Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

MÁSTER UNIVERSITARIO EN DESARROLLO DE VIDEOJUEGOS

Trabajo Fin de Máster

Animaciones procedurales mediante cinemática inversa en Unreal Engine 5.

Procedural animations using inverse kinematics in Unreal Engine 5.

Autor Ernesto Echeverría González

D./Dña. José Ignacio Estevez Damas, con N.I.F. 43.786.097P profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

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

Que la presente memoria titulada:

“Animaciones procedurales mediante cinemática inversa en Unreal Engine 5”

ha sido realizada bajo su dirección por D. Ernesto Echeverría González, con N.I.F. 54064117-H.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de julio de 2023.

Agradecimientos

Quiero agradecer en primer lugar a mis padres por darme la oportunidad de estar aquí y aguantar más que yo. Hablando de aguantar, tengo que nombrar a mi mujer, Laura, por aguantar mis nervios, sobre todo en los últimos momentos.

También merecen una mención especial Nacho, mi tutor, y Jesús, el director de este máster, por brindarme su ayuda y apoyo cuando lo necesitaba, aunque yo mismo no lo pidiera.

Por último, quiero agradecer a todos mis amigos que me han ayudado en algún punto durante el máster. A Antuan por su música y sacarse herramientas de la manga, a Nacho por sus redacciones, a José por sus ideas, a Manu por devolverme las ilusiones con sus proyectos y al resto de Agentes de Chill por su apoyo. Y a Dani por sacar un ratito en su viaje desde Madrid para hablar desde el conocimiento del sector de las capacidades requeridas para sacar esto adelante.

Resumen

A lo largo de este trabajo, presentaremos la implementación de la herramienta de IK Rig (Inverse Kinematics) en Unreal Engine 5 para generar animaciones realistas de escalada en un juego de tipo puzle basado en el clásico de 1965 “Mastermind” [B10]. El objetivo principal será comprobar las ventajas que proporciona esta herramienta a la hora de interactuar con el entorno del personaje.

En primer lugar realizaremos una breve introducción al concepto de animación en videojuegos y el desarrollo histórico del mismo así como las mejoras más importantes del sector, que ha afectado de igual manera a los videojuegos y al mundo cinematográfico.

A continuación se detalla el GDD (Game Design Document) del minijuego creado para dar contexto al experimento realizado, que a su vez explica las necesidades de las diferentes animaciones generadas.

Posteriormente desarrollaremos las posibles herramientas a utilizar para generar animaciones con cinemática inversa que se probaron en este trabajo con Unreal Engine 5: el Control Rig y el IK Rig, así como sus ventajas y limitaciones en la creación de animaciones de escalada.

Para concluir, trataremos aquellas posibles mejoras a desarrollar en un entorno óptimo, es decir, con personal cualificado y sin limitaciones temporales.

Palabras clave: animación, animación procedural, cinemática inversa, cinemática directa, Unreal Engine, IK Rig, Control Rig

Abstract

Throughout this work, we will present the implementation of the IK Rig (Inverse Kinematic) tool included within Unreal Engine 5 in order to generate realistic climbing animations for a puzzle game based in the 1965 classic “Mastermind” [B10]. The main objective will be testing the advantages that these tools offer us with the goal of allowing the character to interact with the environment.

Firstly, we will provide a brief introduction of the video game animation concept and its historical development as well as the most important upgrades of its environment, that have shown its impact in the video game and the cinematic industries.

Secondly, the GDD (Game Design Document) of the minigame created to add context to the experiment is described. This test also explains the necessity of the set of animations created.

We will then expand on the possible tools that could generate animations using inverse kinematics and forward kinematics within Unreal Engine 5 that have been tested throughout this assignment: the Control Rig and the IK Rig, and their advantages and limitations while creating climbing animations.

Finally, we will provide some upgrades that could be made to the game if an optimum environment was allowed, with qualified animators and 3D modelers and without any temporal limitation.

Keywords: animation, procedural animation, inverse kinematic, forward kinematic, Unreal Engine, IK Rig, Control Rig

Índice

Capítulo 1
Introducción

Los campos de la producción cinematográfica y el desarrollo de videojuegos han presenciado a lo largo de su historia el desarrollo de las animaciones por ordenador hasta llegar al objetivo de este trabajo, la animación procedural.

La animación procedural es el paradigma de la animación que nos ofrece mayor flexibilidad y eficiencia sin reducir el realismo ni la naturalidad de los movimientos, ya que crea o modifica dichas animaciones de acuerdo a unas reglas preestablecidas y elimina la necesidad de su creación fotograma a fotograma.

Por este motivo, se ha propuesto como foco de este trabajo la creación de un prototipo que haga uso de esta técnica en el desarrollo de una mecánica de un videojuego 3D, como la mecánica de escalada utilizada en juegos como Uncharted o Shadow of the Colossus. Para ello, al principio intenté utilizar Unreal Engine 5.0 con su herramienta experimental Power IK, aunque la aparición de Unreal Engine 5.1 y el IK Rig me convenció lo suficiente como para decidir cambiar a esta versión del motor.

Unreal Engine es un motor de videojuegos creado por Epic Games con la intención de dotar de herramientas a los desarrolladores de videojuegos de tipo shooter. Sin embargo, debido a las mejoras en versiones posteriores, se ha consolidado como uno de los mayores motores de la industria. Ahora mismo la versión vigente es la 5.3, más centrada en la creación de mundos abiertos, la mejora de procesos mediante la IA y el fotorrealismo.

1.1. Antecedentes

En sus inicios, los videojuegos no contaban con interfaz gráfica y, por tanto, tampoco con animaciones, pero todo cambiaría con el nacimiento de Spacewar!, donde varias naves se movían por la pantalla. El Spacewar!, sin embargo, no aportaba nada nuevo a la animación, ya que la única función que debía desarrollarse era el movimiento de elementos estáticos por la pantalla, como ya hacía previamente el Pong.

La evolución natural de este tipo de animación fue la inserción de vídeos como un conjunto de fotogramas dibujados píxel a píxel, que luego pudieron ser realizados mediante herramientas de dibujo o escaneo de dibujos físicos.

