# Creación de Herramientas de Apoyo por Animación Procedural en Unity: Animath

Luisa Fernanda Arias Pachon y Daniel Felipe Aldana Martínez

Universidad San Buenaventura, Bogotá DC, Colombia

Universidad San Buenaventura

Facultad de ingeniería (Bogotá)

Ingeniería Multimedia

Director: Rafael Alberto Reyes

Bogotá DC, Colombia

2023

# Creación de Herramientas de Apoyo por Animación Procedural en Unity: Animath

Luisa Fernanda Arias Pachon y Daniel Felipe Aldana Martínez

Director: Ing. Rafael Alberto Reyes

Trabajo de grado para optar por el título de: Ingeniero Multimedia

Universidad San Buenaventura

Bogotá, 2023.

# Resumen

Esta tesis presenta un marco teórico acerca de las técnicas utilizadas para la creación y producción de un plug-in que hace uso de la animación procedural, agilizando así los tiempos de producción de videojuegos en la plataforma Unity. Para esto, se plantea el uso de ejemplos con distintos movimientos, en este caso, ciclos de caminado con fotogramas clave o captura del movimiento (Keyframes). Se tiene en cuenta como parámetros principales la amplitud y velocidad del movimiento, los cuales ajustan los huesos del personaje, generando así una variación en su movimiento, lo cual le da una característica única al personaje en el desarrollo del videojuego.

**Palabras clave:** Animación, Procedural, Locomoción, Unity, Plug-in, Trigonométrica.

Contenido

[Creación de Herramientas de Apoyo por Animación Procedural en Unity: Animath 1](#_Toc130410583)

[Creación de Herramientas de Apoyo por Animación Procedural en Unity: Animath 2](#_Toc130410584)

[Resumen 3](#_Toc130410585)

[1. Introducción 6](#_Toc130410586)

[1.1. Antecedentes 7](#_Toc130410587)

[1.2. Justificación 10](#_Toc130410588)

[1.2.1. Problema: 11](#_Toc130410589)

[1.2.2. Hipótesis: 11](#_Toc130410590)

[1.2.3. Método: 12](#_Toc130410591)

[1.2.4. Alcances y Limitaciones: 12](#_Toc130410592)

[1.3. Objetivos: 13](#_Toc130410593)

[1.3.1. Objetivo General: 13](#_Toc130410594)

[1.3.2. Objetivos Específicos: 13](#_Toc130410595)

[1.4. Marco Teórico: 14](#_Toc130410596)

[1.4.1. Animación: 14](#_Toc130410597)

[1.4.2. Animación en los Videojuegos: 16](#_Toc130410598)

[1.4.3. Animación Procedural o Procedimental: 17](#_Toc130410599)

[1.4.4. Locomoción: 19](#_Toc130410600)

[1.4.5. Cinética Inversa (IK): 20](#_Toc130410601)

[1.4.5. Funciones trigonométricas en la animación: 21](#_Toc130410602)

[1.4.6. Unity: 22](#_Toc130410603)

[1.4.7. Rigging en el modelo 3D: 23](#_Toc130410604)

[2. Metodología 24](#_Toc130410605)

[2.1. Análisis inicial: 25](#_Toc130410606)

[2.2. Fase de Preproducción: 25](#_Toc130410607)

[2.3. Fase de Prototipado: 26](#_Toc130410608)

[2.4. Fase de Producción: 26](#_Toc130410609)

[2.5. Fase de Pruebas y Correcciones: 26](#_Toc130410610)

[3. Desarrollo de Animath 27](#_Toc130410611)

[3.1. Pasos iniciales: 27](#_Toc130410612)

[3.2. Set-Up de Animath: 27](#_Toc130410613)

[3.3. Funciones trigonométricas: 28](#_Toc130410614)

[3.4. Funcionamiento de la función del seno: 29](#_Toc130410615)

[3.4.1. Amplitud: 29](#_Toc130410616)

[3.4.2. Velocidad: 30](#_Toc130410617)

[3.4.3. Punto Inicial: 31](#_Toc130410618)

[3.4.4. Desface: 32](#_Toc130410619)

[3.5. Animaciones: 32](#_Toc130410620)

[4. Conclusiones 38](#_Toc130410621)

[Tabla de figuras 40](#_Toc130410622)

[Lista de referencias 41](#_Toc130410623)

# Introducción

En la industria, el mercado de los videojuegos se ha expandido a través del tiempo haciendo que la producción de estos sea necesaria para la actualidad. La demanda de estos, ha hecho que sea necesaria la implementación de entornos y personajes realistas, que permitan cierto detalle al momento de realizar movimientos o de interactuar con el entorno. Si bien se sabe, la representación gráfica de los personajes ha alcanzado altos estándares en cuanto a producción, procesos como la animación, pierden calidad en el proceso debido a los largos tiempos a los que debe someterse todo el proceso desde el modelado del personaje, pasando por su etapa de texturizado, riggin y finalmente la animación, consumiendo así bastantes recursos y frenando de cierta forma los estándares de calidad en cuanto a realismo de movimiento que se quiere llegar.

Suelen presentarse casos en los que animar un ciclo de caminado completo para determinado personaje, se presenta como una tarea compleja, esto debido a problemas con la cinemática utilizada para lograr dichos movimientos de rotación en pies, rodillas y demás huesos fundamentales. Estos movimientos suelen crearse a partir de una captura de movimientos manual, en la que se anima cada segundo de movimiento para lograr una animación sintetizada. Esto puede ser problemático para desarrolladores con recursos limitados, debido a que, si se quiere realizar un ciclo de caminado con alguna variación, por ejemplo, en velocidad, se debería volver a rehacer el proceso de animación desde su fase inicial, por esto mismo se pueden ver afectados desarrolladores grandes, con alta cantidad de recursos, ya que también representa un gasto en los mismos.

Para solucionar esto, se necesitan métodos que puedan generalizar y ajustar estos cambios de movimiento haciendo que el proceso de animación sea más ágil y consistente. Dado que los ciclos de caminado son los más utilizados para el desarrollo y creación de videojuegos, este movimiento es el punto central de esta tesis.

Con el pasar del tiempo, se han propuesto métodos que permitan la locomoción sin generar movimientos de forma exagerada y de apariencia natural. Hablamos entonces de métodos procedurales que permiten un movimiento basado en algoritmos, es decir, el movimiento no está creado manualmente por los artistas de animación que tienen los equipos de desarrollo de videojuegos; el desarrollador es capaz de darle una personalidad y estilo al personaje según las características en sus movimientos. El objetivo de esta tesis en generar un sistema de animación procedural que pueda usarse para darle realismo al movimiento de los personajes y a su vez, para optimizar el proceso de animación mediante la configuración de parámetros mínimos.

