ajustados, procedimos a aplicar pesos automáticos a cada uno de ellos. Este proceso consiste en asociar los vértices a los diferentes huesos, de acuerdo con su proximidad. De esta manera, al animar cada hueso por separado, los vértices se moverán de manera fluida y natural, siguiendo las acciones del rig.

Ya una vez aplicado el rig a nuestro personaje es posible mover, rotar o incluso agrandar nuestro modelo en 3D a nuestro antojo para crear estas animaciones, en la Figura 3 podemos observar a la derecha como quedaría el modelo una vez tiene los pesos de los huesos aplicados. Cada cubo pertenece a un vértice de uno de los huesos, el cual se puede mover con total libertad aplicando este movimiento al modelo.

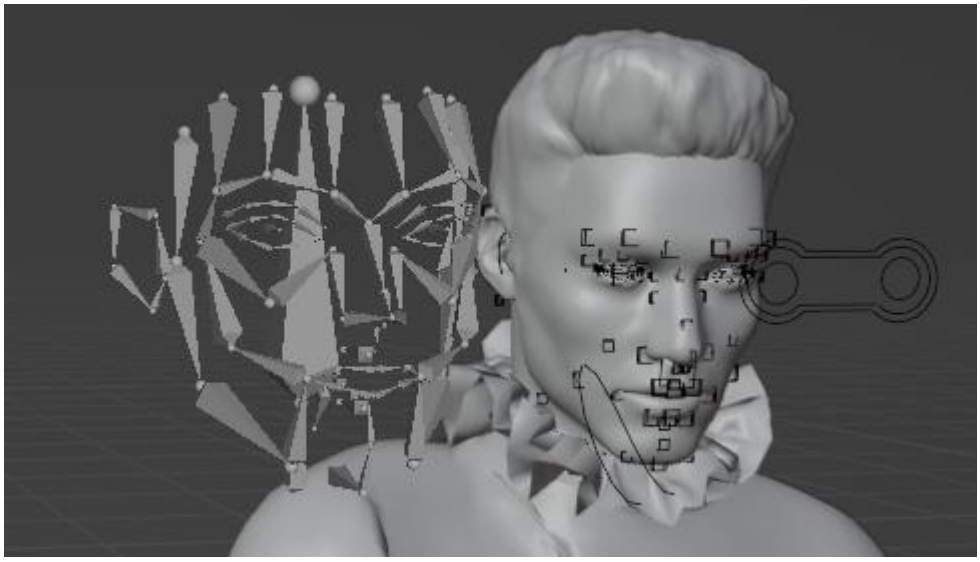

Figura 3: Los huesos de la cara del esqueleto (izquierda) y el modelo con los huesos y pesos aplicados (derecha)

4.2.2. Problemas con el Rig y Oculus Lipsync

Mientras se desarrollaba el Rig de los modelos como primera aproximación de solución para las animaciones, se decide finalmente contar con la tecnología Oculus Lipsync para nuestro proyecto. Cuando se comienza a investigar más a fondo el uso de esta tecnología se

descubre el concepto de los BlendShapes y se descarta el uso de los rigs pues no nos van a ser útiles con el Lipsync.

4.2.3. Blendshapes

En animación 3D, se conoce como Blendshape, realizar una deformación de un modelo 3D base, pero conservando la topología original (el número de vértices), y luego vinculando las deformaciones. La figura deformada es conocida como Shape, de ahí el nombre de Blendshape. Debido a que la figura base y el shape son iguales en topología, la deformación puede ocurrir de manera sutil, pasando del mínimo al máximo poco a poco.

Los Blendshapes son comúnmente utilizados para crear movimientos y expresiones faciales realistas. Por ejemplo, un modelo 3D de un personaje puede tener diferentes Blendshapes para simular sonrisa, fruncir el ceño o abrir o cerrar los ojos [Figura 4].

Figura 4. De izquierda a derecha, Caballero con Blendshapes inactivos, Caballero con Blendshape "fruncir ceño" acivo y Caballero con Blendhsape "pestañear" activo

Para crear un Blendshape en un modelo de Blender se siguen los siguientes pasos:

1. Se selecciona la malla (Mesh) a la que se desea realizar cambios.
2. En el menú de opciones seleccionar la pestaña *Propiedad de Datos del Objeto*.
3. En esta pestaña se encuentran dos grandes campos: *Grupos de Vértices y Shape Keys*.
4. Al lado de cada campo, existen varios botones. Se utiliza el bóton + para crear un nuevo Blendshape en el campo de *Shape Keys*.
5. Una vez creado seleccionar el nuevo Blendshape y aumentar el campo *Valor (Value)* de 0 a 1, esto hará que se visualicen los cambios en el modelo.

6. Deformar modelo según las necesidades mientras este BlendShape está seleccionado.

4.3. Animaciones de los personajes

Las animaciones de los personajes han supuesto un gran reto por distintos motivos. Dividiremos en este trabajo las animaciones faciales de las del cuerpo entero tal y como se ha hecho en el proyecto.

4.3.1. Animaciones corporales

Como la meta del proyecto era crear animaciones faciales para los personajes, las animaciones corporales se han abordado al final. No obstante, se describirán antes, pues en el proceso final es necesario generarlas en primer lugar para posteriormente crear Blendshapes para la cara sobre ese modelo con las animaciones aplicadas.

El principal reto de las animaciones del cuerpo viene dado por las animaciones de la ropa de los modelos. Estas fueron desarrolladas por un miembro del equipo de proyecto y se llevaron a cabo mediante el Software especializado Marvelous Designer [20] [21]. Esto supuso un reto adicional pues los personajes no parten de una base inicial, sino que ya tienen varias capas de complejidad por las cuales el desarrollo de este trabajo se ha ido viendo afectado, ya que no es lo mismo empezar desde cero e ir construyendo poco a poco que tener que construir sobre una base invariable, es limitante a la hora de hacer según qué cambios.

La primera vez que se unieron las animaciones faciales con las corporales, se hizo únicamente con el modelo sin la ropa animada, es decir, el cuerpo se movía, pero la ropa estaba estática. Esto no dio muchos problemas pues como se trata de agregar animaciones faciales a personajes cuya cara no está animada el trabajo realizado no supone tocar nada de lo realizado anteriormente en los modelos.

Los problemas comienzan al querer compaginarlo con la ropa animada. Las animaciones de la ropa junto con el cuerpo vienen descritas en un archivo *.mdd*. Este formato de archivo se utiliza para almacenar animaciones de modelos en 3D, las siglas provienen de "*Motion Description Data*". Estos archivos contienen información sobre la posición de los vértices de un modelo en cada fotograma de una animación y esta información es usada por Blender para reproducirla de forma precisa. Para importar estos modelos con su respectivo fichero *.mdd* basta con agregarle una componente *meshCache*, el uso principal este modificador

es que los datos de una malla animada se apliquen y se reproduzcan. Para agregar el componente basta con dirigirse a las propiedades de la malla en Blender, a la pestaña *Propiedades del modificador (Modifier Properties)*, añadirla e incluir el dichero *.mdd.*

De esta manera ya estarían incluidas las animaciones de la ropa, el problema viene a la hora de exportar el modelo para Unity. Hay que exportar en un formato válido, que no solo exporte la malla (Mesh) del modelo, sino también sus materiales, texturas y animaciones. Es por esto que se elige la opción del formato *.fbx*, este se utiliza para intercambiar modelos, animaciones y otros objetos 3D entre distintos programas. Cuando se exporta un modelo con este formato, este encapsula varias características de este como su geometría, sus materiales y texturas, sus animaciones o las cámaras y luces.

