E.T.S. de Ingeniería Industrial,  
Informática y de Telecomunicación

Desarrollo de herramienta software para la edición de animaciones para retargeting de personajes



Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor: Adrian Guiral Mallart

Director: Óscar Ardáiz Villanueva

Pamplona, fecha de defensa



# Agradecimientos

WIP

# Palabras Clave

-Unity

-Blender

-Mixamo

-Rigging

-Pathing

-Mesh

-Collider

-Retargeting

-Keyframe

-API

# Motivación

La principal motivación del proyecto es la creación de una herramienta accesible para la modificación de animaciones 3D y su ‘retargeting’ a otros modelos 3D de forma sencilla y rápida. Además, esta motivación parte de la familiaridad con el entorno de desarrollo de Unity, en el cual ya tenía experiencia previa en el desarrollo de videojuegos en ambos Unity 2D y Unity 3D. Asimismo, esta experiencia facilitaría el trabajo de investigación en los distintos campos que abarca el proyecto y su implementación en la herramienta.

# Introducción

El ‘animation retargeting’ o reorientación de animaciones es una función que permite la reutilización de animaciones entre personajes que utilizan el mismo esqueleto, pero no comparten las mismas dimensiones [1]. Esta técnica es muy útil debido a que reduce la carga de trabajo del animador enormemente y solventa el problema de animar la misma animación para cada personaje nuevo.

Se trata de una técnica muy común y extendida principalmente dentro de la industria del videojuego, no obstante, también está presente en la industria del cine para ajustar animaciones a avatares 3D utilizados en la producción tanto de películas como series.

Para facilitar este trabajo, se va a utilizar el motor de desarrollo Unity 3D, uno de los motores más conocidos a nivel mundial y con mayor disponibilidad, teniendo a su disposición una amplia lista de herramientas y versiones.

# Objetivos

Los objetivos de este proyecto son muy claros, en primer lugar, se busca ajustar animaciones de un modelo 3D a modelos 3D con misma estructura ósea (‘rigging’) pero dimensiones irregulares de manera que la animación reorientada (‘retargeted’) opere de forma natural sobre los nuevos modelos. Esto se consigue a través de una edición mínima de la animación para solucionar errores causados por la naturaleza de las dimensiones del modelo. Se trata de una edición mínima para no provocar cambios en la finalidad de la animación.

En segundo lugar, se persigue desarrollar una herramienta de fácil uso que permita un acercamiento accesible al mundo de la animación en el ámbito principalmente de la creación y desarrollo de videojuegos.

# Resumen

Se busca desarrollar una herramienta en Unity 3D con C# para la edición y  
modificación de animaciones sobre modelos 3D. Haciendo uso de la  
herramienta el usuario podrá observar la animación y modificarla a través  
de controles tales como controles deslizantes o indicadores 3D sobre  
huesos del modelo. La herramienta permitirá la creación de estas nuevas  
animaciones y su almacenamiento con el fin de realizar un 'retargeting',  
es decir, ajustar la animación original a distintos modelos 3D.

# Animación en la industria

Para entender el ámbito en el que vamos a trabajar merece la pena entender la situación actual de la animación en la industria. Cabe destacar que la animación se encuentra en varios sectores de la industria como la industria del cine, la industria de los videojuegos, tecnología industrial y prácticamente todo el sector tecnológico.

Una vez visto el alcance de la animación en la industria podemos entender porqué el mercado global de la animación se estimó en 2022 con un valor de 391.19 mil millones de dólares americanos y se espera que crezca hasta los 587.1 mil millones de dólares americanos para el comienzo de 2030. Este crecimiento en la industria se debe principalmente a los avances en las comunicaciones entre sectores tecnológicos, el creciente interés y éxito de la industria del entretenimiento y la extensiva accesibilidad a internet en distintas regiones del mundo [2].

Gráfico, Gráfico de barras

Descripción generada automáticamentePor otro lado, el continuo desarrollo de las tecnologías de realidad virtual y realidad aumentada también brindan nuevas oportunidades a la industria de la animación, como pueden ser el desarrollo de videojuegos en realidad virtual o el desarrollo de metaversos, tendencia en aumento dentro del sector tecnológico, aunque sin mucho éxito como por ejemplo el metaverso de Meta (antiguamente Facebook) debido a un pobre desarrollo de este.

Figura 1: Estimación del crecimiento del mercado

Una caricatura de una persona

Descripción generada automáticamente con confianza baja

Figura 2: Fotografía del estado del metaverso por Meta

# Retargeting

Como se ha mencionado antes, el ‘retargeting’ o reorientación de animaciones es una técnica que consiste en la reutilización de animaciones entre personajes con mismo esqueleto.

El esqueleto de un modelo 3D está formado por articulaciones, esto son puntos designados a lo largo del modelo usados para realizar transformaciones como en la gran mayoría de ocasiones, rotaciones [3].

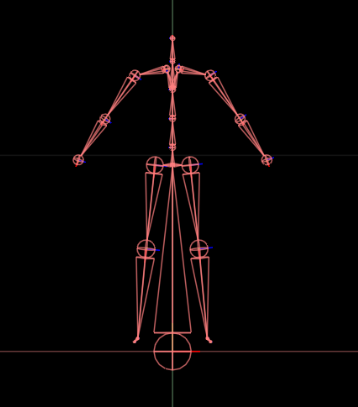


Figura 3: Esqueleto Modelo 3D [3]

Las articulaciones del esqueleto tienen una jerarquía 3D con relación padre-hijo que establece el orden de las transformaciones que se ejecutan en cada articulación para conformar una animación completa. Con esto en mente, la reorientación consigue trasladar las transformaciones almacenadas en cada articulación al esqueleto del modelo 3D que queremos animar con la limitación de que el esqueleto debe ser idéntico o casi idéntico manteniendo como mínimo el orden de jerarquía del esqueleto original.

# Materiales

## Hardware

### PC

Para desarrollar la herramienta se ha hecho uso de varios PCs, siendo uno de ellos el PC personal y otro el PC de la sala de proyectistas. Como previsualizador del contenido desarrollado en las reuniones se ha utilizado un portátil personal. Ambos PCs, tanto el personal como el de la sala de proyectistas tienen tarjetas gráficas potentes que cumplen el requerimiento de desarrollo que expresa Unity 3D. Adjunto las especificaciones de mi PC personal ya que no tengo las especificaciones exactas del PC de proyectistas más que la tarjeta gráfica es una Nvidia GeForce RTX 3060.