## Antecedentes

En la siguiente lista se condensan una serie de referencias tomadas a la hora de la elaboración de este proyecto, con respecto a estas se tomó gran parte de la inspiración inicial que permitieron llevar este proyecto acabo. Gracias a estos proyectos, se entiende que la animación procedural, se ha convertido en una de las herramientas más importantes, siendo esta utilizada para distintos fines en producciones como:

* "The Wave" de Aurelien Alvarez: Este proyecto utiliza funciones trigonométricas para generar una animación de onda en tiempo real en una instalación de arte interactiva. (Institut Henri Poincaré, 2014)
* "Kinematics" de Daniel Shiffman: que utiliza funciones trigonométricas para simular la cinemática inversa de un brazo robótico en un ambiente virtual en 3D.
* "Modelo de una Mano Humana Animada Mediante Unity3D y Sensores Inerciales" de la Universidad de Antioquia: que utilizó funciones trigonométricas para generar una animación de la mano en tiempo real utilizando sensores inerciales. (The Coding Train, 2017)
* “Unreal Engine 4 procedural Animation Toolkit” de Lionade Games: Este proyecto utiliza funciones trigonométricas para animar modelos 3D de forma procedural en Unreal Engine 4. Utiliza un sistema de nodos para definir las animaciones y permite una gran flexibilidad en la creación de animaciones procedurales. (Sneaky Kitty Game Dev, 2022)
* “Procedural animation in Blender” de Simon Thommes: Este proyecto utiliza funciones trigonométricas para crear animaciones procedurales en Blender, un software de modelado 3D similar a Unity. Utiliza la función seno para crear movimientos repetitivos y fluidos, como la oscilación de un péndulo. (Cleverpoly, 2022)
* “Modelo de animación procedural de árboles en Unity” de Rafael A. Ferreira: Este proyecto utiliza funciones trigonométricas para animar modelos 3D de árboles de forma procedural en Unity. Utiliza la función seno para crear movimientos oscilatorios en las ramas y hojas de los árboles. (game3Dover, 2016)
* “Oscillations” de la Universidad de Stanford: es un proyecto que utiliza funciones trigonométricas para crear animaciones de ondas y oscilaciones. La herramienta permite a los usuarios ajustar los parámetros de las funciones trigonométricas para crear diferentes patrones de movimiento. (Maything, 2020)
* “Procedural Animation using Trigonometric Functions in Unity” de YouTube: es un tutorial que explica cómo utilizar funciones trigonométricas para crear animaciones procedimentales en Unity. El tutorial cubre los conceptos básicos de las funciones seno y coseno, y muestra cómo utilizar estas funciones para crear animaciones de ondas y oscilaciones. (Codeer, 2020)
* “Procedural animation for humans in Unreal Engine 5” de Udemy Academy: es un curso en línea que enseña cómo utilizar animación procedimental en Unreal 5 utilizando funciones trigonométricas. El curso cubre conceptos como la interpolación lineal, la animación de ciclos y el control de animaciones procedimentales mediante el uso de parámetros. (Margison, 2022)
* “Trigonometric Animation in Unity” de GitHub: es un proyecto de código abierto que utiliza funciones trigonométricas para animar modelos 3D en Unity. El proyecto incluye una variedad de ejemplos de animaciones procedimentales, incluyendo animaciones de ondas y oscilaciones, y está disponible para su descarga gratuita. (SebLague, n.d.)

## 1.2. Justificación

Los videojuegos en la era actual requieren animaciones realistas para que el personaje pueda involucrarse en el entorno de forma efectiva y ágil. Se necesita optimizar el proceso de animación de los personajes para ahorrar recursos en la creación de los videojuegos, esto tanto para animadores como para desarrolladores. Para que esto sea posible, es necesaria la existencia de una compatibilidad entre el proceso de animación y el flujo de trabajo existente, si bien es innovador el uso de un algoritmo que permita solucionar la problemática mediante eventos procedurales, es importante entender que el uso de keyframes o capturas de movimiento hechas a mano, permite un total control de la animación que se va a realizar, por esto mismo es necesario que el plug-in cuente con controles de movimiento para que el arte no se vea afectado.

La animación procedural es una técnica utilizada en la industria de los videojuegos y la animación para crear movimientos orgánicos y realistas en los personajes y objetos 3D. Esta técnica se utiliza para reducir el trabajo de animación manual, ahorrar tiempo y crear animaciones más realistas. Un script en Unity que use funciones trigonométricas para animar un modelo 3D usando animación procedural sería una excelente opción para crear una animación de movimiento natural y realista. Por ejemplo, se podría animar la caída de hojas de un árbol en una simulación realista usando senos y cosenos para simular el movimiento de la brisa y el movimiento de las hojas. Además, las funciones trigonométricas también pueden utilizarse para animar personajes en juegos de lucha, como animar golpes de puño y patadas, lo que daría una sensación de movimiento más fluido y natural.

En resumen, el uso de funciones trigonométricas para animar un modelo 3D en Unity usando animación procedural puede ayudar a crear animaciones realistas y orgánicas en menos tiempo que el proceso de animación manual, lo que a su vez, puede mejorar la calidad del juego o la animación.

### 1.2.1. Problema:

Creación de una herramienta de apoyo para industria de los videojuegos, la cual, haciendo uso de animación procedural pueda solventar problemáticas de tardanza en los procesos de animación, siendo esta totalmente controlable por el usuario final.

### 1.2.2. Hipótesis:

Mediante el uso de algoritmos procedurales, es posible lograr animaciones cíclicas que pueden agilizar el proceso de desarrollo de un videojuego sin perder calidad en las animaciones podría generar movimientos realistas y fluidos para el personaje, lo que mejoraría significativamente la calidad de la animación. La animación procedimental basada en funciones trigonométricas podría ser especialmente útil para generar movimientos naturales y fluidos, como la animación de ondas de agua, la animación de hojas de árboles movidas por el viento, la animación de una multitud de personajes caminando, entre otros. La precisión y control que ofrecen las funciones trigonométricas permitiría generar animaciones complejas de forma más eficiente y autónoma, lo que ahorraría tiempo y recursos en el proceso de producción. Además, la combinación de funciones trigonométricas con otras técnicas de animación procedural, como la cinemática inversa (IK) o la simulación de física, podría permitir generar animaciones aún más realistas y complejas, lo que mejoraría la experiencia del usuario y la calidad del producto final.

### 1.2.3. Método:

Se identifican keyframes o momentos claves en el movimiento, otros aspectos de estilo y funciones trigonométricas clave para la naturalidad del movimiento en un ciclo de caminado y sus variaciones. Se evalúa la naturalidad del movimiento, así como la flexibilidad en los controles para la variación del mismo.

### 1.2.4. Alcances y Limitaciones: 3D Character Rigging: Joint placement reference | Animation reference, Character rigging, Reference

#### Figura 1: Limitación en cuanto a la cantidad de huesos del personaje.

En cuanto a limitaciones, esta herramienta, solo funciona específicamente para la plataforma de desarrollo de videojuegos UNITY, además de esto solo funciona con personajes de dos extremidades inferiores (bípedos) y con misma cantidad de huesos que se muestra en la Figura 1. Además, como limitación, el desarrollo no puede grabar o guardar las configuraciones echas con el plug-in para futuros proyectos. En cuanto a los alcances de esta herramienta, se tiene total control de la velocidad y la amplitud del movimiento y un desarrollador puede añadir más animaciones si lo considera necesario.

## 1.3. Objetivos:

### 1.3.1. Objetivo General:

Diseñar e implementar un plug-in para la plataforma Unity, que permita la animación procedural para personajes bípedos de un ciclo de caminado y algunas variaciones de este ciclo.