Una vez el modelo está exportado en Blender e importado en Unity se comprueba que a la exportación no se le han aplicado bien las animaciones de la ropa, por este motivo, es necesario buscar otra manera de realizarla.

La manera en la que se soluciona es haciendo un paso intermedio para que no haya dos animaciones por separado (cuerpo y ropa) a la hora de exportar el .fbx sino una. Para ello se realizan los mismos pasos que antes hasta el punto de aplicar el fichero .mdd al modelo. En este momento se exporta el modelo en formato Wavefront .obj seleccionando la opción de *export animation*. Al tratarse de un .obj, este no almacena tanta información como el .fbx, sino únicamente su geometría. La manera en la que se exporta la animación es creando un objeto por cada frame que tiene la animación [Figura 5].

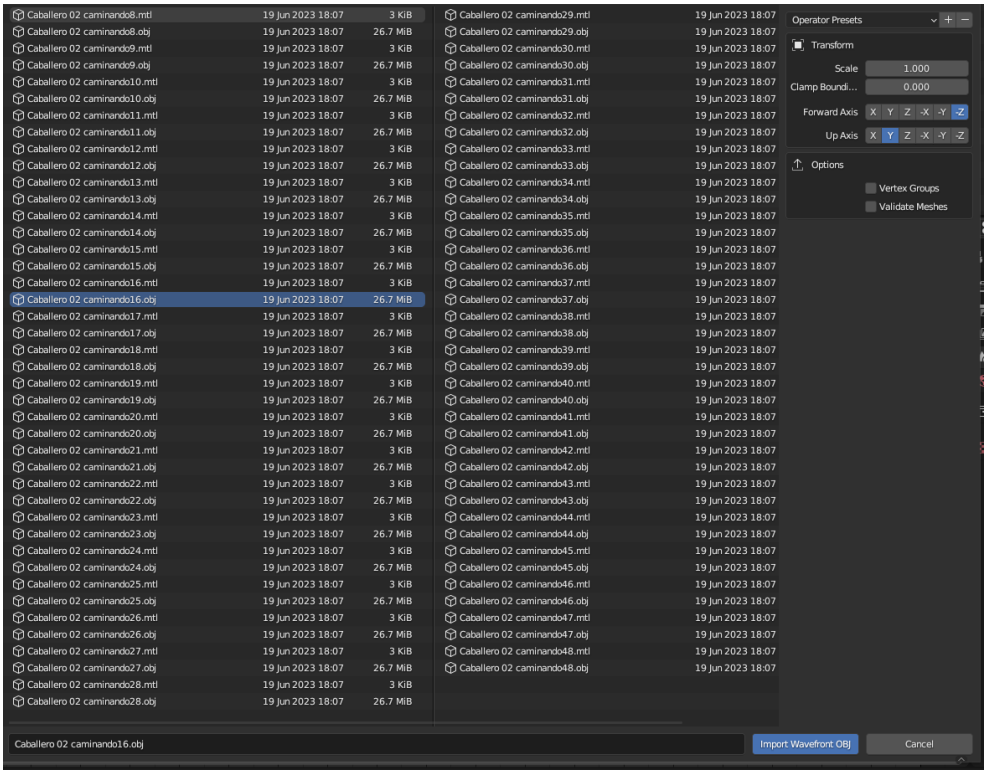

Figura 5. Objetos creados después de exportar una animación como .obj junto con sus correspondientes ficheros que almacenan la información de los materiales

Una vez se tienen todos los objetos se reimportan en un nuevo proyecto de Blender. Como vemos en la [Figura 6] al importar los objetos al Blender estos tienen el cuerpo y la ropa alineados en cada uno de los objetos. Una vez importados los objetos se seleccionan todos con la tecla *ctrl* a excepción del primer objeto (el que representa el frame 1 de la animación) que se selecciona el último. Una vez seleccionados se navega a la pestaña *Propiedad de Datos del Objeto* y en la opción *Shape Keys* se selecciona *Join as Shapes*. Con esto se consigue unir todos los objetos en el primero de ellos como Blendshapes del mismo.

Figura 6. Caballero Caminando tras importar los distintos .obj al Blender

Ahora se ha de recrear la animación que tenía el objeto anteriormente, pero usando los Blendshapes. Al unir en el paso anterior todos los objetos como Blendshapes del primero procedemos a la eliminación de todos ellos ya que ahora tenemos contenidas todas las mallas en un mismo Objeto. Como se ha comentado anteriormente un Blendshape puede tomar valores decimales desde cero hasta uno, un cero implicaría que el modelo se encuentra en la animación inicial, y un uno en el primer Blendshape implicaría que el objeto se encuentra en la animación correspondiente al frame concreto del que se ha cogido el objeto importado.

Figura 7. A la izquierda Caballero con todos sus Blendshapes a 0. A la izquierda con el primer Blendshape a 1

Para recrear la animación hay que encajar de nuevo en cada frame la animación que le correspondía, dándole el valor de uno a su Blendshape y cero al anterior y al siguiente, de modo que se cree una suave transición entre los Blendshapes [Figura 7]. Una vez hecho esto el modelo se encuentra animado y listo para ser exportado como *.fbx*, esta vez con ambas animaciones en una misma [Figura 8].

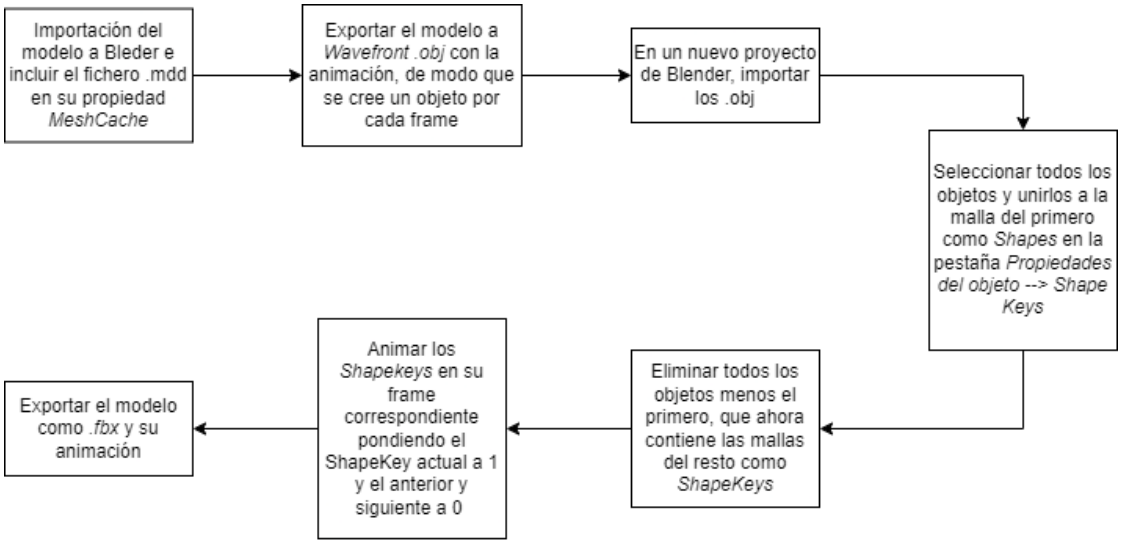

Figura 8. Esquema de animacion corporal de los modelos

4.3.2. Animaciones faciales

Las animaciones faciales son el primer elemento que se abordó del proyecto pues ese era el objetivo principal del trabajo, no obstante, se incluyen tras las corporales pues en el