La revolución en el mundo de la animación vio su chispa arder en la cultura cinematográfica con la creación del rigging para la película Toy Story, de 1995. El rigging es una técnica que asigna a cada personaje una estructura ósea virtual que permite la abstracción y la creación de animaciones como un conjunto de posiciones y rotaciones de cada uno de los huesos del esqueleto. Tras este gran logro, se volvería una técnica ampliamente utilizada tanto en el mundo cinematográfico como en el de los videojuegos.

En estas últimas décadas hemos podido ver la inclusión de la técnica de cinemática inversa, utilizada principalmente como un medio para asignar posiciones a los elementos de un sistema robótico. Al aplicarse a la animación, nos permite asignar a un esqueleto posiciones de objetivos con los que interactuar. Precisamente en este proceso se centra el presente trabajo.

1.2. Objetivo General

El objetivo de este trabajo es generar animaciones procedurales, justificadas como parte de una mecánica que las requiera. Para ello se ha tomado como referencia el vídeo "Animation Bootcamp: An Indie Approach to Procedural Animation" [B9] de GDC, que proporciona unas directrices generales para desarrollar animaciones procedurales en distintos entornos.

1.3. Objetivos Específicos

Nuestro objetivo específico será crear animaciones procedurales para la escalada, tomando en este caso como referencia la saga Uncharted. Esta escalada forma parte de una mecánica que permite avanzar o completar el nivel. Para alcanzar este objetivo, primero se requiere la creación de un rig, al que posteriormente le añadiremos animaciones de cinemática directa que serán modificadas programáticamente mediante cinemática inversa, con el fin de adaptar el personaje a la posición de piedras y paredes.

1.4. Estado Actual del Tema

En esta última década no sólo hemos podido ver la adopción de la cinemática inversa como un elemento más de la caja de herramientas del desarrollo de videojuegos, sino que ante el acelerado crecimiento de las tecnologías basadas en inteligencia artificial también ha salido beneficiada la animación, ya que estas herramientas empiezan a ser capaces de animar completamente un rig utilizando prompts. Sin embargo, debido al tiempo de desarrollo requerido para un videojuego, la mayoría de los que salen a la venta actualmente siguen

utilizando animación mediante rigging con cinemática directa (como Toy Story) o cinemática inversa.

A pesar de haber utilizado la cinemática inversa en un contexto de escalada a lo largo de este trabajo (como Uncharted o Shadow of the Colossus con su escalada de paredes o Mirror's Edge al subir tuberías [Figura 1.2]), este modo de trabajo no se limita a esta posibilidad. De hecho, es más común ver juegos que incluyen la cinemática inversa para solucionar pequeños problemas visuales como partes del cuerpo atravesando paredes o suelos o incluso flotando sobre escaleras [Figura 1.1].

Figura 1.1. Pokémon Leyendas Arceus

Figura 1.2. Mirror's Edge

Capítulo 2
Documento de Diseño de Videojuegos

Climbing Mastermind

Categoría: Puzles

Licencia: N/A

Tecnología: PC

Público objetivo: Desarrolladores interesados en las animaciones procedurales

2.1 Visión general del juego

La función principal de este juego no es el ocio, aunque se vea reflejado como un elemento secundario, sino la experimentación de las animaciones procedurales mediante el uso de cinemática inversa. En cuanto a la jugabilidad, pretende ser un puzle de mecánicas sencillas que utilice la animación de escalada y que pueda desarrollarse de forma iterativa, como un juego arcade.

2.2 Mecánica del juego

Durante el desarrollo de cada nivel, el jugador se debe mover utilizando las teclas WASD. El salto se efectuará con la barra espaciadora, que también sirve para inicializar la escalada. Una vez en posición de escalada, se puede mantener las teclas anteriormente nombradas para apuntar a la siguiente piedra en caso de haberla. Mientras el personaje apunta a una piedra, la barra espaciadora propiciará un salto hacia la piedra apuntada. Mediante la tecla E se selecciona el color de la piedra en la que se encuentra el personaje como siguiente entrada del puzle.

2.3 Estado del juego

Número de nivel: se almacena como entero positivo el número de niveles superados a razón de utilizarlo como puntuación de la partida.

Número de intentos del nivel: se almacena como un entero positivo que puede ir de cero a doce. Al alcanzar los doce intentos sin solucionar el puzle, se pasará a la pantalla de *game over*.

Objetivo: se trata de un vector de colores con 5 posiciones que varía de forma aleatoria en cada nivel. El vector almacena los colores que el jugador debe alcanzar, así como su orden.

Entrada del jugador: se trata de un vector que comienza vacío y debe alcanzar las cinco posiciones con los colores introducidos por el jugador. Al llegar a la quinta posición, si este vector se corresponde con el objetivo, se crea un nuevo nivel, aleatorizando un nuevo objetivo, incrementando el número de nivel, reiniciando el número de intentos de nivel y vaciando la entrada; si el vector no se corresponde con el objetivo, se incrementa el número de intentos y se vacía igualmente la entrada.

2.4 Interfaces

El juego se compondrá inicialmente de dos pantallas:

- La pantalla del nivel, que incluye un HUD con el nivel en el que se encuentra el jugador, el número de intentos de los que dispone en el nivel actual y los colores que ha introducido a lo largo de este nivel [Figura 2.1].