### 1.3.2. Objetivos Específicos:

* Identificar características y requerimientos que puedan implementarse dentro de la herramienta y que faciliten el proceso de animación.
* Diseñar un sistema que permita la modificación de parámetros de animación mediante su función trigonométrica.
* Identificar variaciones en el ciclo de camino que puedan darle un estilo propio y personalidad al personaje que se esté animando.

## 1.4. Marco Teórico:

A continuación, se presenta una descripción general de términos y palabras claves que son necesarios para el entendimiento de este proyecto.

### 1.4.1. Animación:

La animación es el proceso de crear la ilusión de movimiento al mostrar una serie de imágenes estáticas en rápida sucesión. Estas imágenes, llamadas cuadros o fotogramas, se muestran en una secuencia que da la impresión de que los objetos o personajes están en movimiento. En las películas de figuras animadas, sistema para desarrollar los movimientos de los personajes o de los objetos y elementos («Animación», s. f.). Las primeras aplicaciones de los gráficos computacionales, se utilizaban principalmente en ámbitos militares o para ciencias aplicadas, como por ejemplo simuladores de vuelos de avión que servían para entrenar pilotos sin poner en riesgo la vida u otros recursos como aviones reales. (Garza Mireles, 2014). Con el pasar de los años, se ha experimentado con varios métodos de animación que permite una compatibilidad con las nuevas tecnologías que han surgido a través de los años; hablamos entonces de métodos como el *hand-drown animation, stop motion* y, por último, la animación realizada a computador, método del cual se hace mayor uso hoy en día. Suele considerarse que estas técnicas de movimiento pertenecen exclusivamente a la industria cinematográfica, sin embargo, esto ha dado paso a volverse pieza fundamental en el desarrollo de videojuegos, convirtiéndose en un pilar fundamental de estos.

En la animación, se crean una serie de dibujos o imágenes, cada uno ligeramente diferente al anterior, y se reproducen en rápida sucesión para crear la ilusión de movimiento. Esta técnica se conoce como animación de imágenes clave o animación por fotogramas clave, lo que llamamos “*Keyframe*”. Los keyframes son los puntos de referencia en una secuencia de animación que establecen la posición, la escala, la rotación y otros atributos de un objeto o personaje. Los fotogramas intermedios se crean para unir las imágenes clave y su número y frecuencia determinan la suavidad y la velocidad de la animación.

La animación se puede realizar de forma tradicional, dibujando cada fotograma a mano, o utilizando técnicas digitales, como la animación por ordenador. La animación por ordenador utiliza software especializado que permite a los animadores crear personajes y objetos en un entorno digital y animarlos mediante el ajuste de los parámetros de la animación, como la posición, la rotación y la escala. (Thomas & Johnston, 1995).

La animación se utiliza para una variedad de pr­opósitos, como contar historias, educar, entretener y publicitar productos y servicios. En la industria del cine y la televisión, se utiliza para crear películas de animación y efectos visuales para películas en imagen real. En la industria de los videojuegos, se utiliza para crear personajes y entornos interactivos para los jugadores. En publicidad, se utiliza para crear anuncios animados y promociones de productos y servicios.

En resumen, la animación es una técnica artística que utiliza una serie de imágenes fijas para crear la ilusión de movimiento. Se utiliza en una variedad de medios para contar historias, educar, entretener y publicitar productos y servicios. La animación puede realizarse de forma tradicional o mediante técnicas digitales, y es una herramienta poderosa para crear contenido visualmente atractivo e impactante. (Hernández Rielo, s. f.).

### 1.4.2. Animación en los Videojuegos:

La animación en los videojuegos es un componente crucial para crear una experiencia de juego inmersiva. Se utiliza para dar vida a los personajes, objetos y entornos del juego. La animación en los videojuegos se centra en la interacción en tiempo real entre los personajes y los objetos en el mundo virtual. A pesar de decirse que la animación cinematográfica y la animación para videojuegos se resumen en los mismos procesos, hay algunas diferencias que es importante remarcar: En la animación cinematográfica, el usuario o espectador, no está en control de lo que ve sobre la pantalla, es decir, su punto de vista es inmóvil y no interactúa directamente con el audiovisual que se le está mostrando. Por otro lado, los videojuegos deben ser interactivos, esto quiere decir que el usuario final o jugador, está en constante control de lo que visualiza en la pantalla, así como los movimientos de los personajes y la interacción que tienen estos con el entono, por lo tanto, la cantidad de detalle que requiere un videojuego en cuanto a animación es mayor, esto tiene en cuenta no solo el punto de vista del jugador sino también las interacciones y eventos consecuentes a las acciones que este realice.

La animación en los videojuegos se ha desarrollado significativamente a lo largo de los años, desde las técnicas manuales de animación de *sprites* hasta técnicas digitales avanzadas como la animación por captura de movimiento y la animación en tiempo real. Las técnicas de animación en los videojuegos implican la utilización de herramientas y software especializados, como software de animación, motores de juego y herramientas de captura de movimiento.

La animación en los videojuegos también es un elemento importante en la narrativa del juego, ya que ayuda a contar la historia y desarrollar los personajes. Las animaciones de los personajes pueden proporcionar pistas visuales sobre su personalidad, motivaciones y emociones, lo que puede hacer que los jugadores se sientan más conectados con ellos y con la historia del juego.

En resumen, la animación en los videojuegos es un componente crucial para crear una experiencia de juego inmersiva y atractiva. Desde los movimientos básicos de los personajes hasta las escenas cinemáticas, la animación en los videojuegos juega un papel fundamental en la creación de un mundo virtual creíble y emocionante para los jugadores. (Solarski, 2012).

### 1.4.3. Animación Procedural o Procedimental:

La animación procedural es un método de animación que utiliza algoritmos y reglas para generar animaciones en lugar de crearlas manualmente. Este método se utiliza para crear animaciones complejas que serían muy difíciles o imposibles de lograr con la animación tradicional. Al hablar de contenidos procedurales, entendemos por esto todo aquel algoritmo que funcione de manera inteligente o con manipulación humana limitada. La razón por la cual se hace uso de estos algoritmos es para reducir el tiempo de desarrollo de contenidos dentro del videojuego (mapas, texturas, animaciones, etc.) y costos del mismo.

Por otra parte, un problema que se presenta de manera frecuente en este tipo de generación procedural, puede ser encontrado en que el contenido generado se puede percibir como genérico, robótica y poco original, así como carentes de personalidad, motivo por el cuál el posible jugador puede perder interés debido a este problema. En adición, cabe mencionar que en ocasiones pueden faltar detalles en la generación de animaciones, que no aportarían mucho al contenido que se busca desarrollar. Sin embargo, en la animación procedimental, los algoritmos se basan en el comportamiento físico y las leyes del movimiento para crear animaciones realistas.

La animación procedural se utiliza en una variedad de campos, como la creación de personajes en videojuegos y la animación de efectos visuales en películas y televisión. Se utiliza para generar movimientos y comportamientos complejos, como la animación de multitudes, la simulación de fenómenos naturales, como el agua y el fuego, y la animación de criaturas y personajes que requieren movimientos realistas y naturales.