proceso final de animación es necesario hacer este paso tras haber realizado las animaciones de cuerpo y ropa a los Blendshapes.

Como se ha descrito en el apartado 3, en un principio, se intentó una aproximación de la solución con un Rig, pero fue rápidamente descartada cuando finalmente se decidió usar Oculus Lipsync y cuando se descubrieron los Blendshapes, que son muy utilizados en el campo de las animaciones faciales.

Oculus Lipsync utiliza un repertorio de 15 visemas (sil, aa, CH, DD, E, FF, ih, kk, nn, oh, ou, PP, RR, SS y TH) asociados a fonemas diferentes que se asignan a los distintos objetivos de transformación geométrica del avatar (los blendshapes), además de otros 10 relacionados con los ojos, las cejas, y acciones como mirar hacia arriba o reírse (brow, eyesClosed, innerBrowRaiser, jawOpen1, lookDown, lookLeft, lookRight, lookUp, outerBrowRaiser y laughter). Para que posteriormente la librería reconozca dichas acciones y visemas es necesario implementarlas en los modelos antes de exportarlos en Blender, esto se consigue mediante Blendshapes.

Para ello se crean 25 nuevos Blendshapes. Es importante que estos no interfieran con los Blendshapes de la animación corporal. Esto se consigue creándolos en un momento en el que la animación ya no esté transcurriendo. Por ejemplo, si la animación va desde el frame 0 hasta el frame 60, se crean estos nuevos Blendshapes en el frame 70, de modo que no se solapen en la línea de tiempo.

Para hacer las animaciones lo más precisas posibles, grabe un vídeo de mí mismo pronunciando cada uno de los fonemas y exagerando muchos las facciones de la cara para así poder recrearlas. El modelo cuenta con un gran número de vértices (162.000 aproximadamente) así que se puede editar el modelo con mucho nivel de detalle, aunque esto supone un reto ya

que el modelo cuenta con distintas mallas para la cara, los ojos, el interior de la boca, los dientes (arriba y abajo por separado), la lengua y los labios [Figura 9].

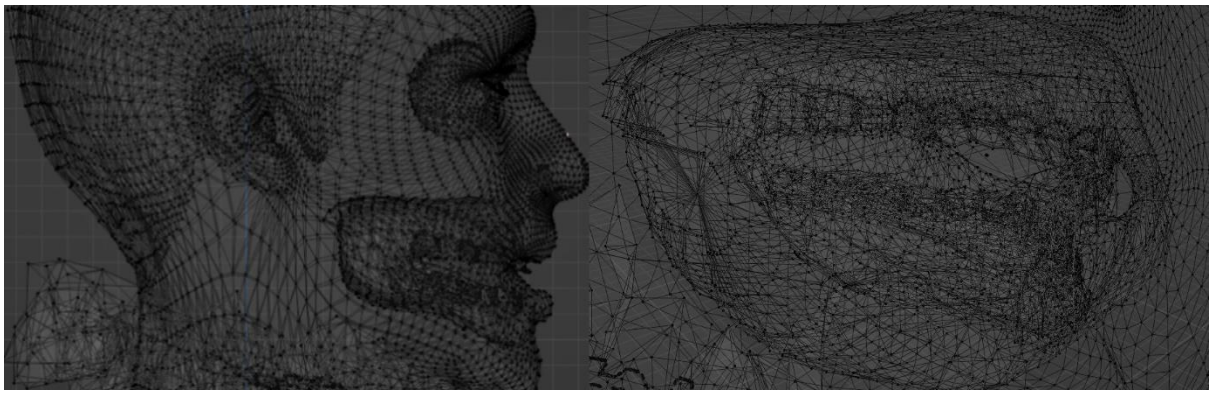

Figura 9. A la izquierda un plano general de los vertices de la cara, los ojos y la boca. A la derecha, los vertices de la boca en detalle, se aprecian los dientes, la lengua, y el interior de la boca

El proceso [Figura 10] para crear cada una de estos Blendshapes es el mismo:

- Se crea el Blendshape deseado con su correspondiente nombre
- Se coloca el valor del Blendshape a 1 para ver los cambios mientras se edita la malla.
- Se entra en *Edit Mode* para deformar el modelo, y se van seleccionando vértices manualmente para moverlos y colocarlos de modo que creemos la expresión deseada.

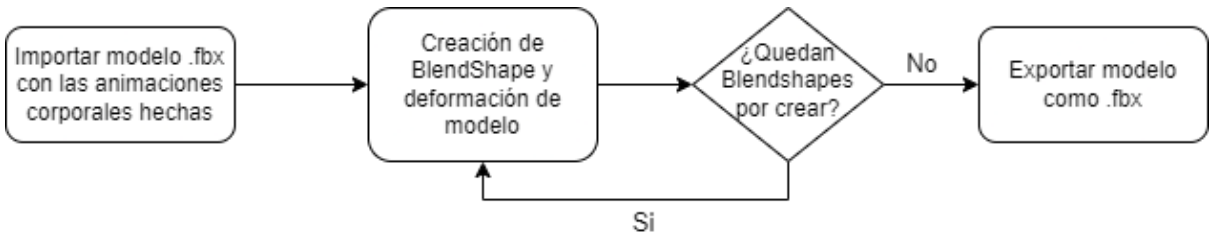

Figura 10. Diagrama de flujo de creación de Blendshapes en el modelo

Para el último punto del proceso de creación de un Blendshape, a la hora de mover y colocar Blendshapes existe una opción muy útil en Blender y que sirve de gran ayuda. Se trata de la opción *Edición Proporcionada (Proportional Editing)*, esta opción nos permite que a la hora de mover uno o más vértices, los demás sigan dicho movimiento creando una deformación suave y conjunta.

Una característica especial que tienen estos nuevos Blendshapes es que, a diferencia de los del cuerpo, a estos nos se les da ningún valor a lo largo de los frames, están siempre a 0. Lo

valores de estos Blendshapes se verán modificados por Oculus Lipsync cuando el modelo esté en la aplicación de Unity.