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza mediaPC personal:

Figura 4: PC

* CPU: AMD Ryzen 5 5600X 3.7GHz
* RAM: 16GB DDR4 3200Mhz
* Placa base: MSI MPG B550
* GPU: Gygabyte GeForce RTX 2060 (Nvidia)

Estas especificaciones son más que suficientes para desarrollar la herramienta sin problemas gráficos o de procesamiento a pesar de las limitaciones de Unity 3D.

## Software

### Unity 3D

Como se ha mencionado anteriormente, la herramienta se ha desarrollado en Unity 3D. Se trata de un motor gráfico para desarrollo de videojuegos multiplataforma cuyo desarrollador es Unity Technologies. El motor tiene soporte en Windows, Mac y Linux y fue lanzado al mercado el 30 de mayo de 2005.

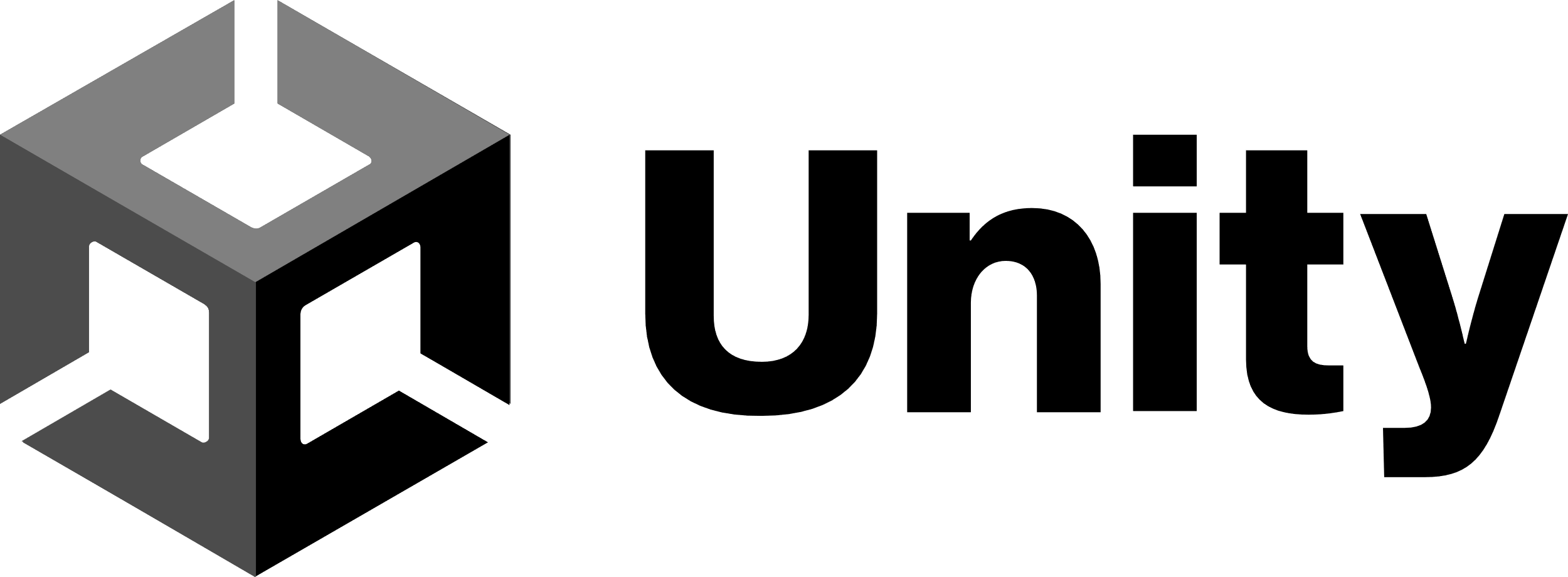
La herramienta llego al mercado con el objetivo de facilitar el desarrollo de videojuegos a desarrolladores independientes ofreciendo un motor de juego que sería muy difícil de crear por sus propios medios. De esta forma consiguieron que el desarrollo 2D y 3D fuera más accesible para personas interesadas en el desarrollo, pero sin medios para obtener un motor [4].

Figura 5: Unity Logo

### Blender

Blender es un programa de “open source” dedicado al modelado, renderizado y creación de gráficos 3D. Se trata de un “software” multiplataforma gratuito con una gran capacidad para crear modelos y animaciones 3D, disponiendo de una amplia gama de herramientas para su desarrollo. Además, es una herramienta muy popular con una curva de aprendizaje accesible, existen una gran cantidad de videos de la comunidad de Blender sobre su uso, aprendizaje, consejos y acercamiento a la herramienta [5].

Hemos usado esta herramienta para comprobar conceptos técnicos de modelaje 3D que no se pueden modificar desde Unity 3D y que ocasionaban problemas en el motor de desarrollo, este acercamiento a Blender como una herramienta de apoyo ha resultado de gran ayuda para una mejor implementación del proyecto.

Figura 6: Blender Logo

### Mixamo

Mixamo es una tecnología 3D para animación de personajes 3D, utiliza métodos de “machine learning” para automatizar los pasos del proceso de animación, incluyendo el modelaje 3D. Se trata de una tecnología perteneciente a Adobe y será nuestra base para recoger animaciones y modelos 3D.

Un dibujo de una cara

Descripción generada automáticamente con confianza bajaTodas las animaciones y modelos utilizados serán de mixamo ya que para realizar un retargeting al menos los modelos deben compartir una jerarquía de “rigging” muy parecida y las animaciones deben saber sobre que huesos (“rig”) deben ejecutarse, por lo tanto, es importante que las curvas de animaciones conozcan los nombres de los huesos (miembros del personaje 3D) para que Unity reproduzca correctamente la animación.

Figura 7: Mixamo Logo

# Tareas

## Pruebas iniciales con animaciones

Antes de comenzar el proyecto, se decidió que sería conveniente realizar una investigación previa sobre el problema que queríamos solucionar. Para ello se cargaron distintas animaciones sobre un modelo 3D y se exageró el modelo para obtener un personaje con piernas largas, uno con brazos largos y otro con una cabeza exageradamente grande.