La animación procedural se basa en una variedad de técnicas, como la cinemática inversa (IK), la simulación de física, el aprendizaje automático y la inteligencia artificial. También se pueden utilizar técnicas de modelado procedural para crear geometrías y texturas de forma automática.

La animación procedural es especialmente útil para crear animaciones que deben ser generadas de forma repetitiva, como la animación de multitudes de personajes en un videojuego o la animación de hojas y ramas en un árbol. Al utilizar la animación procedural, se pueden crear movimientos y comportamientos únicos y realistas para cada objeto de forma autónoma, sin necesidad de crear manualmente cada movimiento.

En resumen, la animación procedural es una técnica avanzada de animación que utiliza programación y reglas matemáticas para crear movimientos y comportamientos realistas de forma automática. Es una herramienta poderosa para crear animaciones complejas y repetitivas en una variedad de campos, incluyendo videojuegos, películas y televisión. (Short & Adams, 2017).

### 1.4.4. Locomoción:

­ La locomoción es el movimiento de un personaje en un videojuego o animación. Esto puede incluir caminar, correr, saltar, nadar y otras formas de movimiento. La locomoción es importante para la jugabilidad y la inmersión en el mundo virtual. Partiendo de una breve definición, locomoción refiere a un traslado de cierto elemento desde un punto a otro. Para este proyecto, es importante entender que, en el ser humano o cualquier personaje bípedo, la locomoción parte de dos actividades importantes: Caminar y correr. Caminar, para un bípedo, significa el movimiento en los pies cuya ejecución incluye en su línea de tiempo, que ambos pies estén juntos en el suelo en un momento determinado, mientras que el correr, implica que no exista un doble apoyo por ambos pies, lo que significaría que el bípedo tuviera la llamada “fase de vuelo” y esta fuera predominante durante la actividad. (Menard, 2011)

La locomoción puede variar en función del tipo de personaje u objeto que se esté moviendo y del tipo de ambiente en el que se esté moviendo. En el caso de los personajes, la locomoción puede ser desde caminar o correr hasta saltar, nadar o trepar. En los objetos, la locomoción puede ser desde el movimiento básico de un vehículo hasta el movimiento más complejo de un objeto en el espacio.

La animación de locomoción implica el uso de diferentes técnicas y herramientas, como la cinemática inversa (IK), la animación de esqueletos, la simulación de física, la captura de movimiento, entre otros. En algunos casos, se utilizan algoritmos y herramientas de inteligencia artificial para lograr una locomoción más realista.

La locomoción es una parte importante de la experiencia de los jugadores en los videojuegos, ya que afecta directamente la jugabilidad y la inmersión. Una locomoción bien diseñada y ejecutada puede ayudar a los jugadores a sentirse más conectados con los personajes y el entorno del juego, y puede mejorar su experiencia de juego en general. (Biancard et al., 2020).

### 1.4.5. ­­Cinética Inversa (IK):

La cinemática inversa (IK) es una técnica utilizada en animación para controlar los movimientos de un personaje u objeto al manipular los huesos que lo componen. Se utiliza principalmente en animación 3D y es útil para crear movimientos realistas y precisos.  
La cinemática inversa se basa en las leyes de la física y en la geometría para determinar la posición de los puntos de control del esqueleto virtual. Es una técnica muy utilizada en la animación de personajes y en la robótica para controlar el movimiento de objetos de manera más eficiente y precisa. (Spong et al., 2006)

En la animación, la cinemática inversa se utiliza para animar personajes y objetos en función de su posición en el espacio. En lugar de mover manualmente cada parte del cuerpo de un personaje, se utilizan los puntos de control del esqueleto para manipular la posición y orientación de los huesos, logrando un movimiento más realista y fluido. Por ejemplo, para animar el brazo de un personaje para que agarre un objeto, se selecciona el punto de control del objeto y el esqueleto del brazo se ajusta automáticamente para que la mano se posicione en la ubicación correcta .

En la robótica, la cinemática inversa se utiliza para controlar los movimientos de los robots, como los brazos robóticos, para que realicen tareas específicas. Por ejemplo, se puede programar un brazo robótico para que recoja un objeto de una ubicación específica y lo coloque en otra ubicación utilizando la cinemática inversa.

### 1.4.5. Funciones trigonométricas en la animación:

Las funciones trigonométricas son ampliamente utilizadas en la animación para crear movimientos realistas y fluidos en modelos 3D. En particular, las funciones seno y coseno son fundamentales para la creación de animaciones procedurales en Unity y otras plataformas de desarrollo de juegos. (Williams, 2001)

En animación, las funciones trigonométricas se utilizan para crear movimientos suaves y realistas en los objetos animados. Estas funciones permiten crear una variedad de patrones de movimiento que pueden ser controlados por parámetros específicos, como la amplitud, la frecuencia y la fase.

Las funciones trigonométricas más utilizadas en animación son el seno y el coseno, que producen ondas periódicas. Estas ondas pueden ser modificadas y combinadas para producir patrones de movimiento complejos y variados. Además, la mayoría de las funciones de animación en 3D utilizan cuaterniones para representar las rotaciones de los objetos, lo que permite una mayor precisión y control en la animación.

En la animación 3D, las funciones trigonométricas se utilizan para crear animaciones procedurales, lo que significa que los movimientos son generados automáticamente en tiempo real en lugar de ser creados manualmente. Esto permite una mayor variedad y flexibilidad en la animación, ya que los movimientos pueden ser generados de forma aleatoria o basados en ciertos parámetros, como la velocidad o la dirección del movimiento.

### 1.4.6. Unity:

Unity es un motor de videojuegos multiplataforma que permite a los desarrolladores crear videojuegos, aplicaciones interactivas y experiencias de realidad virtual y aumentada para una variedad de plataformas, incluyendo PC, consolas de videojuegos, dispositivos móviles y otros dispositivos. (Unity s.f.).

El programa se basa en un modelo de desarrollo en tiempo real que permite a los desarrolladores trabajar en un entorno visual, creando y modificando activos en tiempo real. Los desarrolladores pueden agregar y manipular objetos en una escena, modificar la iluminación y la cámara, y crear interacciones y comportamientos usando programación.

Unity utiliza el lenguaje de programación C# como su lenguaje de scripting principal, aunque también admite otros lenguajes de programación como JavaScript y Boo. Los desarrolladores pueden usar el editor visual de Unity para crear interfaces de usuario y lógica de juego, o pueden escribir código personalizado para implementar una funcionalidad más avanzada. Una de las principales ventajas de Unity es su amplia comunidad de desarrolladores y el ecosistema de complementos y herramientas que ha surgido alrededor del motor. Unity admite una amplia variedad de complementos y herramientas que pueden ayudar a los desarrolladores a mejorar su flujo de trabajo y agregar nuevas funcionalidades a sus proyectos.

### 1.4.7. Rigging en el modelo 3D:

El rigging de un modelo 3D es el proceso de crear una estructura de huesos y controladores en un modelo 3D para permitir su animación. Este proceso permite al artista de animación manipular el modelo 3D de manera más fácil y natural, lo que puede mejorar la calidad y la eficiencia de la animación. (Mullen, 2011).