4.3.3. Migración de Blendshapes a los distintos modelos

Existen muchos modelos distintos en la aplicación, así que, para no estar uno por uno editando y deformando cada uno de ellos se ha pensado una manera de pasar los Blendshapes de un modelo a otro.

Varios personajes tienen la misma cara, lo único que cambia entre ellos es el pelo y la ropa con la que están vestidos. La solución que se ha llevado a cabo consiste en elegir alguno de los modelos que tienen los Blendshapes creados e intercambiar su cabeza por la del modelo al que se desea aplicar los mismos. Para el ejemplo que se mostrará a continuación se usarán los modelos de la Figura 1

Aunque a priori pueda parecer una tarea simple, la complejidad surge tras entender que cada modelo tiene aplicada una propia animación corporal. No es tan sencillo como cortar la cabeza de uno y pegársela al otro, hay que hacer un paso intermedio aprovechando los huesos del cuerpo que provienen del Rig creado por Mixamo.

El primer paso es cortarle la cabeza al modelo en el que tenemos nuestros Blendshapes creados, en este caso será el de la mujer, y al modelo que le reemplazaremos la cabeza, el del hombre [**Error! Reference source not found.**]. Una vez hecho esto tendríamos que situar la cabeza acorde con donde tendría que ir en el otro cuerpo. Ahora hay que hacer un paso intermedio para que a la nueva cabeza se le aplique la animación del cuerpo, porque si solo se unen ambas mallas el cuerpo se moverá pero la cabeza estará estática. Para realizar este paso, utilizando Blender hay que:

1. Seleccionar el cuerpo, la cabeza y hacer un join entre ellas (*ctrl+j*), esto las combinará dentro de la misma capa.
2. Una vez hecho esto se va a *Edit Mode* de la malla y se seleccionan manualmente todos los vértices de la nueva cabeza.
3. Con todos los vértices seleccionados hay que ir al apartado de *Vertex Group* en donde se encuentran todos los huesos del Mixamo.
4. Por último, se busca el hueso principal de la cabeza y se le asigna a los vértices que habíamos seleccionado previamente con peso uno.

Una vez hecho esto la animación del cuerpo también se le está aplicando a la cabeza. Ahora solo queda unir ambas partes. Para ello, aun en el *Edit Mode* hay que seleccionar el corte del cuello del cuerpo y el corte del cuello de la cabeza. Una vez seleccionados se usará la opción llamada *Bridge Edge Loops* que se encuentra en la pestaña *Edge* [Figura 11]. Esto nos unirá ambas selecciones creando nuevas caras entre los vértices.

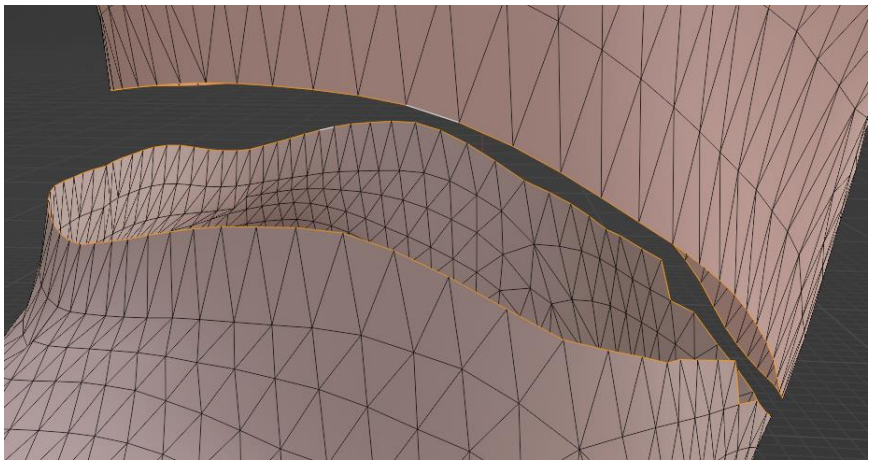

Figura 11. Cabeza y cuello con los bordes seleccionados antes de ser unidas

Ahora el modelo estaría unido y queda darle los últimos ajustes cambiando el pelo de los modelos [Figura 12]. Una vez hecho esto el modelo está listo para exportar[Figura 13].

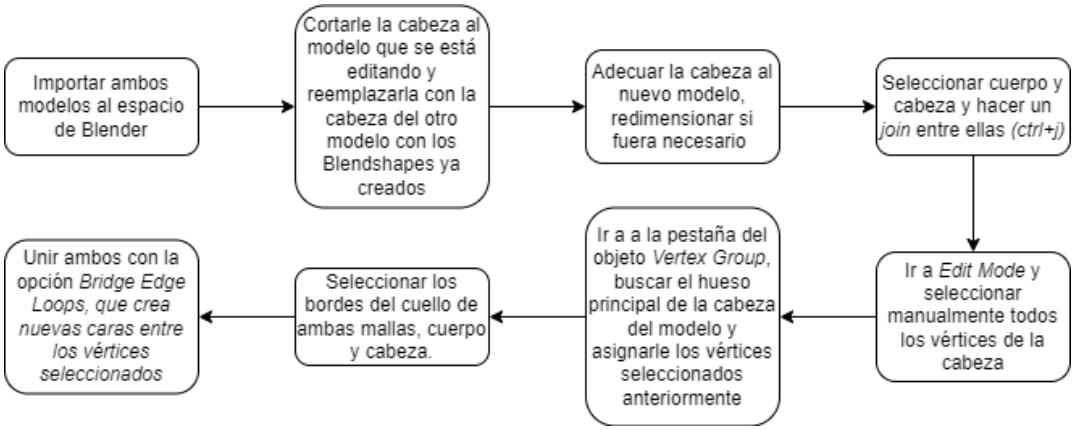

Figura 12. Esquema de sustitución de cabeza en los modelos

Figura 13. De izquierda hacia abajo. Los modelos antes del cambio de cabezas. Los modelos con el cuerpo preparado para colocarle la cabeza. A la derecha el modelo con ambos pelos y el Blendshape de abrir la boca activo.

4.4. Implementación de los personajes en Unity

En este apartado se describe la implementación de nuestros personajes en Unity, su integración con las librerías de Oculus Lipsync y la puesta a punto del escenario y la aplicación. Se divide por subapartados comenzando por este último.

4.4.1. Escenario

Lo primero en llevar a cabo al abrir un nuevo proyecto de Unity es la importación del escenario en el que se va a trabajar. Se trata de un escenario ya desarrollado para anteriores aplicaciones que se remonta al San Cristóbal de La Laguna del siglo XVI. Podemos observar como la estética del escenario va completamente acorde con la estética de los personajes que se han animado [Figura 14].