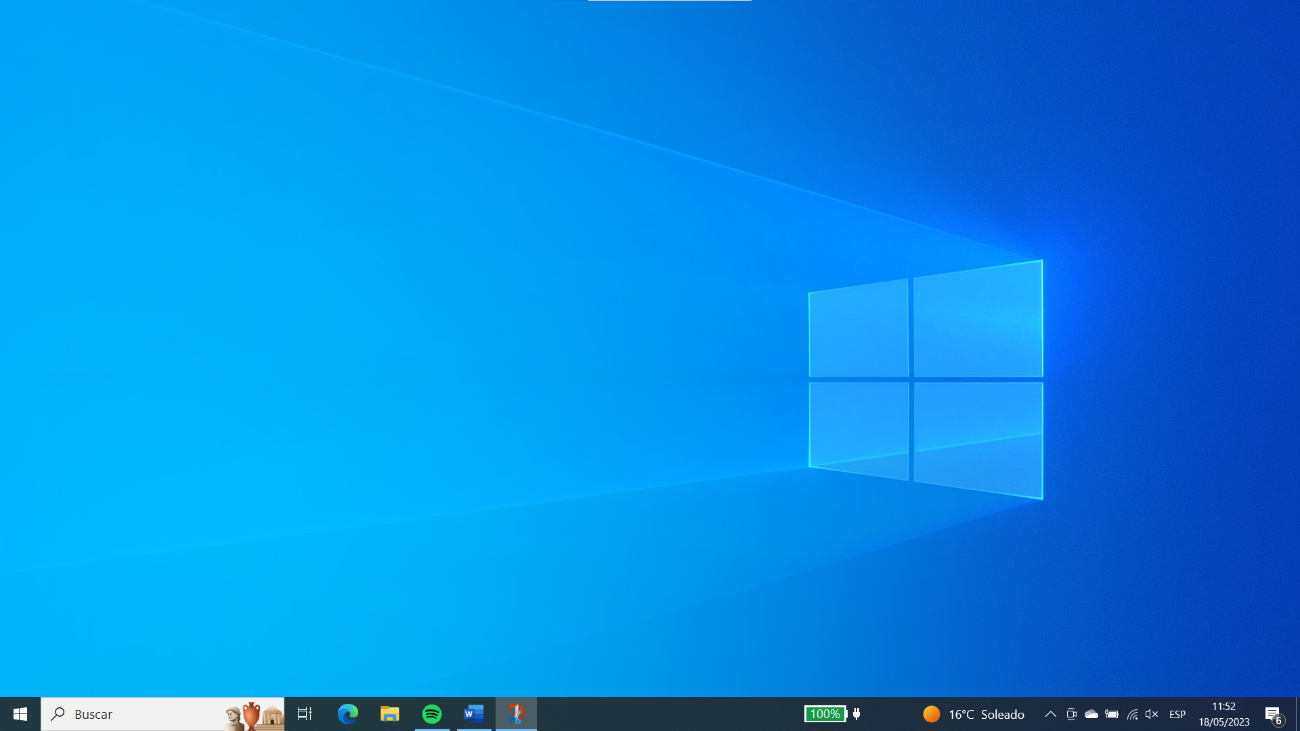
Seguidamente, se probaron las animaciones sobre estos tres personajes para visualizar el problema existente en el retargeting que pretendíamos resolver. Efectivamente, se podía observar que las animaciones en estos nuevos personajes producían comportamientos indeseados como superposición de miembros del personaje.

Figura 8: Personajes creados

Pantalla de computadora con fondo azul

Descripción generada automáticamente

Figura 9: Problema en retargeting sobre modelos

Se hicieron pruebas con otras animaciones para confirmar la existencia del problema.

Pantalla de computadora con fondo azul

Descripción generada automáticamente

Figura 10: Problema sobre animación lanzar pelota

Pantalla de computadora con fondo azul

Descripción generada automáticamenteTras confirmar la existencia del problema se procedió a la implementación de la herramienta y su respectiva investigación acerca de los temas que nos interesaban para conseguir una implementación de la herramienta accesible.

Figura 11: Problema sobre animación

## Selección de modelos y animaciones

Para hacer pruebas de la herramienta a lo largo del desarrollo se realizó una selección inicial de animaciones y modelos sobre los que trabajar. El modelo elegido fue el “bot” genérico de “mixamo” y las animaciones fueron un salto hacia delante y la animación de tirar la pelota usada en las pruebas iniciales con animaciones.

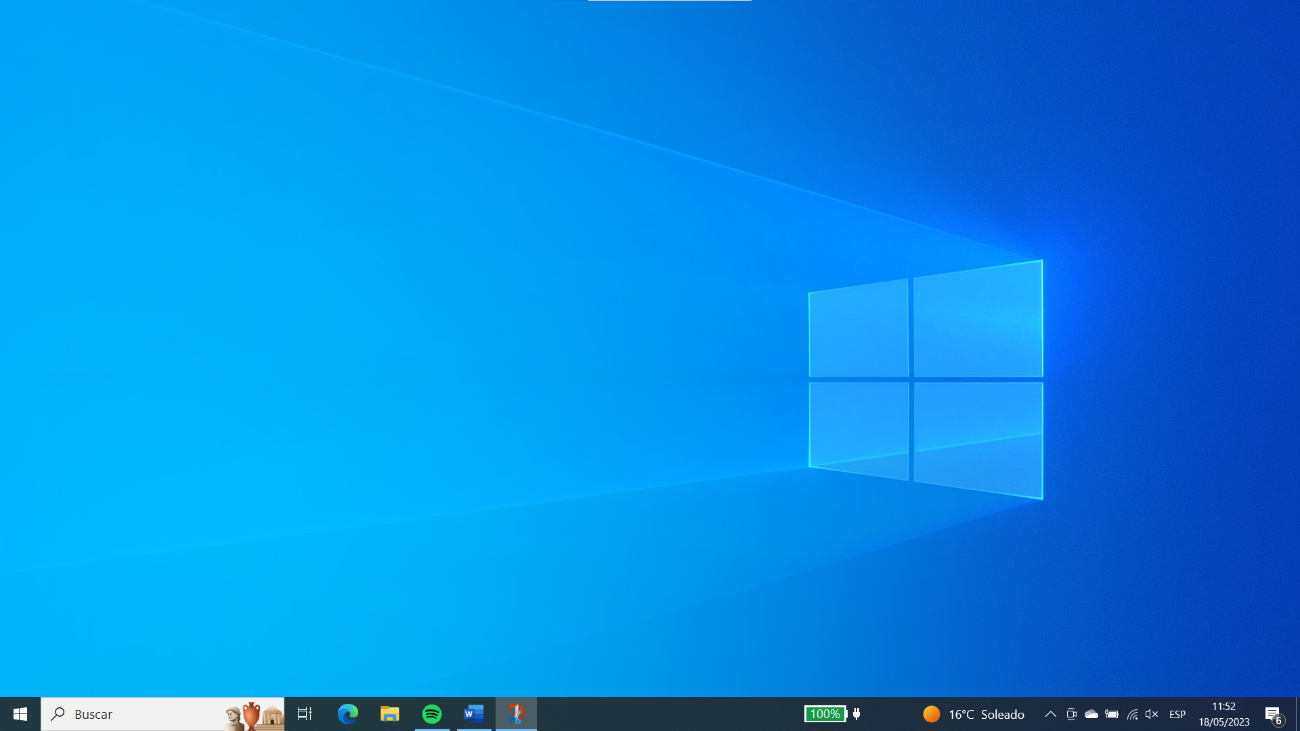


Figura 12: Modelo "mixamo" y animaciones

## Investigación sobre Unity 3D

El entorno de desarrollo de Unity 3D de forma resumida está conformado por un editor, un sistema de archivos, un sistema de escenas y una jerarquía de objetos para cada escena que está formada principalmente por una cámara, una fuente de luz y un conjunto de objetos.