El rigging se lleva a cabo mediante software especializado, como Maya o Blender. El proceso comienza con la creación de una malla de polígonos que representa el modelo 3D. Luego, se crean los huesos que se ajustan a la forma del modelo 3D, como si fueran un esqueleto interno. Los huesos se posicionan y ajustan para que se ajusten a la forma del modelo 3D y se les asignan pesos que indican cuánto influyen en la deformación de la malla.

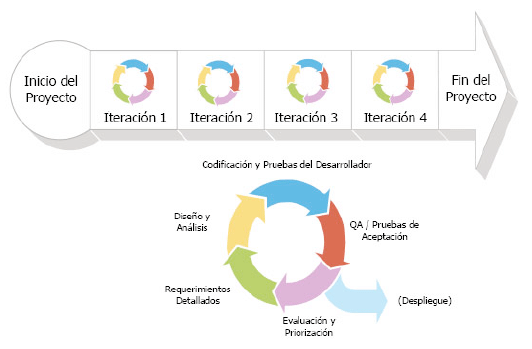
Después de crear la estructura de huesos, se agregan controladores que permiten al animador manipular el modelo 3D de manera más fácil y natural. Los controladores se pueden crear en cualquier forma imaginable, como esferas, cajas o incluso caracteres personalizados.

Por último, se asignan las animaciones a los controladores, lo que permite al animador manipular el modelo 3D para crear la animación deseada. El rigging también puede incluir la creación de sistemas de física, como la simulación de la gravedad, la fricción y la colisión, que pueden mejorar la calidad de la animación y hacerla más realista.

# Metodología

La metodología iterativa incremental es un enfoque que se adapta de manera efectiva a nuestro proyecto. Esta metodología se basa en el desarrollo de iteraciones o fases, que pueden considerarse como pequeños proyectos que avanzan en paralelo y se integran gradualmente para dar forma al producto final. La ventaja de este enfoque es que permite evidenciar el desarrollo del proyecto de manera incremental, realizando pequeñas pruebas en cada iteración finalizada y obteniendo un producto final más completo y pulido que cumpla con los objetivos del desarrollador.

En nuestro caso, implementamos esta metodología en 5 fases, tal como se muestra en el siguiente gráfico. Cada fase se diseñó de manera que se lograra un avance significativo y se pudiera evaluar el progreso del proyecto de forma continua. De esta manera, pudimos abordar de manera eficiente los desafíos y obstáculos que surgieron durante el desarrollo del proyecto, y asegurar que el producto final cumpliera con nuestras expectativas y requerimientos.



#### Figura 2: Modelo de iteración. (Metodología iterativa, s. f.)

## 2.1. Análisis inicial:

El análisis inicial de nuestro proyecto se enfoca en la selección y diseño de las animaciones que serán implementadas en el plug-in. Para ello, se han identificado cinco fases de animación: Inactividad (Idle), Caminar (Walk), Correr (Running), Cansado (Tired) y Zombie (Zombie Walk), con el objetivo de lograr movimientos orgánicos y realistas que mejoren la calidad de la animación.

Una vez seleccionadas las animaciones, se procede a la implementación de un script que contenga funciones para manipular los huesos mediante el uso de senos y cosenos, permitiendo la modificación de estos por parte del desarrollador. Para mejorar la experiencia del usuario, se han agregado manipuladores a modo de slider con títulos descriptivos que facilitan el entendimiento del programa.

La implementación de estas técnicas garantiza una mayor precisión y control en la creación de animaciones, permitiendo obtener resultados de alta calidad que cumplan con los objetivos del proyecto. Es importante destacar que la creación de movimientos realistas es un aspecto clave en la animación y, por lo tanto, se ha puesto un énfasis especial en su logro a través de la implementación de senos y cosenos en el script procedimiento e implementación.

## 2.2. Fase de Preproducción:

En esta fase, se definirían los objetivos y requisitos del proyecto, se diseñaría el modelo 3D, se planificarían las animaciones y se establecerían los plazos y recursos necesarios.

## 2.3. Fase de Prototipado:

En esta fase se crearía un prototipo simple del proyecto, para probar su funcionalidad y ajustar los requisitos si es necesario. En este caso, se podrían crear animaciones simples utilizando funciones trigonométricas para probar su efectividad en el modelo 3D.

## 2.4. Fase de Producción:

En esta fase se crearían las animaciones finales utilizando funciones trigonométricas. Se definirían los joints y el rigging necesarios para el modelo 3D, y se crearían las funciones de animación procedural necesarias para lograr los efectos deseados.

## 2.5. Fase de Pruebas y Correcciones:

En esta fase se realizarían pruebas exhaustivas del proyecto para identificar posibles errores o problemas en las animaciones. Se corregirían los errores y se ajustarían las animaciones según sea necesario.

# Desarrollo de Animath

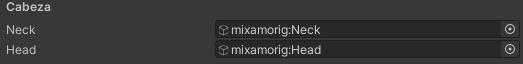
## 3.1. Pasos iniciales:

Para comenzar, es fundamental que el modelo que se va a utilizar ya tenga un sistema de Rigg con los huesos asignados correctamente a sus respectivas articulaciones (ver Fig. 1). Estas articulaciones incluyen:

* *Cabeza, Cuello*
* *Muslos, Rodilla*
* *Hombros, Codo, Muñeca*
* *Pecho, Tórax, Cadera*

Para facilitar el proceso, se recomienda que los joints del modelo se nombren según su función específica. Una vez que se tiene el modelo con su sistema de Rigg, se procede a importarlo a un nuevo proyecto en Unity y se crea un script que actuará como el controlador principal del plug-in.

## 3.2. Set-Up de Animath:

Para empezar a programar el script, es necesario llamar a cada uno de los joints del modelo que importamos en el código, utilizando la siguiente línea: 

#### Figura 3: Set-up joints del modelo dentro del plug.in

Este paso es esencial para emparentar el modelo u objeto 3D dentro de Unity, en este caso, hemos emparentado el cuello del modelo. Puedes ver una figura que muestra cómo emparentar el hueso con el código en la imagen adjunta.



#### Figura 4: Emparentamiento cuello

Una vez que hemos emparentado el cuello, podemos hacer el set-up de cada uno de los joints utilizando el mismo método. Es importante mencionar que para evitar errores en el código más adelante, es necesario nombrar correctamente cada una de las extremidades, incluyendo su respectiva posición (izquierda o derecha). De esta manera, aseguramos que el proceso de animación se realice de manera correcta y sin inconvenientes.

## 3.3. Funciones trigonométricas:

Una vez que hemos emparentado el modelo y definido cada uno de los joints en el código, podemos crear un efecto de animación fluido utilizando una variable de tipo float. Esta variable permite generar una variación en el valor de la rotación del joint en relación con el tiempo y el eje correspondiente, en este caso estamos animando el cuello. En la siguiente imagen se muestra cómo crear y utilizar los floats en el script:



#### Figura 5: Animación por ejes

Para lograr que la animación sea más orgánica y realista, utilizamos las funciones trigonométricas de seno y coseno. Con ellas, podemos modificar la amplitud, velocidad, punto inicial y desfase de la animación. La fórmula utilizada para generar la variación de la animación es:

***AMPLITUD \* SENO ((VELOCIDAD \* TIEMPO) – PUNTO INICIAL) + DESFACE***

## 3.4. Funcionamiento de la función del seno:

### 3.4.1. Amplitud:

La amplitud es un valor que determina la altura de la gráfica del seno en el eje y. Por ejemplo, si se establece una amplitud de 2, la gráfica tendrá una altura de 2 unidades en el eje y.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

#### Figura 6: Función Sen(x)

#### Figura 7: Función 2\*Sen(x)

Es importante tener en cuenta que este valor afecta directamente la amplitud de la rotación del joint correspondiente. Por lo tanto, elegir la amplitud adecuada es crucial para lograr la animación deseada.