Figura 14. Vista aérea del escenario de San Cristóbal de La Laguna

El escenario es un objeto padre del que cuelgan muchos hijos, donde cada uno de ellos conforma un elemento de este. Para hacer el escenario navegable por los personajes lo primero es darle físicas. La primera aproximación para hacer esto fue agregar un componente *Mesh Collider* al objeto padre Escenario, pero debido a que está formado por muchos objetos hijos esta aproximación no funcionó. Finalmente, lo que se decidió es agregar un componente *Mesh Collider* a cada uno de los hijos a la hora de ejecutar la aplicación. Esto se hace mediante un script que se le aplica al escenario. El funcionamiento del script [Figura 15] es sencillo:

- Se almacenan todos los objetos del padre en un Array.
- Para cada uno de ellos se busca su componente *Mesh Filter*
- Se crea un componente *Mesh Collider* y se le aplica al hijo

```
using UnityEngine;

public class MeshColliderGenerator : MonoBehaviour
{
    void Start()
    {
        GenerateMeshColliders();
    }

    void GenerateMeshColliders()
    {
        // Se obtienen todos los hijos del objeto padre
        Transform[] children = GetComponentsInChildren<Transform>();

        // Recorre cada hijo y agrega un Mesh Collider si tiene un Mesh Filter
        foreach (Transform child in children)
        {
            if (child != transform) // Evita agregar colisionadores al objeto padre
            {
                MeshFilter meshFilter = child.GetComponent<MeshFilter>();
                if (meshFilter != null)
                {
                    // Agrega un Mesh Collider al hijo
                    MeshCollider meshCollider = child.gameObject.AddComponent<MeshCollider>();
                    meshCollider.sharedMesh = meshFilter.sharedMesh;
                }
            }
        }
    }
}
```

Figura 15. Algoritmo que genera los meshes para cada uno de los hijos del escenario

Una posible mejora de este script consistiría en generar los *Mesh Colliders* de los objetos hijos dinámicamente según la distancia a la que se encuentre el personaje principal, de este modo se ahorraría algo de tiempo de cómputo.

4.4.2. Personajes y Oculus Lipsync

Una vez importado y configurado el escenario de manera correcta lo siguiente es importar algún personaje de los animados previamente. Para ello se selecciona alguno de los *.fbx* exportados del paso anterior con los Blendshapes de la animación del cuerpo y los Blendshapes de la cara.

Lo primero de todo al importar el modelo a nuestro proyecto es dotarlo de físicas para que se comporte de manera coherente con el entorno. Se le agregan dos nuevas componentes, un *Rigidbody* para que se le aplique gravedad y un *Capsule Collider* para que colisione con nuestro *Mesh Collider* del suelo de modo que se mantenga de pie.

También hay que asegurarse de que los frames de la animación del modelo vayan acordes con los animados en Blender, de modo que si en Blender la animación empieza en el frame 2 y acaba en el 62, en Unity estén como principio y final esos mismo frames. Una vez

comprobado se arrastra la animación al modelo y se le dota de inteligencia en el animator de Unity [Figura 16].

Figura 16. Caballero caminando en mitad de su animación

Ahora es el momento de poner a prueba los blendshapes creados para el modelo. Lo primero es descargar e importar al proyecto el paquete de Oculus Lipsync de su web [15]. Con el paquete vienen distintos elementos, para el proyecto nos interesan únicamente algunos de los scripts pero también vienen modelos, audios, escenas y prefabs.

De los scripts disponibles en la librería se añaden únicamente tres al Objeto, uno que procesa el audio, otro que lo aplica a los blendshapes del modelo y otro en caso de que en lugar de coger el audio de un archivo de audio se quiera procesar el audio del micrófono a tiempo real. Pero realmente se usan más de tres ya que estos scripts son clases que heredan de otros de los scripts incluidos en la librería.

El script de Audio (*OVRLipSyncContext.cs*) que se usa es una clase heredada de uno de los otros Scripts (OV*RLipSyncContextBase.cs*). Al iniciar la aplicación, en la clase padre se declaran distintos atributos importantes para el posterior funcionamiento como el objeto *AudioSource*, que tomará el valor del Audio que se le pase en el editor de propiedades, o en caso de que no se le pase ninguno, el valor de un *AudioSource* asociado al objeto. Como podemos ver en la Figura 17, esta sería la jerarquía de los objetos en la aplicación y los scripts y el *AudioSource* se aplicarían al objeto inputMic. También en este script del procesado de audio se crea un objeto de la clase *Frame* (clase presente en otro de los scripts de la librería, *OVRLipSync.cs)* que representa el frame actual con los resultados del fonema y una variable de tipo *uint* que representa un contexto de lipsync.

El script de procesado de audio tiene implementado distintos algoritmos, centraremos la explicación en los más importantes ya que muchos de ellos son para depurar las o testear las animaciones.

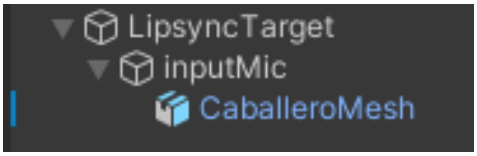

Figura 17. Jerarquía de Objetos para el procesado de Audio y las animaciones en Unity

En el Script de Audio (*OVRLipSyncContext.cs*) se declara una función llamada *OnAudioFilterRead*. Esta función implementada en un Script de Unity, es llamada automáticamente cuando un *AudioSource* en el mismo *GameObject* intenta reproducir un sonido. Una vez se inicia esta función llama a otra que empieza los algoritmos de procesado de audio, se realiza un preprocesado, un procesado y un postprocesado, antes de ser enviado al script de transformación del objetivo [Figura 18].

```
/// <summary>
/// Process F32 audio sample and pass it to the lip sync module for computation
/// </summary>
/// <param name="data">Data.</param>
/// <param name="channels">Channels.</param>
public void ProcessAudioSamples(float[] data, int channels)
{
    // Do not process if we are not initialized, or if there is no
    // audio source attached to game object
    if ((OVRLipSync.IsInitialized() != OVRLipSync.Result.Success) || audioSource == null)
    {
        return;
    }
    PreprocessAudioSamples(data, channels);
    ProcessAudioSamplesRaw(data, channels);
    PostprocessAudioSamples(data, channels);
}

/// <summary>
/// Raises the audio filter read event.
/// </summary>
/// <param name="data">Data.</param>
/// <param name="channels">Channels.</param>
void OnAudioFilterRead(float[] data, int channels)
{
    if (!skipAudioSource)
    {
        ProcessAudioSamples(data, channels);
    }
}
```

Figura 18. Funciones OnAudioFilterRead y de procesado de audio

De los tres métodos de procesado de audio el más importante es el *ProcessAudioSamplesRaw()*, el *PreProcessAudioSamples()* simplemente aumenta la ganancia del audio, y el método de *PostprocessAudioSamples()* evita que, en caso de tener activada la opción de escuchar el audio que se está reproduciendo, este no retroalimente los altavoces a través del micrófono.

El *ProcessAudioSampleRaw()* recibe el audio actual (formato *float[]*) y lo envía al motor de Lipsync llamando a la función con los datos del audio, el contexto de lipsync y el frame actual del visema. El audio es procesado por el motor del lipsync y retorna un resultado que queda almacenado en el objeto de la clase.