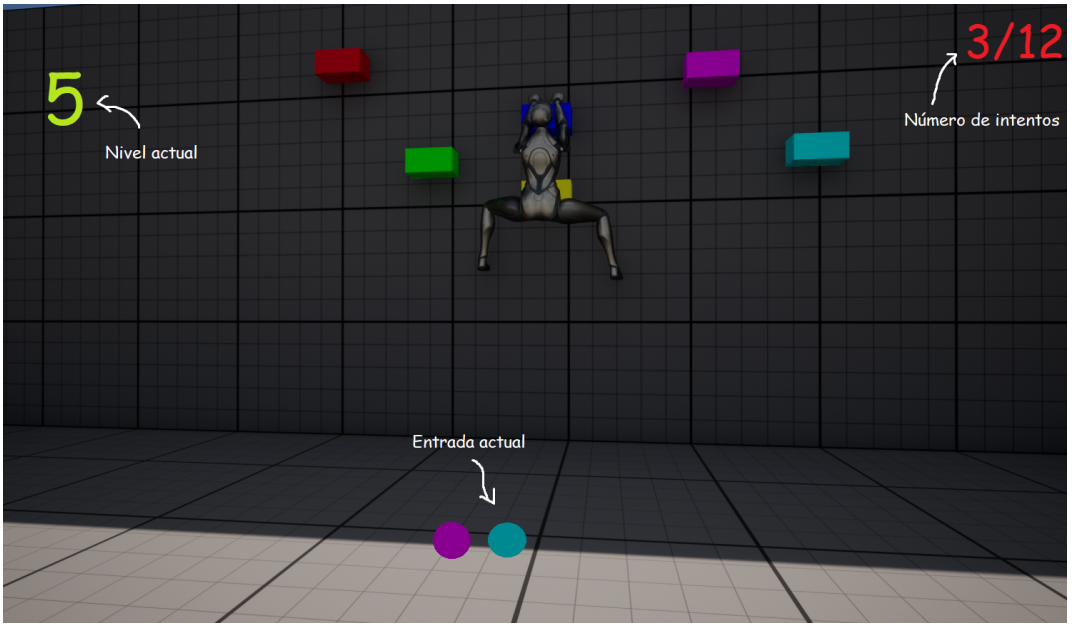

Figura 2.1. Pantalla de nivel de juego

- La pantalla de game over, con dos botones para reiniciar o cerrar el juego.

2.5 Niveles

Los niveles del juego se basan en una habitación con piedras para escalar en una pared. Los colores de cada piedra, que se corresponden con las entradas del puzle, se asignan de forma aleatoria al comienzo de cada nivel.

2.6 Progreso del juego

El juego tiene un progreso de tipo arcade, donde se pierde al no solucionar el puzle en el número máximo de intentos y se puede avanzar infinitamente por sus niveles.

2.7 Personajes

El único personaje existente en el juego es el avatar del jugador. Al tratarse de un puzle, la personalidad del mismo no tiene ningún tipo de influencia, por lo que no será desarrollada.

2.8 Música y sonidos

La música de fondo planea ser poco llamativa pero rítmica, a fin de inducir al jugador una sensación de prisa inexistente. También se utilizarán sonidos cada vez que el jugador active una piedra para indicarle que la entrada ha sido recibida por el programa.

Capítulo 3
Desarrollo del trabajo

En este apartado describiremos los pasos a seguir mediante dos enfoques diferentes para desarrollar la cinemática inversa en Unreal Engine 5

3.1. Animaciones en Unreal Engine 5.1

Para generar animaciones en Unreal Engine, debemos producir un conjunto de assets por orden:

- En primer lugar, necesitaremos el Rig de un personaje (es decir, la representación de su esqueleto que facilita su movimiento), así como su modelo. Generalmente ambos assets se generan en un programa externo, como podría ser Blender o Maya. También es posible descargarlos y/o comprarlos desde alguna web externa, como podría ser Mixamo. Sin embargo, en nuestro caso, utilizaremos como base los dados por el starter content de Unreal Engine. También en este paso se crea el Control Rig en caso de usar esta herramienta.
- Una vez tengamos el rig, crearemos un Animation Sequence Asset o secuencia de animación. Para ello utilizaremos la herramienta Animator, que nos permite posicionar y rotar los elementos del rig deseados en una escala temporal y cíclica.
- En el caso de desear fusionar de forma orgánica dos secuencias de animaciones, utilizaremos los Blend Space, que mediante un vector dimensional (generalmente de una o dos dimensiones), mezcla un conjunto de secuencias con mayor o menor importancia de cada una de ellas.
- Para terminar la animación, utilizaremos el Animation Blueprint, donde lanzaremos animaciones y las modificaremos (mediante el vector del Blend Space o IK Rig) al detectar diferentes eventos. Este Animation Blueprint es un elemento que actúa sobre el rig de un actor de tipo Character. Tiene dos partes: el grafo de eventos [Figura 3.1] y el grafo de animación [Figura 3.2], que a su vez posee una máquina de estados que permite diversas operaciones (como el IK Rig y el Blend).