### Gráfico Descripción generada automáticamente 3.4.2. Velocidad:

La velocidad, no produce ningún cambio en la forma en que se mueve el joint en sí mismo, pero se utiliza para multiplicar la velocidad, lo que influye en la velocidad general de la animación. Por lo tanto, si aumentamos el valor de tiempo, la velocidad también aumentará. En el código, utilizamos la variable de tiempo de la siguiente manera:

Imagen que contiene Diagrama

Descripción generada automáticamente

#### Figura 8: Función Sen(8x)

 En este caso, multiplicamos el tiempo real del programa por el valor de velocidad para obtener la velocidad de animación adecuada. Es importante tener en cuenta que el valor de tiempo puede variar dependiendo de la duración de la animación que se está creando:

#### Figura 9: Código tiempo

### 3.4.3. Punto Inicial:

El punto inicial de la gráfica es importante ya que permite iniciar la animación en un momento diferente a cero (valor predeterminado) y así mover la gráfica en el eje x para que su ondulación comience en un punto diferente. Si le restamos 10 al punto inicial, la gráfica se desplazará hacia la izquierda en el eje x, como se muestra en el siguiente ejemplo:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

#### Figura 10: Función Sen(x-10)

### 3.4.4. Desface:

Finalmente, el desfase es responsable de mover la gráfica en el eje y, lo que cambia la posición predeterminada de la misma. Al asignarle un valor, como 5, la gráfica se desplaza hacia arriba o abajo, según corresponda, como se muestra en la siguiente imagen:

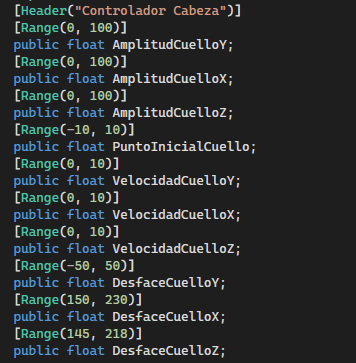
## Gráfico, Gráfico de líneas Descripción generada automáticamente3.5. Animaciones:

#### Figura 11: Función Sen(x)+5

Para continuar con las animaciones, volvemos al código y aplicamos las funciones dentro de este. Es necesario crear los floats correspondientes a:

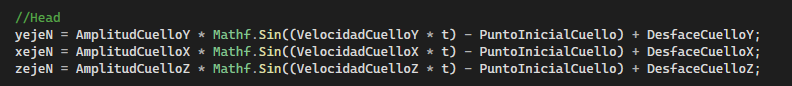
* *Amplitud*
* *Velocidad*
* *Punto de inicio*
* *Desfase*

Teniendo en cuenta esto, se hicieron varias funciones, ya que, al ser un espacio tridimensional, son necesarios 3 ejes, por lo cual, se crean amplitudes, velocidad y desfases diferentes para cada uno, pero el punto de inicio es el mismo para cada función del joint. A esto le vamos a dar un rango de 0-100, este será el rango del slider que permite modificar los valores en la interfaz de Unity.



#### Figura 12: Creación de floats

Después de crear los floats correspondientes a los joints, insertamos la función en el update del script, para ello utilizamos el float que creamos anteriormente, en este caso para el cuello cuyo nombre es **yejeN** el cual representa el Eje Y para el cuello, entonces se inserta la función:

 Al hacer esto, debemos continuar con la siguiente función, la cual permite que el joint haga la rotación en su eje según los valores que se le haya puesto. Utilizamos Quaternion para que en la simulación la función se pueda modificar y así crear diferentes estilos de animaciones sin tener que detener la simulación

#### Figura 14: Función de rotación

#### Figura 13: Inserta función en el Update

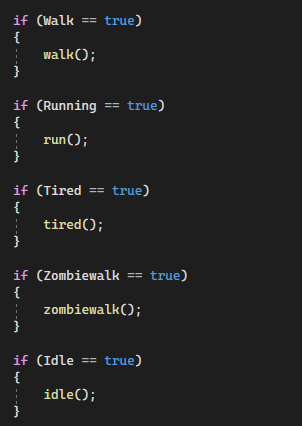
 En este punto, ya podemos iniciar la simulación, que nos muestra los sliders que permiten las modificaciones de las amplitudes, velocidades, desfases y puntos de inicio, para ver la animación hay que moverlos.

El plug-in Animath, cuenta con 5 animaciones definidas dentro de sí mismo, estas son: Idle, Walk, Running, Tired y Zombie Walk, estas animaciones se hicieron con apoyo de la herramienta Mixamo (*Mixamo*, s. f.). Al momento de tener los valores ya puestos dentro del código se ponen las animaciones, no sin antes crear un bool con el nombre de cada animación, lo cual permite que el plug-in tome los valores de la animación seleccionada.

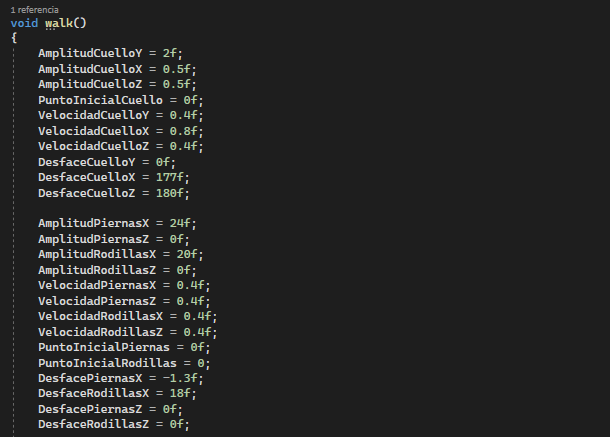
Texto

Descripción generada automáticamente

#### Figura 15: Nombre de cada animación dentro del código



#### Figura 16: Función que permite el cambio de animación.

­­­

#### Figura 17: Script toma valores de la animación

[](https://www.youtube.com/embed/CMVToGLutOc?feature=oembed)Para finalizar al dar play dentro de Unity y seleccionamos una animación predeterminada el modelo pasara a hacer la animación que se asigna ya sea Walk, Running, Tired o ZombieWalk.

#### Figura 18: Animación predeterminada (Walk)

Con esto, daríamos por terminado el proceso de creación del plug-in Animath.