Lo siguiente es el script de control de los Blendshapes (*OVRLipSyncContextMorphTarget*). Cuando se inicia la aplicación se declaran distintas variables como el *SkinnedMeshRender,* que es nuestro modelo y se lo especificamos en el editor de propiedades de Unity. Una vez se ha asociado el modelo hay que compaginar bien los Blendshapes con la lógica interna de visemas de Lipsync. Internamente existe un objeto *Ennumerable* que hace referencia a cada uno de los visemas asociados a los fonemas, hay que informar dicha variable con la posición que toma cada uno de nuestros Blendshapes en el modelo para que cuando se detecte el fonema "E" se active correctamente el Blendshape asociado a "E" [Figura 19].

Figura 19. De izquierda a derecha. Ennumerable que se asocia a cada Visema. Propiedad de Unity en la que mapeamos cada visema a un blendshape: Los Blendshapes del modelo

Al lanzar la aplicación también se crea una instancia de la clase *lipsyncContext* asociada al Objeto, esto permite en el *Update()* llamar al *getter* del Frame actual de *lipsyncContext*, el cual ya tiene el resultado procesado por el motor de Lipsync en la clase Audio para aplicar dicho resultado a los Blendshapes [Figura 20].

El otro Script que se le puede aplicar al Modelo es el *OVRLipSyncMicInput* que recogería la entrada del Microfono y esto sería lo que recogería el *Context* como el *AudioSource*.

```
/// <summary>
/// Update this instance.
/// </summary>
void Update ()
{
    if((lipsyncContext != null) && (skinnedMeshRenderer != null))
    {
        // get the current viseme frame
        OVRLipSync.Frame frame = lipsyncContext.
GetCurrentPhonemeFrame();
        if (frame != null)
        {
            SetVisemeToMorphTarget(frame);

            SetLaughterToMorphTarget(frame);
        }

        // TEST visemes by capturing key inputs and sending a signal
        CheckForKeys();

        // Update smoothing value
        if (smoothAmount != lipsyncContext.Smoothing)
        {
            lipsyncContext.Smoothing = smoothAmount;
        }
    }
}
```

```
/// <summary>
/// Sets the viseme to morph target.
/// </summary>
void SetVisemeToMorphTarget(OVRLipSync.Frame frame)
{
    for (int i = 0; i < visemeToBlendTargets.Length; i++)
    {
        if (visemeToBlendTargets[i] != -1)
        {

// Viseme blend weights are in range of 0->1.0, we need to make range
100
            skinnedMeshRenderer.SetBlendShapeWeight(
                visemeToBlendTargets[i],
                frame.Visemes[i] * 100.0f);
        }
    }
}
```

Figura 20. Código que aplica el resultado del audio procesado por el motor de Lipsync al modelo

Como vemos en la Figura 19, a la derecha podemos ver los Blendshapes del modelo. Los primeros antes del llamado *sil* (Blendshape que representa la posición base de la cara), pertenecen a la animación corporal y estos se activan automáticamente en el momento que se le aplica la animación en el *Animator* al modelo, ya que esta animación fue creada y exportada desde Blender.

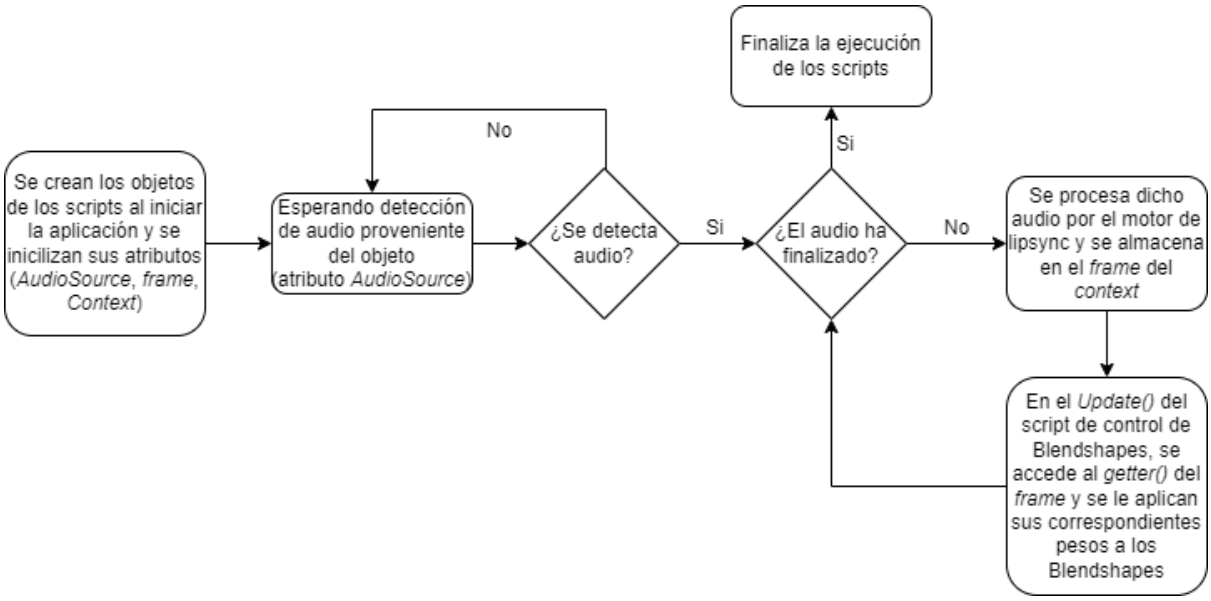

Figura 21. En este esquema podemos visualizar el flujo que siguen los scripts de Lipsync en la ejecución de la aplicación

Una vez realizado todo esto el modelo estaría funcionando completamente [Figura 22].