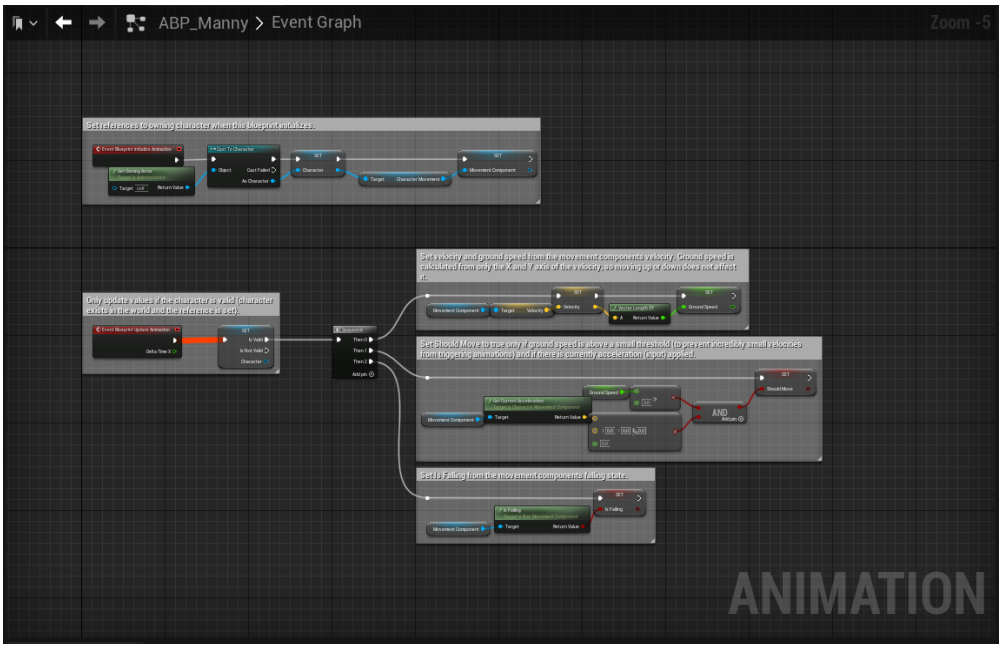

Figura 3.1. Event Graph del Animation Blueprint

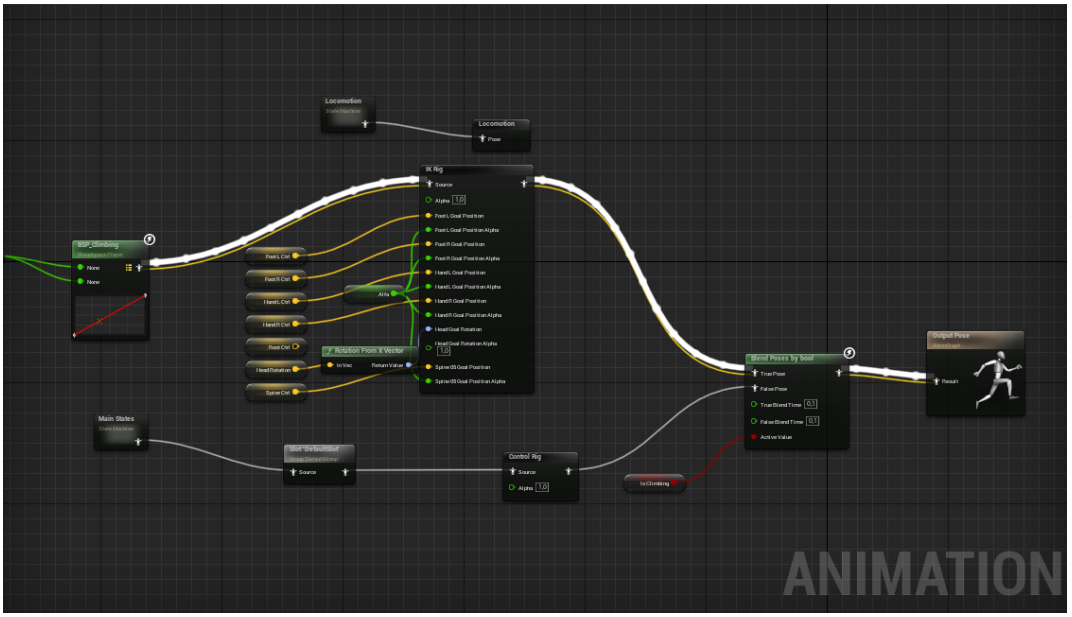

Figura 3.2. AnimGraph del Animation Blueprint

- Para controlar dichos eventos, necesitaremos lanzarlos desde otra sección del juego, en nuestro caso, y como norma general, desde el Character Blueprint.

3.2. Control Rig

El Control Rig permite la creación de un rig más versátil, que permite simultáneamente el control de sus huesos mediante el uso de cinemática directa o creando objetivos para cinemática inversa.

3.2.1. Ventajas del Control Rig

El Control Rig es una herramienta mucho más configurable, que nos permite, mediante blueprints o programación, la asignación de huesos a elementos de tipo Transform con el fin de administrar la cinemática directa [Figura 3.3].

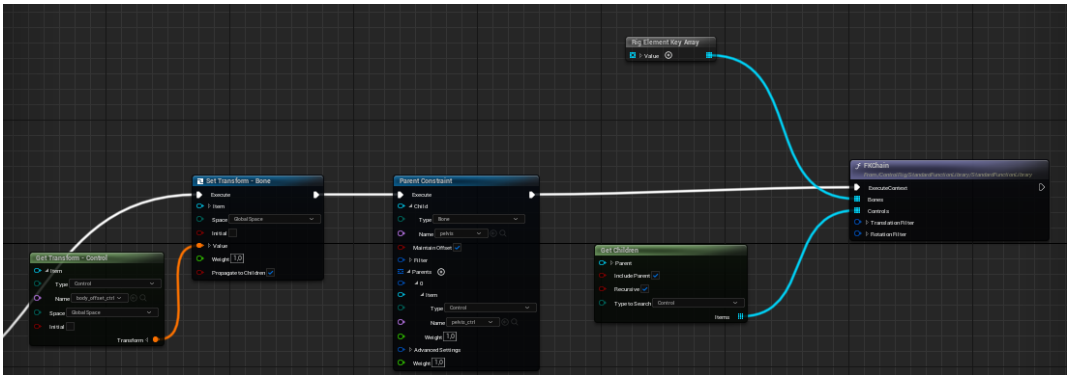

Figura 3.3. Configuración de cinemática directa en Control Rig

De igual modo podemos crear cinemática inversa, para lo que debemos utilizar blueprints que indiquen la cadena de huesos [Figura 3.4], generalmente definida por tres huesos y dos controles: un origen también llamado Pole y un elemento final. En caso de ser una cadena mayor, el bloque a seleccionar cambiaría [Figura 3.5].

Pero el Control Rig no se limita a ambos tipos de cinemática, sino que nos permite también definir la posición de todo un bloque, hacia dónde mira un elemento y la jerarquía de huesos que es, de hecho, de necesaria definición para la modificación de las poses mediante cinemática.