Unity dispone de una web de aprendizaje llamada “Unity Learn” que proporciona videos y procesos descritos paso a paso sobre distintos temas y cuestiones para comenzar a desarrollar en Unity. “Unity Learn” dispone de tutoriales y cursos categorizados por dificultad y familiaridad con el entorno de desarrollo de Unity lo cual la convierte en una web muy útil para aprender conceptos de Unity [6].

Con los conocimientos adquiridos en la titulación de desarrollo de videojuegos de la UPNa sumados a “Unity Learn” para recordar conceptos e implementaciones para montaje de escena, reproducción de animaciones y manejo del editor se ha podido adquirir un nivel más que necesario para el desarrollo del proyecto. Todo esto sumado a las investigaciones sobre conceptos de naturaleza más técnica propios del tema del proyecto que se mencionarán a continuación.

## Investigación sobre Quaternions y rotaciones

Quaternion es un término matemático que hace referencia a un sistema numérico que extiende a los números complejos. Fueron descritos por primera vez por el matemático irlandés William Rowan Hamilton en 1843 y aplicado a la mecánica en el espacio tridimensional. Hamilton definió un “quaternion” como el cociente de dos líneas dirigidas en un espacio tridimensional, dicho de otra forma, el cociente de dos vectores.

La multiplicación de “quaternions” es no conmutativa y se representan generalmente con la forma *a + b i + c j + d k* donde *a, b, c* y *d* son números reales e *i, j, k* son vectores de base, conjunto de vectores *B* en un espacio vectorial *V* en el cual cada elemento de *V* puede ser escrito de forma única como una combinación lineal de *B [7]*.