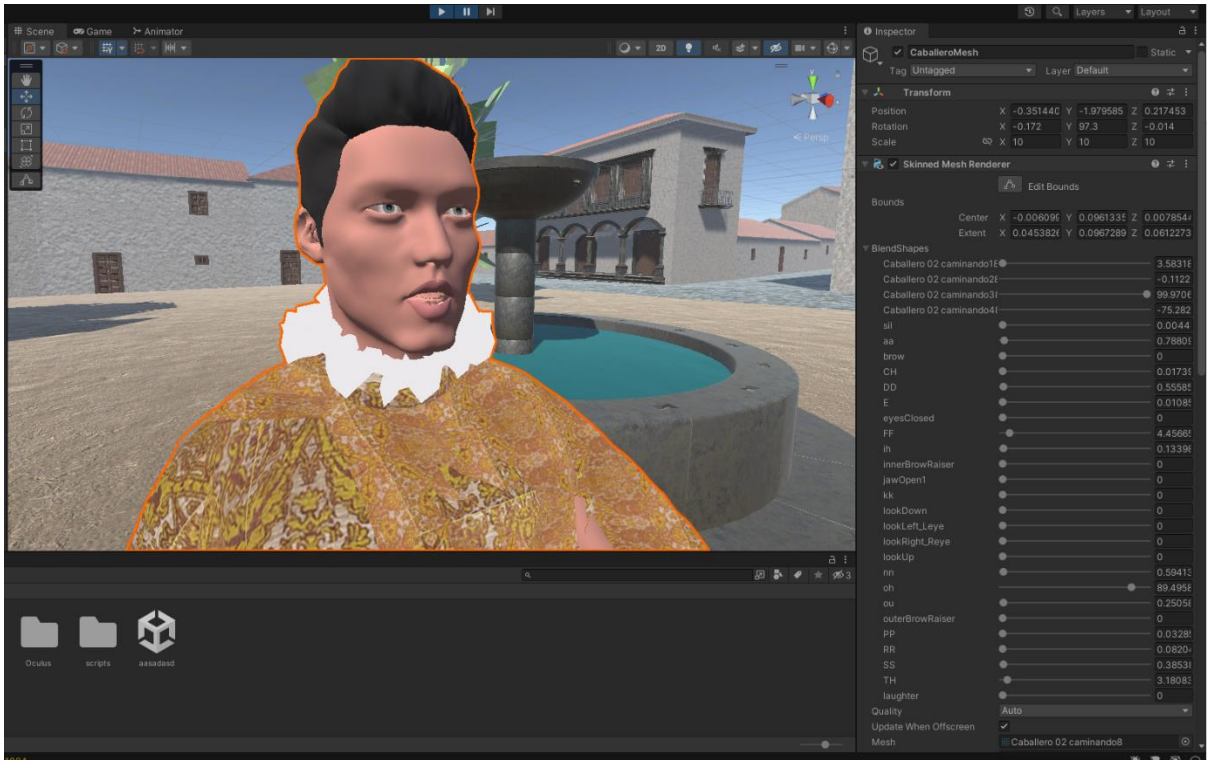

Figura 22. En esta imagen se aprecia como el modelo está en mitad de la animación de caminar mientras habla, el Blendshape de "oh" está activado al 89% y la boca tiene forma de "O"

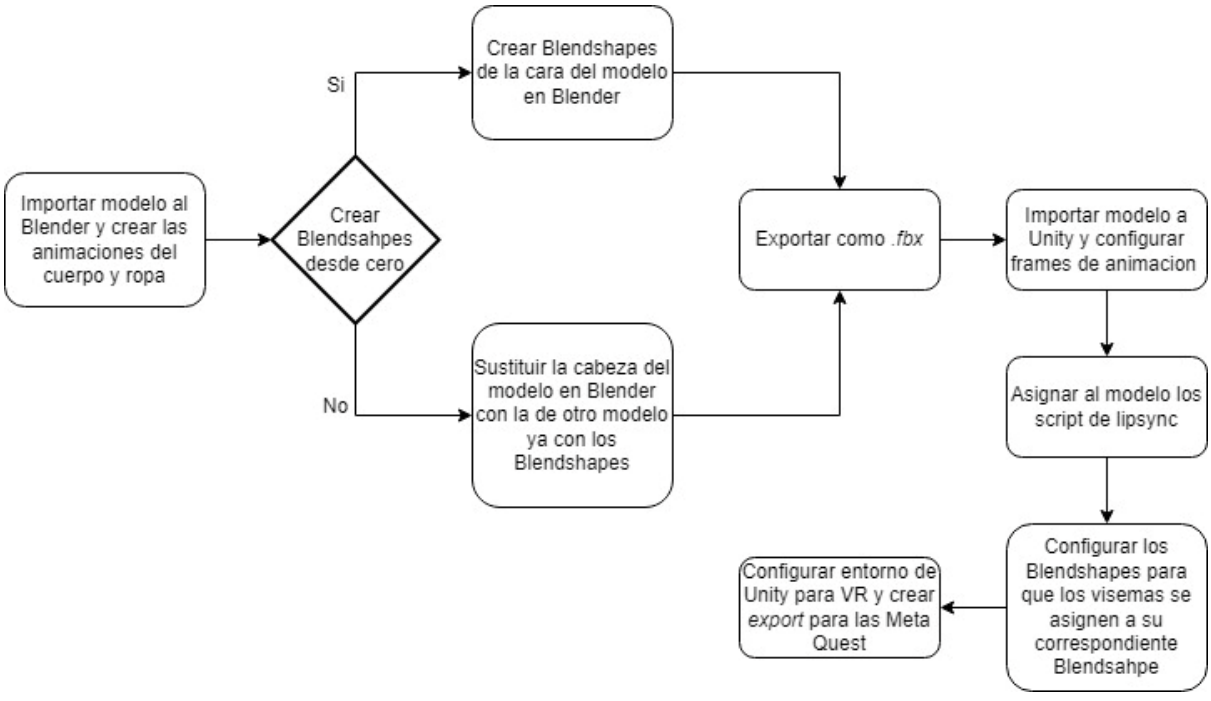

Figura 23. Esquema General del flujo de trabajo final seguido para la creación e implementación de modelos

4.5. Integración con dispositivo Oculus

Una vez la aplicación está finalizada, queda adecuarla para el entorno de realidad virtual. Se usarán las Meta Quest, lo primero es configurar las gafas de modo que se permita ejecutar aplicaciones "desconocidas". Para ello, en el menú de la aplicación de Oculus, es necesario activar la opción de "Fuentes desconocidas" y el modo desarrollador.

Lo primero que debemos hacer en Unity es descargar el paquete con el Plugin que contiene todas las funcionalidades de realidad virtual. Este paquete se llama OpenXR, es desarrollado por Khronos y de código abierto, busca simplificar el desarrollo de VR y de AR dotando a los desarrolladores de distintas herramientas y permitiendo un exportado más sencillo a varios dispositivos de VR. Para instalarlo se abre el Project Manager y se busca el paquete llamado OpenXR Plugin.

Una vez se tiene el plugin se instala también, desde el Project manager, un paquete llamado XR Interaction Toolkit. Este cuenta con varias acciones y scripts ya programados para la locomoción y el movimiento del jugador.

Ya con todo instalado lo primero que se hace es añadir al proyecto los presets predeterminados de entrada que vienen con la librería XR Interaction Toolkit. Para ello hay que irse al directorio dentro del proyecto en el que se haya ubicado por defecto la instalación del paquete y en cada uno de los predeterminados de entrada hacer click en el botón de *Add to ActionBasedControll*. Esto configurará el proyecto automáticamente con los controles correctos para la locomoción del jugador, activar la teletransportación (método que se usa para mover el jugador en aplicaciones de VR), detección y rotación correcta de los mandos, etc.

Lo siguiente una vez se han configurado todos los controles es añadir el objeto que representa al jugador principal. Para ello hay que hacer click derecho en la jerarquía de objetos de Unity y dentro de la categoría *XR* añadir un objeto *XR rig*.

Finalmente hay que correr la aplicación en las Meta Quest, para ello hay dos maneras, o bien con el *Quest Link* que consiste en conectar las gafas con el ordenador por cable o por la red Wifi, o bien creando una build de la escena para Oculus Quest 2.