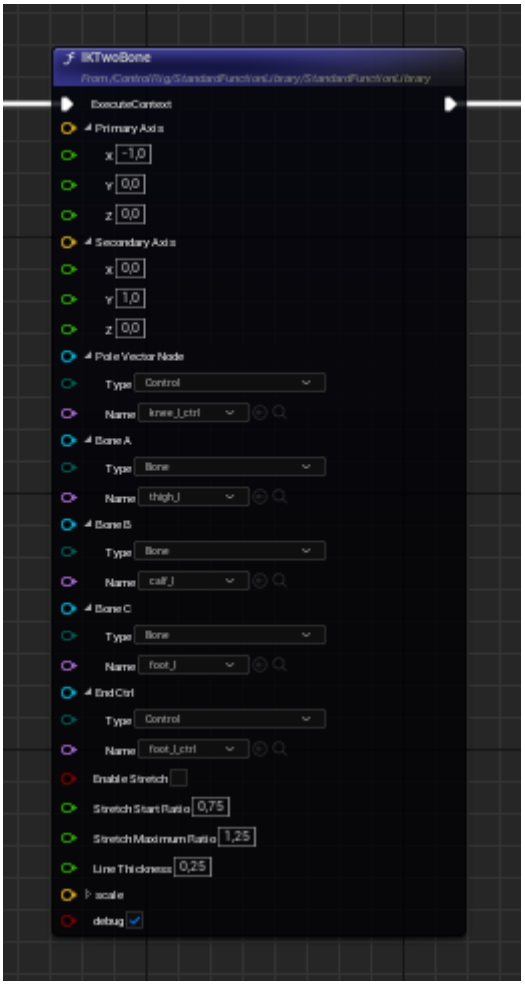

Figura 3.4. Configuración de cinemática inversa para dos huesos mediante Control Rig

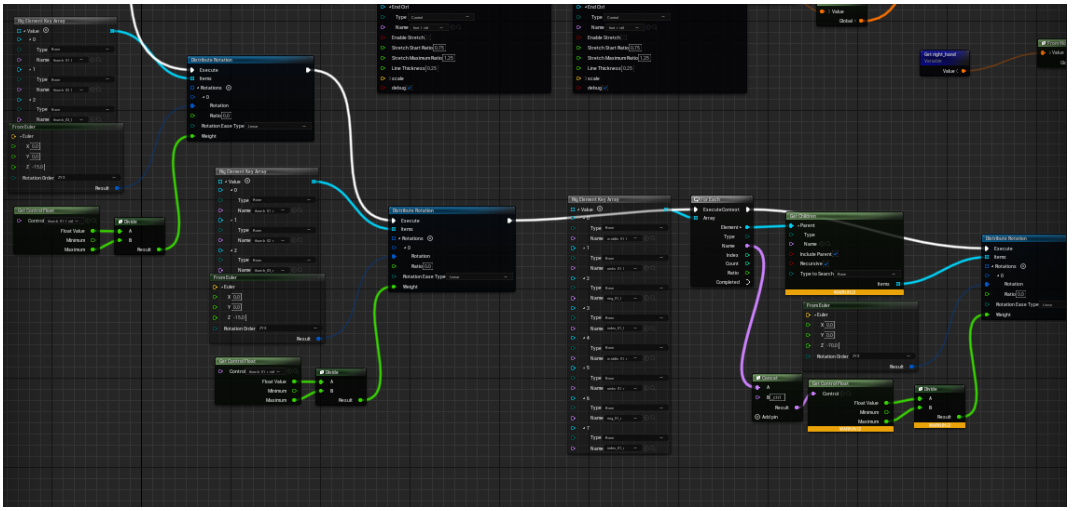

Figura 3.5. Configuración de cinemática inversa para una cadena de más dos huesos mediante Control Rig

3.2.2. Inconvenientes del Control Rig

Precisamente debido a la preparación requerida para el desempeño de todas sus funciones, el Control Rig requiere de un mayor tiempo de configuración y de conocimiento de la misma. Los objetivos para la cinemática inversa son, además, locales al personaje, lo que nos dificulta la conversión de la posición a la hora de interactuar con elementos del nivel.

Además, las animaciones creadas por Control Rig deben ser configuradas por completo mediante secuencias de animación o mediante instrucciones (ya sea por blueprints o codificadas). Es decir, que para crear una animación que interactúe en tiempo real con un elemento, la animación ha de ser definida, hueso a hueso (u objetivo a objetivo) en tiempo de ejecución.

3.3. IK Rig

El IK Rig (de Inverse Kinematic Rig o Rig de Cinemática Inversa) [Figura 3.6 y Figura 3.7], es la forma directa para asignar objetivos a los huesos de un rig ya existente.

Figura 3.6. Previsualización del IK Rig. Los objetivos de la cinemática inversa se resaltan como cubos de aristas amarillas.

3.3.1. Ventajas del IK Rig

El IK Rig es mucho más sencillo de definir, sólo requiere de la asignación a un solver de un tipo, un objetivo y un hueso.

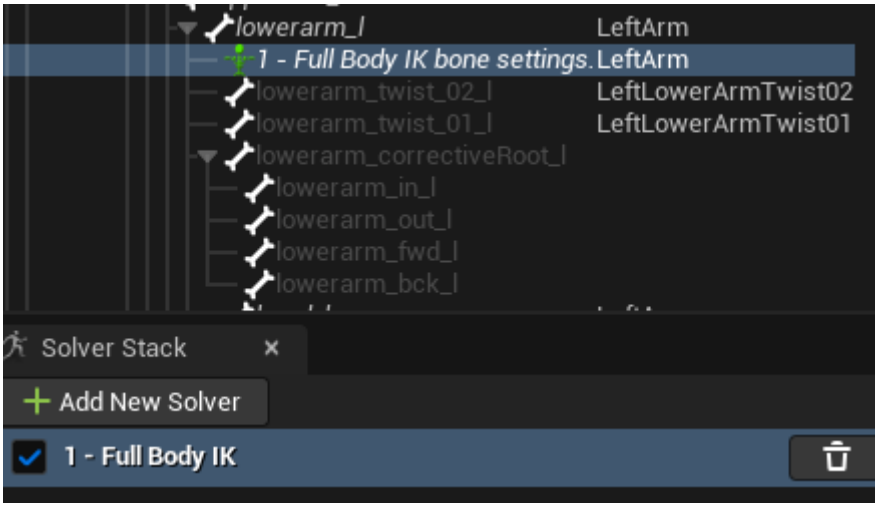

Figura 3.7. Configuración del IK Rig

Podemos utilizar cinco tipos de solvers: Full Body IK, Limb IK, Body Mover, Pole Solver y Set Transform. Cada uno de ellos nos proporciona un funcionamiento diferente. Con el fin de cumplir nuestro objetivo, utilizaremos principalmente el Limb IK, preparado concretamente para extremidades.