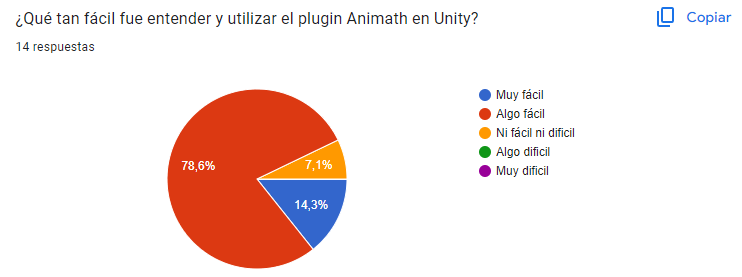
# Análisis de resultados

Para llevar a cabo este análisis, fue necesario implementar pruebas con catorce (14) usuarios, quienes estuvieron en disposición de explorar el plugin y hacer uso de él, en calidad de desarrolladores de videojuegos (en su mayoría principiantes). Se consideró oportuno, hacer estas pruebas de manera remota enviándoles a cada uno, un enlace donde iban a encontrar un manual de usuario, el script del plug-in y un modelo en calidad de archivo FBX, el cual, se aclaró, podía ser cambiado a elección del usuario según las indicaciones dadas en un inicio. Para esto, se especificó:

* Es requisito la versión 2019.4.36f1 del software Unity para correcta implementación de la prueba.
* El modelo a implementar debe contar esta estructura:
  + Cabeza
  + Cuello
  + Pierna izquierda como derecha
  + rodilla izquierda como derecha
  + Hombro izquierdo como derecho
  + Codo izquierdo como derecho
  + Muñeca izquierda como derecha
  + Pecho
* Para hacer más cómoda la experiencia, se recomienda nombrar cada joint con respecto a su funcionalidad en 3D.

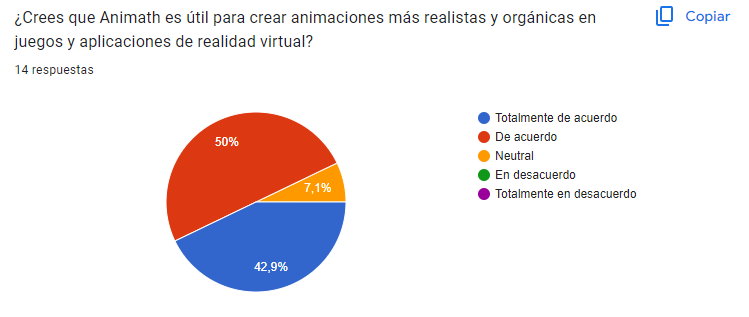
Una vez completado la instalación los usuarios empiezan a probar los sliders al entender su funcionalidad de cada sliders moviendo cada articulación del modelo fueron creando distintas animaciones divertidas y otras mas realistas si el usuarios llegara a tener mas tiempo con el plugin se hubiera podido pulir la animación haciendo la animación realista en sin tener que estar animando cada frame como es normal en una animación para videojuegos.

Al completar esta serie de pasos que se entregó a cada uno de los usuarios, se les solicitó llenar una encuesta formulada con relación a la experiencia de utilización del plug in, esto nos dio como resultado lo que se ve en las siguientes figuras:



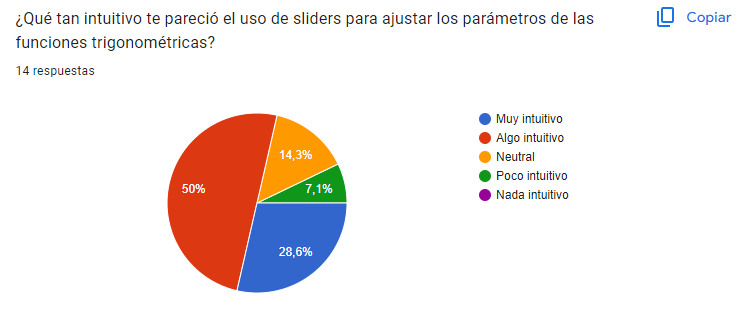
#### Figura 19: Prueba de Usuario. Pregunta 1

En donde la facilidad de utilizar al plug-in, tuvo un total del 78,6% de los votos a favor de la opción “Algo fácil”, lo que nos dice que el plug-in es lo bastante fácil de manejar para cualquier usuario.



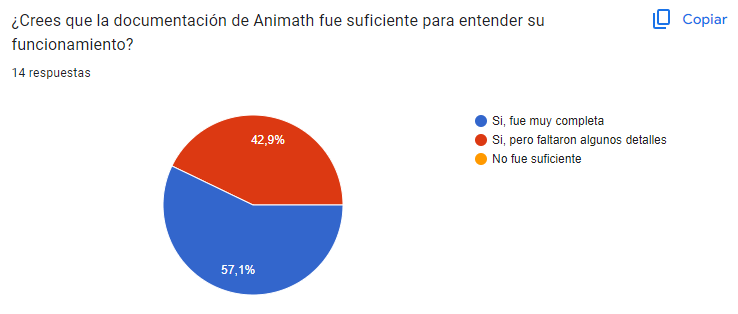
#### Figura 20: Prueba de Usuario. Pregunta 2

En donde podemos evidenciar que 42,9% más el 50% de los votos, aprueban la utilidad del plug-in para la realización de animaciones más realistas tanto en videojuegos como en aplicativos de realidad virtual que pudieran desarrollarse en Unity.



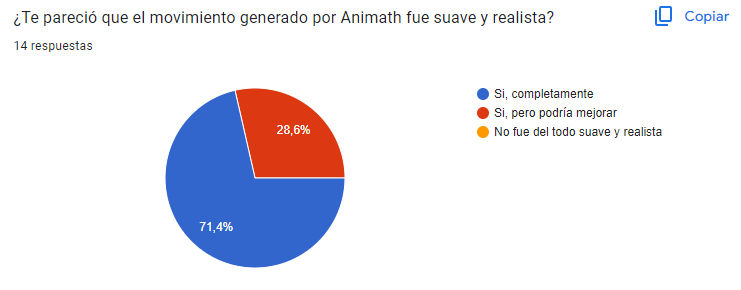
#### Figura 21: Prueba de Usuario. Pregunta 3

En donde tenemos una aprobación del 50% de los votantes con respecto a la facilidad de uso de sliders (deslizadores) como parámetros de cambio en cuanto a las funciones trigonométricas, que como ya vimos, son las que causan la variación en la animación según el desarrollador requiera.



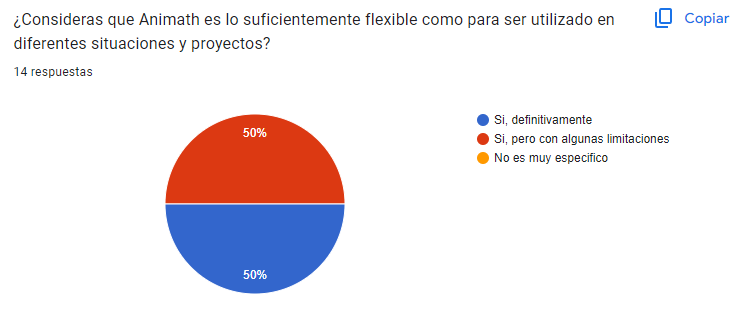
#### Figura 21: Prueba de Usuario. Pregunta 4

En donde se obtuvo una aprobación de 57.1% con respecto a la documentación que se entregó (Manual de usuario).