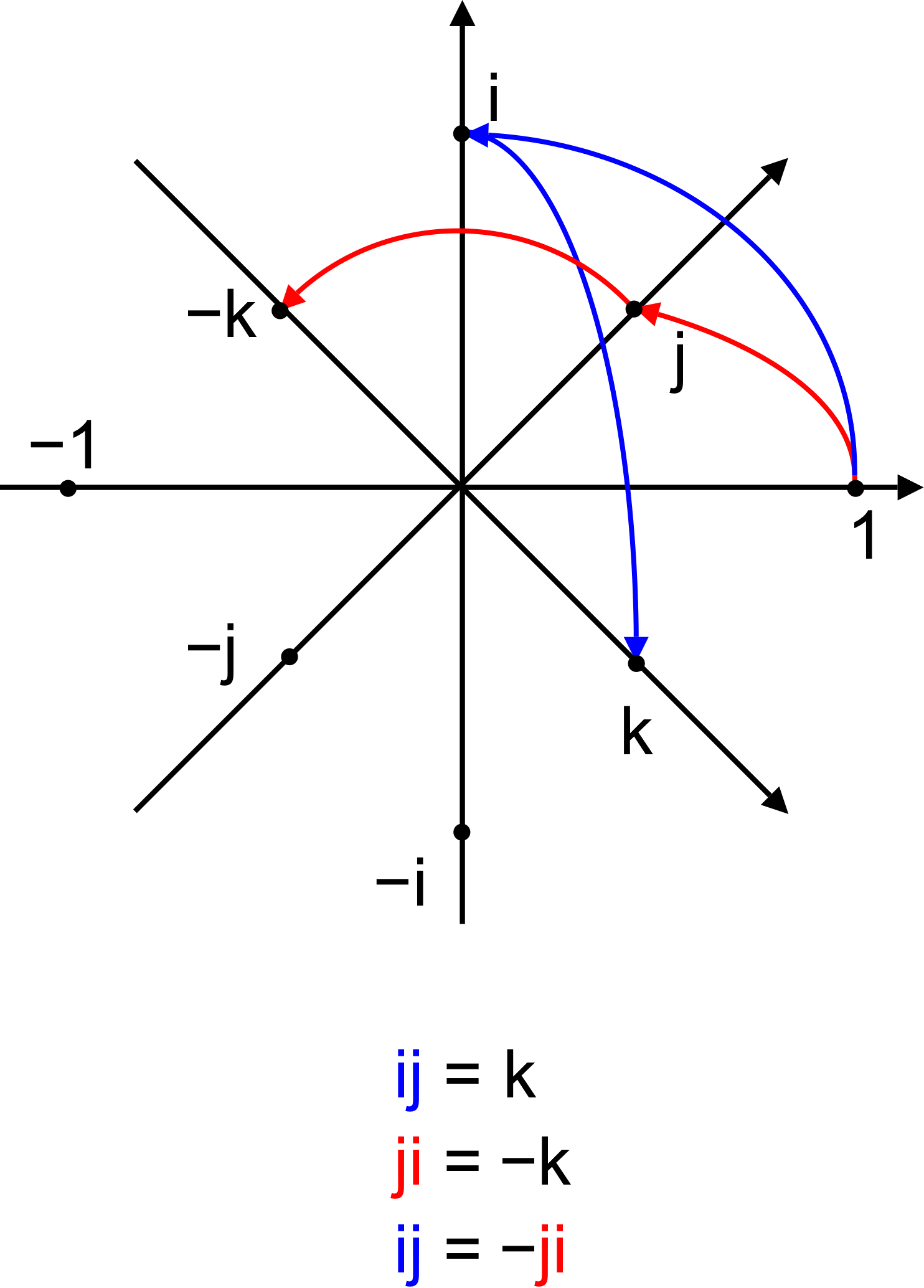


Figura 13: Quaternion

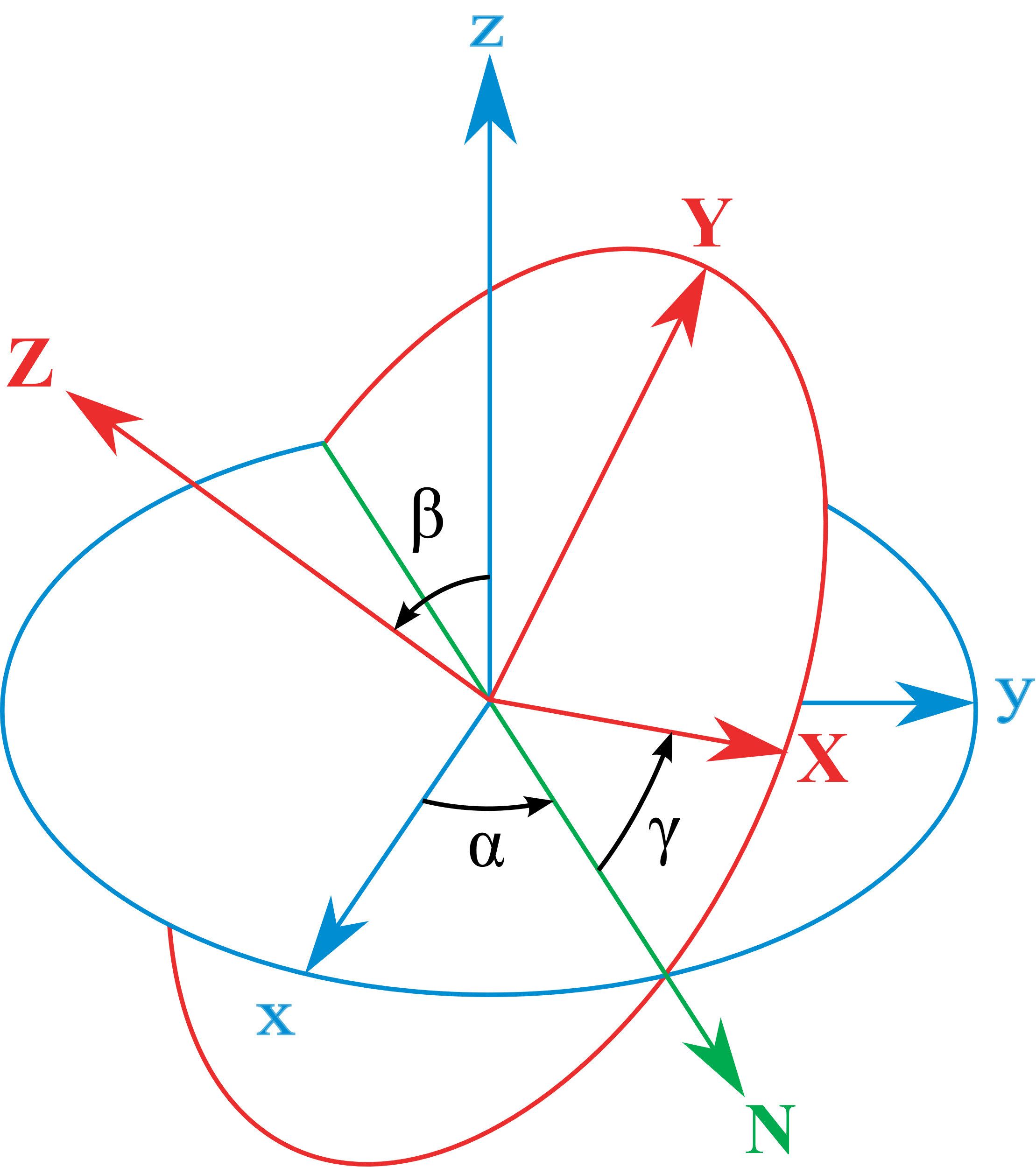
Los “quaternions” se utilizan en matemática pura pero también tienen usos prácticos en matemática aplicada especialmente en cálculos de rotaciones tridimensionales, principal tema de nuestro proyecto ya que las animaciones 3D son un conjunto de rotaciones tridimensionales y Unity maneja las rotaciones con “quaternions”. También se pueden utilizar con otros métodos de rotación como los ángulos eulerianos, tres ángulos introducidos por Leonhard Euler para describir la orientación de un “rigid body” con respecto a un sistema de coordenadas fijo [8], y matrices de rotación, matrices de transformación utilizadas para realizar rotaciones en el espacio euclidiano [9], dependiendo de su aplicación [10].

Figura 14: Ángulos eulerianos

Como se ha mencionado anteriormente, Unity 3D representa las rotaciones como “quaternions” o ángulos de Euler, aunque opera internamente con “quaternions”.

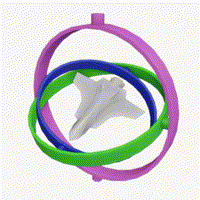
En el editor de Unity, los ángulos de Euler son representados por valores aplicados de forma secuencial en la X, Y y Z. Esto quiere decir que una rotación euleriana aplicada a un objeto sufre primero la rotación en el eje x seguida de la rotación en el eje y seguida de la rotación en el eje z. Los ángulos de Euler no son utilizados internamente por Unity porque tienen una limitación conocida como “Gimbal Lock” (bloqueo de cardán) que consiste en la perdida de un grado de libertad, numero de parámetros independientes en un sistema mecánico [11], en el espacio tridimensional que ocurre cuando dos de los ejes se alinean en paralelo de forma que las rotaciones se realizan en un espacio bidimensional [12].

Figura : Gimbal Lock

En el caso de los “quaternions”, Unity los representa como cuatro números (x, y, z, w), sin embargo, estos cuatro números no representan los ejes y no se deberían manipular directamente. Un “quaternion” puede representar tanto una orientación como una rotación donde la rotación se mide en base al origen de la rotación o identidad, “quaternion” que representa la no rotación. Como la representación se mide como el paso de una orientación a otra, no puede representar una rotación mayor que 180 grados.

Aunque Unity opere internamente con “quaternions”, el editor muestra los valores como ángulos de Euler, sin embargo, dependiendo del tipo de rotación que se quiera aplicar no se pueden tener en cuenta esos valores ya que existen dos tipos de rotaciones en un objeto: la rotación local y la rotación respecto al mundo (escena) y Unity muestra la rotación global. Ambas rotaciones se almacenan en el “transform” de un objeto, componente que determina la posición, rotación y escala de este.

De cara a la programación en Unity que se realiza normalmente en C# existen múltiples métodos que nos ofrece el propio Unity para evitar manipular “quaternions” directamente. Por otro lado, Unity también nos recomienda evitar transformar “quaternions” a ángulos de Euler, modificar y volver a aplicar a una rotación ya que puede causar efectos secundarios no deseados, efectos que no son mencionados ni explicados. Los métodos propios de Unity para los “quaternions” son los siguientes:

Texto

Descripción generada automáticamenteEn la implementación de la herramienta se utilizará alguno de los métodos aquí mostrados y de los que se hablará en puntos de la implementación [13].