Además, los objetivos permiten definirse de forma local en relación a la jerarquía, local en relación al personaje o en coordenadas de mundo; así como seleccionar de qué elementos de cada objetivo queremos permitir modificación [Figura 3.8].

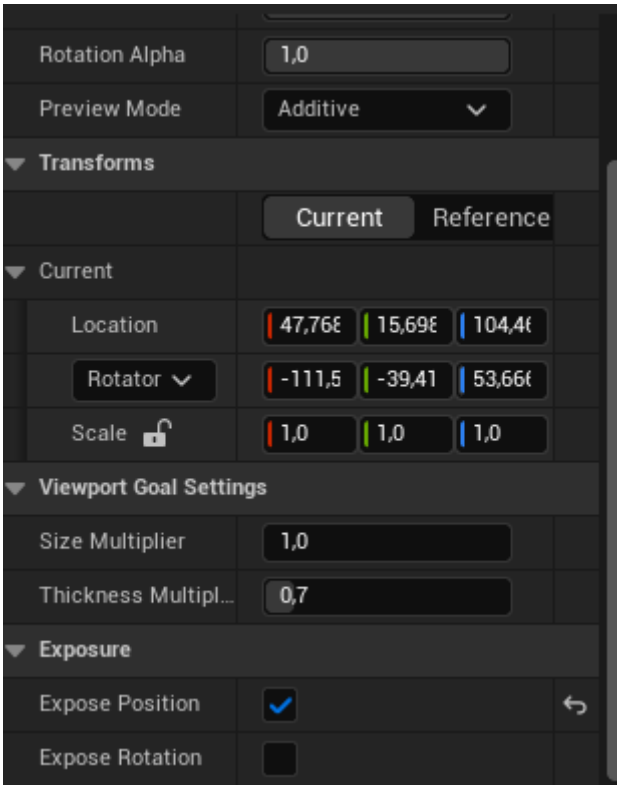

Figura 3.8. Configuración de un objetivo en el IK Rig

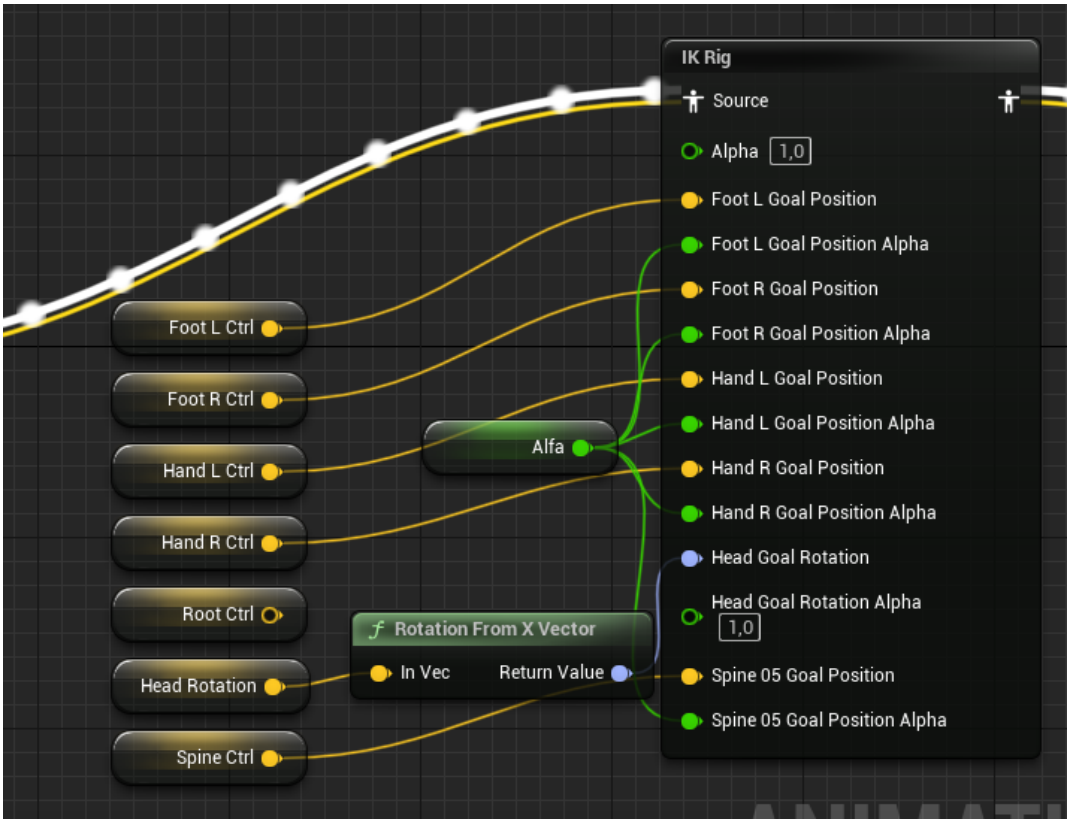

Figura 3.9. Empleo de un IK Rig desde el Animation Blueprint

El IK Rig requiere, dentro del Animation Blueprint [Figura 3.9], de un elemento denominado alpha, que indica qué porcentaje de modificación del movimiento del objetivo ha de ser modificado en el frame actual. Por ello, el alpha se configura como una curva, que permite avanzar hasta un estado y regresar (por ejemplo para añadir suspensión a un salto), o como una recta para mayor simplicidad.

3.3.2. Inconvenientes del IK Rig

El IK Rig aparece a partir de la versión 5.1 de Unreal Engine, que es en la que se ha desarrollado el trabajo. Sin embargo, en esta versión aún cuenta con diversos errores que dificultan su trabajo: al añadir un nuevo solver, los objetivos ya existentes se modifican para atribuirse al nuevo solver y se desordenan en la jerarquía; además de que las rotaciones no permiten modificaciones mediante

coordenadas, sino sólo mediante rotators, lo que tiene en cuenta el estado anterior de la rotación de la articulación en cuestión. Por tanto, las rotaciones para modificar animaciones mediante IK Rig requieren de un cálculo más avanzado, al menos en la versión 5.1.