Aquí podemos ver una video demonstración de la aplicación ejecutándose en las gafas de realidad virtual: https://youtu.be/N1Og4Ew1qSc

5. Conclusions and Future lines of work

5.1. Conclusions

By way of concluding this project, we can emphasize that facial animations are an integral part of any animated character that is given a voice and is expected to convey a message or knowledge. By synchronizing facial expressions with real-time processed audio, we have created a more authentic and immersive experience. Users can now interact with virtual characters that not only speak but also emote, fostering a deeper sense of realism and relatability. Moreover, the accessibility and ease of implementation of these technologies make them widely available, as all of them, except for the virtual reality glasses, are free and relatively easy to learn and implement, empowering creators and developers to enhance their virtual reality experiences.

The goal of this Final Degree Project was precisely to achieve these coherent facial animations and we have succesfully surpassed the initial goals of functioning facial animations and their integration into the VR application.

5.2. Conclusiones

A modo de conclusión de este proyecto, podemos desatacar que las animaciones faciales son una parte integral de cualquier personaje animado al que se le da voz y se espera que transmita un mensaje o conocimiento. Al sincronizar las expresiones faciales con audio procesado a tiempo real, hemos creado una experiencia más auténtica e inmersiva. Los usuarios ahora pueden interactuar con personajes virtuales que no solo hablan sino que también gesticulan, dando un sentido más profundo de realismo y relacionabilidad. Además, la accesibilidad y la facilidad de implementación de estas tecnologías hacen que estén ampliamente disponibles, ya que todas ellas, excepto las gafas de realidad virtual, son gratuitas y relativamente fáciles de aprender e implementar, lo que permite a los creadores y desarrolladores mejorar sus aplicaciones.

El objetivo de este Trabajo de Fin de Grado era precisamente conseguir estas animaciones faciales coherentes y hemos superado con éxito los objetivos iniciales de funcionamiento de las animaciones faciales y su integración en la aplicación VR.

5.3. Future lines of work

The most notorious forecast of this work is in the field of didactic-educational applications. Different improvement or lines of work could be:

- To modify the aforementioned application of San Cristóbal de La Laguna tourist and didactic guide so that the characters that are required to speak do so with these animations that process the audio and adapt the mouth in a coherent way.
- Complete the facial animation process in the rest of the application models.
- Optimize the VR application so it runs smothly.

5.4. Futuras líneas de trabajo

La previsión más notoria de este trabajo es en el campo de las aplicaciones didáctico-educativas. Diferentes mejoras o líneas de trabajo podrían ser:

- Modificar la citada aplicación de guía turística y didáctica de San Cristóbal de La Laguna para que los personajes que deban hablar lo hagan con estas animaciones que procesan el audio y adaptan la boca de forma coherente.
- Completar el proceso de animación facial en el resto de las modelos de aplicación.
- Optimizar la aplicación VR para que funcione sin problemas.

6. Presupuesto

En este apartado redactamos lo que ha supuesto, en términos de presupuesto, el desarrollo de este trabajo.

6.1. Trabajo e investigación

Primero, se dividen los costes por el tipo de trabajo realizado y las horas que se han dedicado a cada uno de ellos.

Tipo de trabajo	Cantidad [horas]	Coste Unitario [€/h]	Coste total [€]
Investigación previa	75	22	1.650
Desarrollo animaciones	100	22	2.200
Desarrollo aplicación	75	22	1.650

Integración VR	50	22	1.100
Subtotal			**6.600**

6.2. Material

A continuación, se muestran en forma de tablas los costes asociados con los materiales y el Hardware utilizado

Material	Cantidad [unidades]	Coste total [€]
Portátil OMEN HP	1	350
Monitor ASUS VZ24EHE 23.8" LED Full HD IPS 75Hz FreeSync	1	120
Meta Quest 2	1	400
Subtotal		**870**

6.3. Presupuesto final

Tipo de Coste	Coste total [€]
Trabajo e investigación	6.600
Material	870
Subtotal	**7.470**

7. Bibliografía

[1] «Inicio | UNESCO». https://www.unesco.org/es

[2] «Reseña histórica», *Portal web del Ayuntamiento de San Cristóbal de La Laguna*. https://www.aytolalaguna.es/ayuntamiento/el-municipio/resena-historica/index.html

[3] «Leonardo Torriani», *Wikipedia, la enciclopedia libre*. 27 de mayo de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Leonardo_Torriani&oldid=151469348

[4] «Reconstrucción Histórica Virtual de San Cristóbal de La Laguna». http://torriani.iaas.ull.es/

[5] J. L. R. Tamayo, «La realidad virtual para comunicar ciencia a través de la inmersión | Ciberimaginario». https://ciberimaginario.es/2018/06/26/realidad-virtual-inmersion-comunicacion-cientifica/

[6] «Half-Life: Alyx», *Half-Life*. https://half-life.com/es/alyx

[7] «Tilt Brush by Google». https://www.tiltbrush.com/

[8] J. R. Aponte, «La importancia de la comunicación no verbal en la enseñanza», *Ing. Solidar.*, 2010.

[9] U. Technologies, «Get Your Unity Pro Subscription Today | Unity». https://unity.com/pages/unity-pro-buy-now

[10] B. Foundation, «blender.org - Home of the Blender project - Free and Open 3D Creation Software», *blender.org*. https://www.blender.org/

[11] «Código abierto», *Wikipedia, la enciclopedia libre*. 7 de junio de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=C%C3%B3digo_abierto&oldid=151705664

[12] «Oculus VR», *Wikipedia, la enciclopedia libre*. 30 de enero de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Oculus_VR&oldid=148946220

[13] «Meta | Social Metaverse Company», *Meta | Social Metaverse Company*. https://about.meta.com/es/

[14] «El metaverso es el futuro de las conexiones digitales | Meta». https://about.meta.com/es/metaverse/

[15] «Viseme Reference: Unity | Oculus Developers». https://developer.oculus.com/documentation/unity/audio-ovrlipsync-viseme-reference/

[16] K. Olszewski, L. Joseph J., S. Saito, y H. Li, «High-Fidelity Facial and Speech Animation for VR HMDs», *High-Fidelity Facial and Speech Animation for VR HMDs*. https://vgl.ict.usc.edu/Research/FacialSpeechAnimation/

[17] W. Paier, A. Hilsmann, y P. Eisert, «Example-Based Facial Animation of Virtual Reality Avatars Using Auto-Regressive Neural Networks», *IEEE Comput. Graph. Appl.*, vol. 41, n.º 4, pp. 52-63, jul. 2021, doi: 10.1109/MCG.2021.3068035.
[18] «SAPI lipsync». http://www.annosoft.com/sapi_lipsync/docs/index.html
[19] «Introducción — Blender Manual». https://docs.blender.org/manual/es/latest/addons/rigging/rigify/introduction.html#main-features
[20] «Marvelous Designer», *Marvelous Designer Official Site*. https://marvelousdesigner.com/marvelousdesigner.com
[21] C. Meier, I. S. Berriel, y F. P. Nava, «Creation of a Virtual Museum for the Dissemination of 3D Models of Historical Clothing», *Sustainability*, vol. 13, n.º 22, Art. n.º 22, ene. 2021, doi: 10.3390/su132212581.