Figura 16: Métodos Quaternions

## Investigación sobre animación y curvas de animación

En el ámbito de animación, Unity utiliza una clase propia llamada “Animation Clip” que representa los bloques para construir una animación en el entorno de Unity y se importan de los archivos FBX dentro del proyecto [14].

Los clips de animación están conformados por curvas de animación, cualquier propiedad animable puede tener una curva de animación dentro de un clip de animación lo cual significa que los clips controlan como estas propiedades cambian con el tiempo para formar una animación.

Interfaz de usuario gráfica

Descripción generada automáticamenteLas curvas de animación están compuestas por un conjunto de “keys” (claves), puntos de control que atraviesa la curva. Si estas llaves coinciden en un “frame” especifico de la curva, a ese “frame” se le denomina “keyframe” (“frame” clave).

Figura 17: Animation Curve para Position.y

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamenteEn el caso de animaciones complejas, hablaremos de animación compleja como cualquiera que contenga más de una curva de animación, se comparten unos keyframes específicos a lo largo de las curvas de animación que normalmente están relacionados con la tasa de refresco a la que se ejecutan las animaciones, es decir, los “frames” por segundo. De esta forma, en cada “keyframe” se actualizan los valores de las propiedades que toman parte en la animación.

Figura 18: La línea indica un keyframe para las curvas

A pesar de que internamente y como se ha mencionado previamente, Unity opera con “quaternions” para operar rotaciones, en el caso de las curvas de animación se puede interpolar entre dos rotaciones usando los valores del “quaternion” o los ángulos de Euler.

La interpolación usando los valores de “quaternion” genera animaciones más fluidas y evita el bloqueo de cardán, pero no puede representar rotaciones mayores que 180 grados. Esta será la interpolación que usaremos ya que no se realizan rotaciones mayores que 180 grados con una tasa de refresco alta como la que suelen tener las animaciones y es muy importante evitar el bloqueo de cardán para obtener animaciones precisas. Las interpolaciones mayores que 180 grados generan interpolaciones más pequeñas ya que por su comportamiento siempre encuentran el camino más corto para realizar la rotación [15].

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

Figura 19: Ejemplo de interpolación con Quaternions mayor que 180 grados

# Implementación

## Introducción

En los próximos apartados se va a describir de forma detallada los pasos, decisiones y desarrollos realizados para completar la implementación de una herramienta software para la edición de animaciones para “retargeting” de personajes y cumplir el objetivo de este proyecto.

Para iniciar el desarrollo del proyecto se requiere de la instalación de “Unity Hub” el cual maneja las licencias y versiones de Unity y el editor (versión) que se desea usar, en este caso, el editor 2021.3.19f1. Por otro lado, se precisa de la instalación de Blender por si debemos realizar cambios extremos en alguno de los modelos 3D y para investigar posibles comportamientos de estos. Por último, se necesita una herramienta de control de versiones para guardar el progreso en la implementación de forma segura, en este caso, se ha usado GitHub.

## Estructuras de datos

La implementación de Unity de los clips de animación mencionados anteriormente hace imposible su lectura ya que Unity no precisa de métodos para leer las curvas o “keyframes” de una animación por lo que para poder editar una animación se llegó a la conclusión de que se necesitaba grabar la animación original para así obtener los datos de rotación de cada uno de los miembros del modelo y modificarlos posteriormente. Esta grabación nos deja con una gran cantidad de datos que manejar y preservar a lo largo de la ejecución de la herramienta de forma que se implementaron dos estructuras de datos para guardar los datos ordenados y poder acceder a ellos de forma sencilla. La primera estructura guarda asociados al instante en el tiempo los datos de rotación y posición si se trata de las caderas, parte del modelo encargado de cambiar de posición si se trata de una animación con movimiento en algún eje ya sea un salto, correr, andar, etc.

Texto

Descripción generada automáticamente

Figura 20: Estructura Coordinates

La segunda estructura almacena para cada hijo (miembro) del modelo, una lista de la primera estructura mencionada junto con el nombre del hijo y su camino, es decir, según la jerarquía del modelo, donde se encuentra el hijo. En otras palabras, la estructura resultante nos permite almacenar para cada hijo los datos de rotación a lo largo del tiempo durante la animación y su posición respectiva a la jerarquía del modelo 3D.

Texto

Descripción generada automáticamente

Figura 21: Estructura Child Coordinates

## Captura de animación

Una captura de pantalla de una computadora

Descripción generada automáticamenteSe monta una escena inicial para comprobar que la grabación funciona, esta escena consta del modelo objetivo sobre el que se va a reproducir la animación original y el “script” de grabación junto con un plano que simulará un suelo para que se pueda percibir bien la posición del modelo.

Figura 22: Escena inicial

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteSe asigna un componente “animation” al modelo para poder reproducir clips de animación sobre el modelo, dentro del componente “animation” introducimos la animación que hemos importado y desde script accederemos a ese componente para activar la reproducción de la animación.

Figura 23: Componente animación del modelo

De cara a grabar la animación se requiere capturar los datos de los miembros del modelo en los instantes que se desea, para lograr este objetivo se ha hecho uso de una corrutina. Las corrutinas en Unity son funciones que permiten su ejecución a lo largo de una secuencia de tiempo, al contrario que las funciones normales que realizan su ejecución en un solo “frame” [16].