3.4. Implementación

En nuestro caso, intentamos en un principio desarrollar las animaciones de escalada mediante un Control Rig haciendo uso del modelo de personaje de Quinn que nos provee el Starter Content de Unreal Engine 5. Tras configurarlo, creamos una secuencia de idle (o posición de descanso) en escalada [Figura 3.10]. Sin embargo, al intentar implementar la animación en el Animation Blueprint, no nos permitía modificar correctamente los objetivos de cinemática inversa. Tras buscar una posible solución, se decidió utilizar la herramienta de IK Rig, que permitía adaptar los objetivos a elementos del mundo. A pesar de permitir este reto, después intentamos girar la cabeza del personaje en la dirección de la próxima piedra como ya habíamos hecho con el Control Rig, pero el inconveniente de rotaciones explicado en el apartado anterior (3.3.2), nos lo impidió. Tras resignarnos a no poder mover la cabeza adecuadamente en esta versión, implementamos en el Animation Blueprint la operación que nos permitía modificar la secuencia creada con el Control Rig [Figura 3.11]. De esta forma conseguimos entender el objetivo del alpha (descrito en el apartado 3.3.1), ya que sin él el objetivo de cinemática inversa alcanzaba su posición de forma repentina, por lo que creamos una curva de Timeline en el Character Component para realizar una transición paulatina. Con el fin de poder comprobar las animaciones, durante las pruebas también definimos los controles del personaje durante este proceso. Finalmente, adaptamos el proyecto al juego indicado en el GDD (apartado 2).

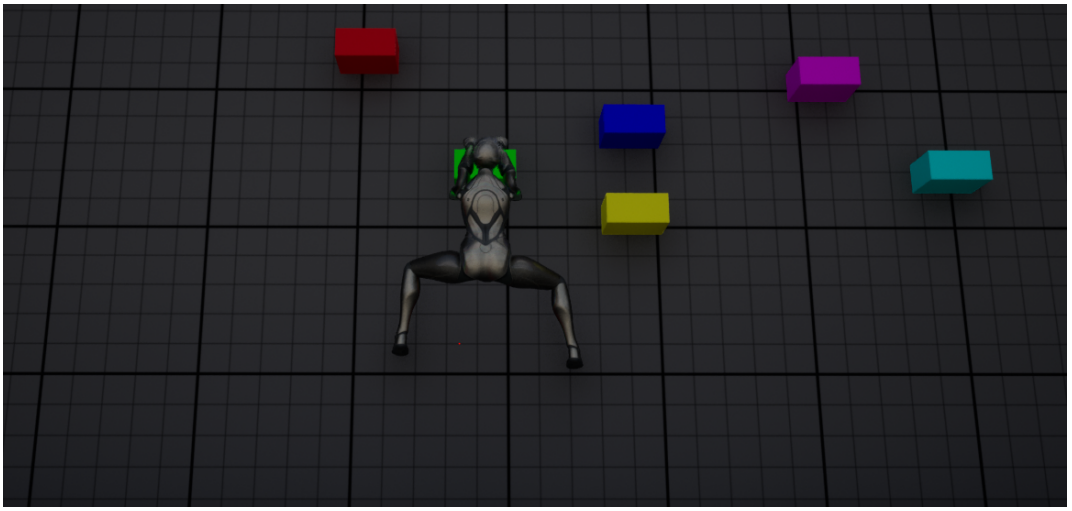

Figura 3.10. Personaje en animación de idle de escalada

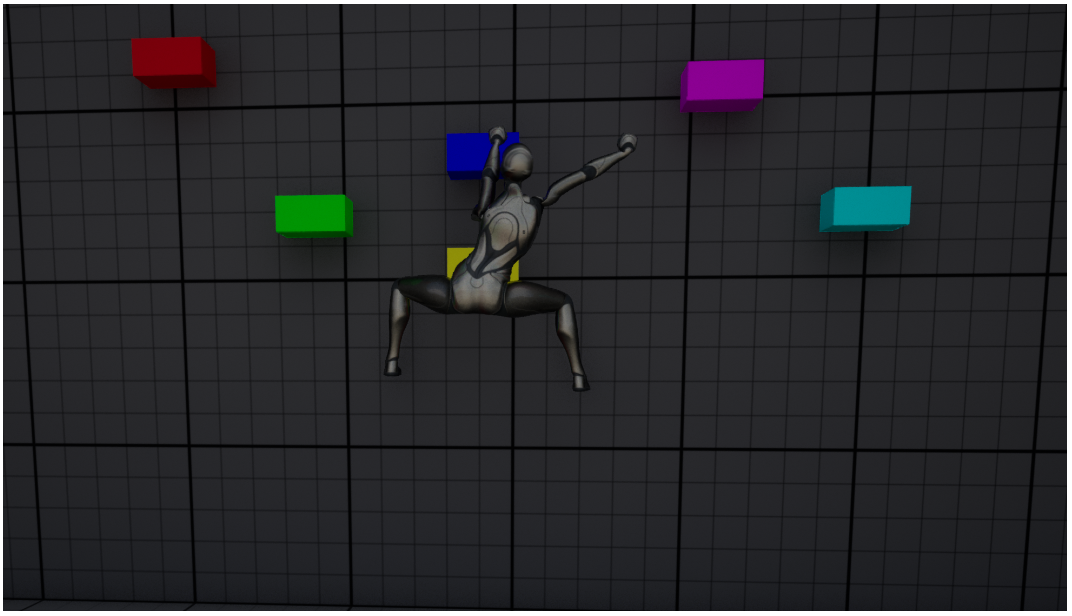

Figura 3.11. Personaje con objetivo de cinemática inversa modificado

Conclusiones y Líneas futuras

Tras analizar diferentes videojuegos y su tipo de animaciones, podemos dilucidar que, si bien en algunos juegos 2D podrían beneficiarse de las animaciones procedurales, se hacen prácticamente necesarios en juegos 3D que cumplan algunas condiciones en el caso de no querer sobrecargar a sus animadores:

- Que el personaje pueda subir o bajar escaleras, ya sean de mano o de escalones. En el primer caso, de no utilizar animaciones procedurales, las manos no agarrarían los travesaños, y en el segundo los pies flotarían en el aire, lo que elimina el realismo y puede producir que el jugador pierda su inmersión.
- Que se produzca en algún momento una escalada por una pared desigual, esto es, que el muro no parezca completamente formado por rocas a las que el personaje pueda asirse, sino que tenga que buscar diferentes salientes limitados.

Figura 4.1. Zelda Breath of the Wild. Link escala por una pared uniforme.

- Que el personaje interactúe con objetos, ya sea recogiéndolos de una mesa, del suelo,.. o presionando botones o palancas.

A pesar de lo que pueda parecer por su nombre, las animaciones procedurales no eliminan la necesidad de animadores, sino que ponen un mayor foco en las necesidades generales de las secuencias, así como en una correcta fusión (blending) de las mismas. Esto es demostrable con el trabajo realizado, ya que utiliza animaciones sencillas o incluidas con el contenido inicial de Unreal Engine 5, lo que da lugar a un movimiento poco realista pero con una clara intención. En este apartado es donde más he notado un déficit de especialización, que podría haber dado lugar a un resultado más profesional.

En cuanto al futuro de las animaciones procedurales, la industria señala en la dirección de la inteligencia artificial, aunque no he podido comprobar de primera mano los resultados de la misma en este sector. Eso sí, si se corresponde con los avances de esta tecnología en otros ámbitos, creo que aunque aún le queda algo de recorrido, sí que podrá convertirse en el estado del arte en menos de una década. Por ahora, nos tendremos que conformar con la cinemática inversa, que a pesar de requerir de más trabajo, da lugar a un gran resultado. Por otro lado, vistas las herramientas proporcionadas por Unreal Engine 5.1, espero que en alguna versión futura, si es que no está ya incluido en Unreal Engine 5.2 (que no he comprobado por no reiniciar el trabajo), proporcione la posibilidad de utilizar una metodología con lo mejor del Control Rig y del IK Rig juntos.

Summary and Conclusions

With the analysis of several videogames and its animations at hand, we can conclude that, even though some 2D games could benefit from procedural animations, they are almost necessary when developing 3D games that have any of this conditions without work-overloading their animators:

- The character can go upstairs or downstairs or even ladder-climbing. In the case of stairs, the feet would be floating mid-air, and in the case of ladders, the hands would not grab them. Both cases would subtract realism and could even remove the immersion of the player.
- The character is allowed to climb a non-uniform wall, which means that the wall does not seem to be formed by a set of equal rocks that the character could grab, but by a limited number of protruding stones for him to climb.

Figura 4.1. Zelda Breath of the Wild. Link escala por una pared uniforme.

- The character must interact with several items like buttons or levers or even grab them from a table, the floor,...

Even taking into account what the name could suggest, procedural animations do not remove the requirements of animators, but focus them into making adaptable sequences and blending them. This is made clear in the assignment, which uses simple animations or those included in the starter content of Unreal Engine 5, producing a low-reality animation set with a clear intent. This is where I felt the biggest specialization deficit that could have resulted in a more professional product.

In terms of the procedural animations future, the industry points in the direction of the artificial intelligence, but I have not been able to test its results in this sector first-hand. Nonetheless, if it has the same expansion as the AI in other fields, I think it will require more time to get better, but it will become the state of art in less than a decade. For now, we will have to settle with inverse kinematics, that achieves better results even though it requires more work. Furthermore, after having seen the tools provided by Unreal Engine 5.1 and without the time to test the upgrades of Unreal Engine 5.2, I hope for a mix of Control Rig and IK Rig that contains the advantages of both of them.

Bibliografía

[B1] **Evans Bohl**. *How to Animate Characters in UE5*. Subido el 20 de mayo de 2022 en https://www.youtube.com/watch?v=w9mijf-gKOg&ab_channel=EvansBohl.

[B2] **CodeLikeMe**. *Unreal Zelda Climbing System - Part 1 (Detect and Grab Wall)*. Subido el 23 de junio de 2021 en https://www.youtube.com/watch?v=MLINAJt3lbs&t=1384s&ab_channel=CodeLikeMe.

[B3] **Rakiz Farooq**. *Retargeting animations in UE5 with IKRig and Full Body IK setup [Uefy v2.5]*. Subido el 30 de mayo de 2022 a https://www.youtube.com/watch?app=desktop&v=mgxGM08bCkA&ab_channel=RakizFarooq.

[B4] **Matt Aspland**. *Animation Retargeting In UE5 | New IK Rig Retargeting System (Tutorial)*. Subido el 6 de mayo de 2022 a https://www.youtube.com/watch?v=N7WdyAeeDrw&ab_channel=MattAspland.

[B5] **Unreal DevOP**. *Unreal Engine - How to use Hand IK to Press Buttons and Pull Levers*. Subido el 25 de enero de 2022 a https://www.youtube.com/watch?v=BewndphQ7aU&ab_channel=UnrealDevOP.

[B6] **Unreal Engine Documentation**. *IK Rig Editor*. Recuperado en octubre de 2022 de https://docs.unrealengine.com/5.0/en-US/ik-rig-in-unreal-engine/.

[B7] **Unreal Engine Documentation**. *Full-Body IK*. Recuperado en octubre de 2022 de https://docs.unrealengine.com/5.0/en-US/control-rig-full-body-ik-in-unreal-engine/

[B8] **Unreal Engine Documentation**. *Control Rig Editor*. Recuperado en octubre de 2022 de https://docs.unrealengine.com/5.0/en-US/control-rig-editor-in-unreal-engine/

[B9] **GDC**. *Animation Bootcamp: An Indie Approach to Procedural Animation*. Subido el 21 de octubre de 2017 a https://www.youtube.com/watch?v=LNidsMesxSE&t=420s&ab_channel=GDC

[B10] **Wikipedia**. *Mastermind*. Recuperado en julio de 2023 de https://www.youtube.com/watch?v=LNidsMesxSE&t=420s&ab_channel=GDC