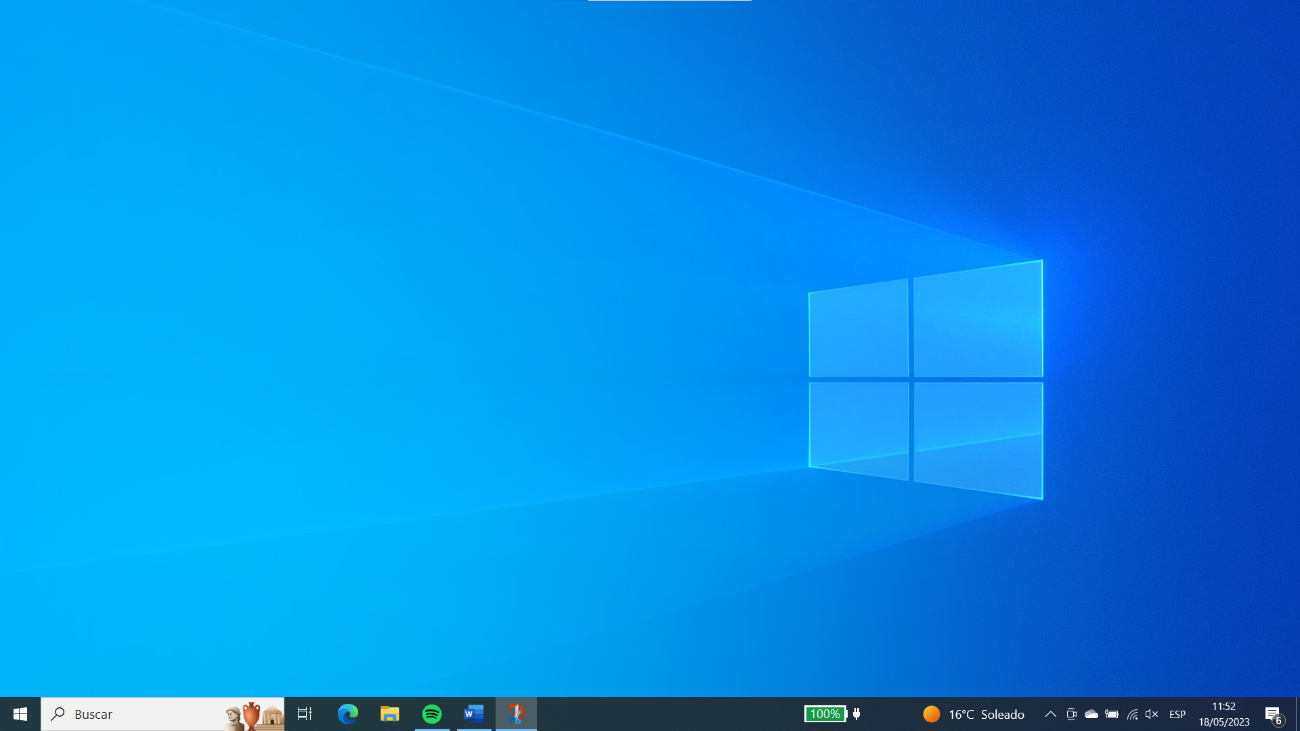


Figura 24: Animación original corriendo

De esta forma, se ha reproducido la animación original al mismo tiempo que empieza la corrutina y cada 0.02 segundos, medida elegida para conseguir 30 datos por segundo, se recorren los miembros del modelo para obtener los datos convenientes en esos instantes y se almacenan en las estructuras anteriormente mencionadas.

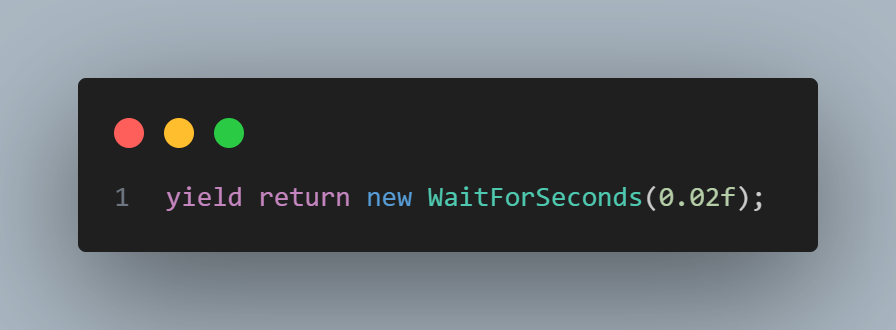


Figura 25: Función ejecutada en la corrutina para grabar datos cada 0.02 segundos

## Guardado de animación

Se necesita guardar los datos como un clip de animación para usar el sistema de reproducción de animaciones propio de Unity de forma que se va a construir un clip de animación con los datos obtenidos de la grabación de la animación original.

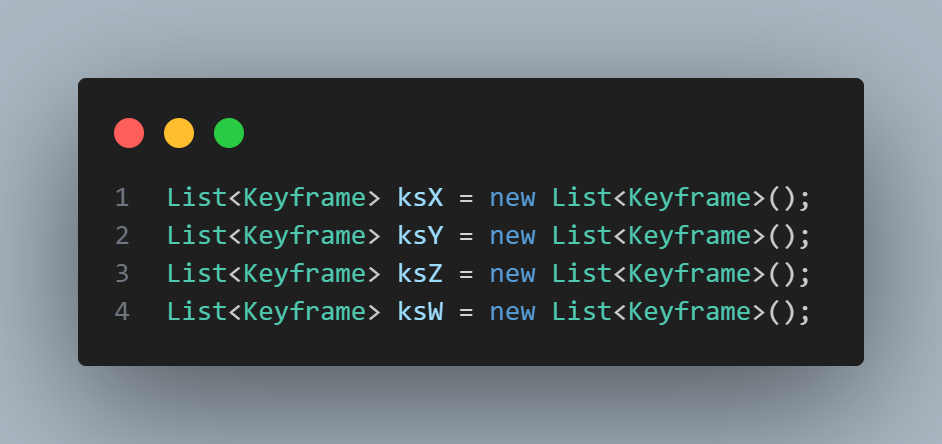
El primer paso para construir un clip de animación es declarar un objeto “AnimationClip” vacío con el nombre de la animación que vamos a guardar. A continuación, guardar cada uno de nuestros datos en forma de “keyframes” por lo que vamos a declarar una lista de keyframes para cada dato que se ha grabado, estos son los datos de rotación: ***x****,* ***y****,* ***z*** y ***w*** del “Quaternion” y los datos de posición: ***x***, ***y*** y ***z*** en caso de las caderas para, como se mencionó anteriormente, mover el modelo en caso de que se trate de una animación con eventos de traslación.

Figura 26: Listas de Keyframes

Texto

Descripción generada automáticamenteCabe destacar que esta declaración (Figura 26) es necesaria para cada miembro del modelo ya que el clip de animación debe conocer todos los datos de todas las partes del modelo para poder realizar una animación fiel y fluida. Después de esta declaración se recorren las estructuras de datos donde inicialmente se almacenó la información de la animación y se recorren desde el instante 0 de la animación hasta el último instante. Este recorrido se efectúa así porque los datos se deben introducir de forma ordenada. Los “keyframes” se añaden asociados a un instante en el tiempo y a su valor correspondiente.

Figura 27: Listas de Keyframes adicionales para las caderas

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteUna vez ya se han transformado todos los datos a “keyframes” e introducido en sus respectivas listas se crean las curvas de animación que serán las encargadas de pasar de un “keyframe” a otro de forma fluida para completar la animación.

Figura 28: Ejemplo de añadir un nuevo keyframe para el instante: chCoord.time con el valor: chCoord.rotX

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 29: Creación de una curva de animación con el listado de keyframes

Para finalizar, las curvas se añaden al clip de animación que hemos declarado vació al principio. Las curvas se deben añadir con una referencia a la posición en la jerarquía del modelo de la parte del modelo y el tipo de dato que es.

Texto

Descripción generada automáticamenteSi se desea exportar la animación como un “asset” para uso en otros proyectos o como archivo para guardar de forma definitiva se hace uso de un método de Unity para la creación de “assets”.

Figura 30: Curva añadida a clip de animación, Posición del miembro: chChoords.path, tipo: transform localRotation.x

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 31: Creación de archivo .anim

## Reproducción de animación

### Reproducción con curvas

### Reproducción manual

## Modificación de animación

## Cámara y movimiento

## UI

### Configuración de sliders

### Menú de elección de personajes

## Detección de colisiones

# Problemas encontrados

## Documentación de Scripting API Unity 3D

## Modificación de rotaciones

desafortunadamente y por la naturaleza de nuestro objetivo, realizar una edición del retargeting para ajustar correctamente una animación a nuevos modelos, la edición de la animación requiere de la manipulación de “quaternions” de forma manual ya que no se conoce el “quaternion” objetivo para realizar una rotación de “quaternion” a “quaternion”

## Reproducción manual de animación

## Pathing de hijos de modelo 3D

## Rigging modelo 3D

## Creación malla 3D para detección

# Posibles próximos objetivos

# Manual

# Conclusión

# Referencias

*[1] Unreal Engine 5. Animation Retargeting. (n.d.).* [*https://docs.unrealengine.com/5.0/en-US/animation-retargeting-in-unreal-engine/*](https://docs.unrealengine.com/5.0/en-US/animation-retargeting-in-unreal-engine/)

*[2] What is the current size of the global animation market? (n.d.). Retrieved May 22, 2023, from* [*https://www.precedenceresearch.com/animation-market*](https://www.precedenceresearch.com/animation-market)

*[3] NVIDIA Blogs. ¿Qué es Animation Retargeting? (2022, June 30).* [*https://la.blogs.nvidia.com/2022/06/30/que-es-animation-retargeting/*](https://la.blogs.nvidia.com/2022/06/30/que-es-animation-retargeting/)

*[4] Unity (motor de videojuego). (n.d.).* [*https://es.wikipedia.org/wiki/Unity\_(motor\_de\_videojuego)*](https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego))

*[5] Blender. (n.d.).* [*https://es.wikipedia.org/wiki/Blender*](https://es.wikipedia.org/wiki/Blender)

*[6] Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, &*

*Real-time 3D | Unity Learn (n.d.).* [*https://learn.unity.com/*](https://learn.unity.com/)

*[7] Basis (linear algebra) (n.d.)* [*https://en.wikipedia.org/wiki/Basis\_(linear\_algebra)*](https://en.wikipedia.org/wiki/Basis_(linear_algebra))

*[8] Euler angles (n.d.).* [*https://en.wikipedia.org/wiki/Euler\_angles*](https://en.wikipedia.org/wiki/Euler_angles)

*[9] Rotation matrix (n.d.).* [*https://en.wikipedia.org/wiki/Rotation\_matrix*](https://en.wikipedia.org/wiki/Rotation_matrix)

*[10] Quaternion (n.d.).* [*https://en.wikipedia.org/wiki/Quaternion*](https://en.wikipedia.org/wiki/Quaternion)

*[11] Degrees of freedom (mechanics). (n.d.).* [*https://en.wikipedia.org/wiki/Degrees\_of\_freedom\_(mechanics)*](https://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics))

*[12] Gimbal Lock (n.d.).* [*https://en.wikipedia.org/wiki/Gimbal\_lock*](https://en.wikipedia.org/wiki/Gimbal_lock)

*[13] Rotación y Orientación en Unity. (Copyright © 2020 Unity Technologies. Publication 2019.4).* [*https://docs.unity3d.com/es/2019.4/Manual/QuaternionAndEulerRotationsInUnity.html*](https://docs.unity3d.com/es/2019.4/Manual/QuaternionAndEulerRotationsInUnity.html)

*[14] Animation Clip (Clip de animación). (© 2016 Todos los derechos reservados. Unity Technologies. Publication 5.3-Q).* [*https://docs.unity3d.com/es/530/Manual/class-AnimationClip.html*](https://docs.unity3d.com/es/530/Manual/class-AnimationClip.html)

*[15] Using Animation Curves. (Copyright © 2021 Unity Technologies. Publication Date: 2023-05-12.).* [*https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html*](https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html)

*[16] Technologies, U. (n.d.). Coroutines (corrutinas). Unity Manual. Retrieved May 23, 2023, from* [*https://docs.unity3d.com/es/2018.4/Manual/Coroutines.html*](https://docs.unity3d.com/es/2018.4/Manual/Coroutines.html)

*Figura 1: What is the current size of the global animation market? (n.d.). Retrieved May 22, 2023, from* [*https://www.precedenceresearch.com/animation-market*](https://www.precedenceresearch.com/animation-market)

*Figura 3: De Unity Technologies -* [*https://brandguide.brandfolder.com/unity/downloadbrandassets*](https://brandguide.brandfolder.com/unity/downloadbrandassets)*, Dominio público,* [*https://commons.wikimedia.org/w/index.php?curid=110761106*](https://commons.wikimedia.org/w/index.php?curid=110761106)

*Figura 4: De ™/®Blender Foundation - Vectorised by Vulphere from http://www.blender.org/ (SVG code), Dominio público,* [*https://commons.wikimedia.org/w/index.php?curid=35129654*](https://commons.wikimedia.org/w/index.php?curid=35129654)

*Figura 5: By Mixamo - personally, CC BY-SA 3.0,* [*https://commons.wikimedia.org/w/index.php?curid=34638391*](https://commons.wikimedia.org/w/index.php?curid=34638391)

*Figura 8: By Maschen - Own work, CC BY-SA 3.0,* [*https://commons.wikimedia.org/w/index.php?curid=27793856*](https://commons.wikimedia.org/w/index.php?curid=27793856)

*Figura 9: By Lionel Brits - Hand drawn in Inkscape by me, CC BY 3.0,* [*https://commons.wikimedia.org/w/index.php?curid=3362239*](https://commons.wikimedia.org/w/index.php?curid=3362239)

*Figura 10: By Drummyfish - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=77738933*

*Figura 11: By Unity Documentation* [*https://docs.unity3d.com/es/2019.4/Manual/QuaternionAndEulerRotationsInUnity.html*](https://docs.unity3d.com/es/2019.4/Manual/QuaternionAndEulerRotationsInUnity.html)

*Figura 12: By Unity Documentation* <https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html>

*Figura 13: By Unity Documentation* <https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html>

*Figura 14: By Unity Documentation* <https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html>