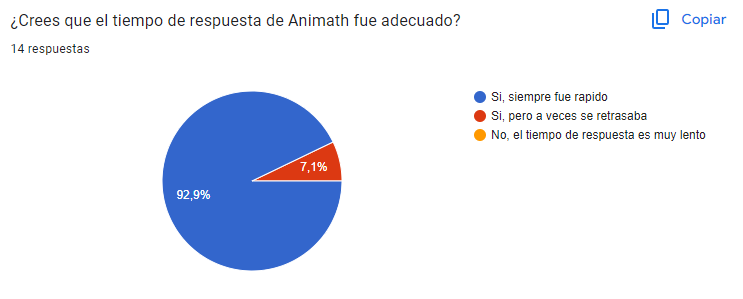
#### Figura 22: Prueba de Usuario. Pregunta 5

En donde se obtuvo una aprobación de 71.4% con respecto al realismo y la naturalidad de los movimientos que se pudieron lograr en la exploración del plug-in.



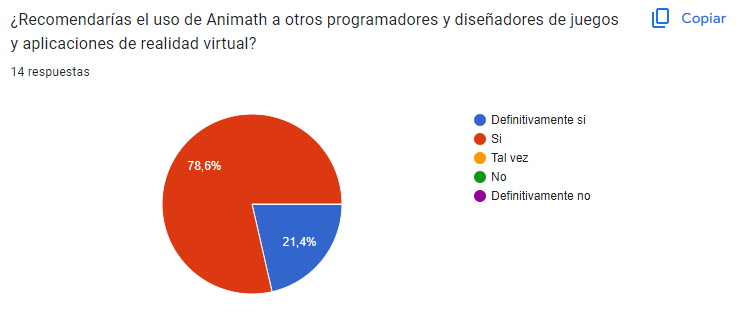
#### Figura 23: Prueba de Usuario. Pregunta 6

En donde se obtuvo una puntuación de 50% en la opción “Si, definitivamente” y 50% en la opción “Si, pero con algunas limitaciones”, esto debido a que no se les especificó a los usuarios los alcances y limitaciones que tenía el plug-in desde un inicio con fines constructivos y críticos para el desarrollo del plugin.



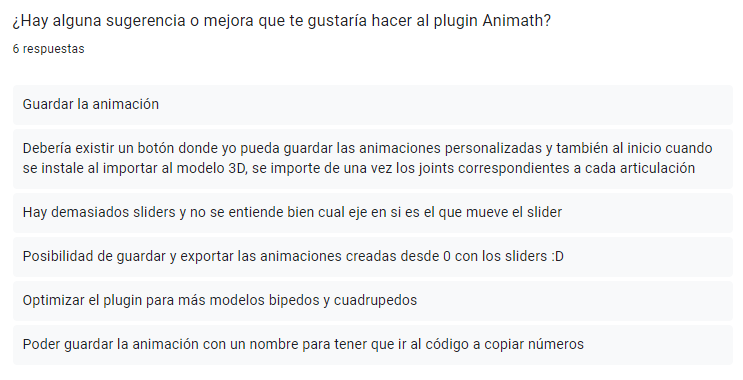
#### Figura 24: Prueba de Usuario. Pregunta 7

En donde se obtuvo una aprobación de 92.9% con relación al tiempo de respuesta y de uso del plugin.



#### Figura 25: Prueba de Usuario. Pregunta 8

En donde se obtuvo una aprobación de 78.6% con respecto a la opinión subjetiva de cada uno de los usuarios y su recomendación dentro de la industria. Lo cual nos dice que el plug-in es necesario y cumple con los objetivos iniciales de crear un plug-in que permite la animación procedural de personajes bípedos en la plataforma de Unity.



#### Figura 26: Prueba de Usuario. Pregunta 9

Con fines constructivos y críticos, se solicitó a los participantes de la prueba, sugerir cambios o hacer mejoras dentro del plug-in, esta opción no era obligatoria. Se obtuvieron 6 respuestas, de las cuales, al menos 3 hablan sobre la posibilidad de guardar las configuraciones, sin embargo, en las limitaciones se contempla que no se puede realizar esa acción, al menos en esta versión del plug-in, ya que las animaciones fueron generadas por los usuarios a través de la ejecución del proyecto en Unity, utilizando sliders para ajustar el movimiento. En caso de que se intente guardar la animación y se produzca un error en el script, el proyecto se detendría y los valores de las sliders se perderían. Por lo tanto, es recomendable tomar nota o capturar los valores utilizados para luego poder agregarlos directamente en el código. De esta manera, si surgen errores, no habría problema en detener el proyecto, ya que se pueden recuperar los valores guardados y reiniciar la animación desde el punto deseado. Esta precaución garantiza que se puedan conservar y reproducir las animaciones de manera precisa, incluso en caso de errores en el script.

## Evidencias:

Aquí se muestran unas animaciones que los usuarios crearon con el plugin, algunas pruebas se hicieron a través de discord lo cual los usuarios compartieron pantalla y se grabaron las animaciones que crearon.

[](https://www.youtube.com/embed/8bAWv6kYAZs?feature=oembed)

#### Figura 26: Prueba de Usuario. Evidencia 1

[](https://www.youtube.com/embed/FqXQTYW0-4U?feature=oembed)

#### [Prueba 3](https://www.youtube.com/embed/dOPt8s1anIM?feature=oembed)Figura 27: Prueba de Usuario. Evidencia 2

#### Figura 28: Prueba de Usuario. Evidencia 3

[](https://www.youtube.com/embed/Qz6ot4Re0cE?feature=oembed)

#### Figura 29: Prueba de Usuario. Evidencia 4

## Conclusiones:

El proceso de creación del plug-in Animath y su investigación, nos permite sacar las siguientes conclusiones:

* Mediante las funciones trigonométricas, se puede lograr que un personaje en 3D pueda adquirir una animación de un ciclo de caminado natural, lo cual ahorra tiempo y recursos en el proceso de creación.
* La calidad de la animación influye en la calidad del rigg que se le haga al modelo, según esto el movimiento podría llegar a verse afectado en algunos casos, más si se pasa por alto la cantidad de huesos o joints necesarios para la utilización del plug-in.
* El uso de funciones trigonométricas permite controlar la amplitud, velocidad, punto de inicio y desfase de la animación de manera precisa, lo que permite ajustar la animación para que se adapte perfectamente a los requisitos del proyecto. Además, las funciones trigonométricas son útiles para crear movimientos repetitivos y armónicos, como el movimiento de las olas del mar o el movimiento de una cuerda en una guitarra.
* El uso de funciones trigonométricas en la animación 3D a través de la animación procedural en Unity es una técnica poderosa que puede generar animaciones realistas y suaves en tiempo real. La animación procedural ofrece una gran flexibilidad y control sobre la animación, lo que permite a los animadores crear animaciones complejas y dinámicas con facilidad.
* La combinación de funciones trigonométricas y animación procedural en Unity ofrece una gran flexibilidad en la creación de animaciones, lo que permite explorar diferentes estilos y efectos visuales.
* El plug-in es útil para generar diversas animaciones que incluso pueden variar de las ya predispuestas, según las configuraciones de cada desarrollador.

# Tabla de figuras

[Figura 1: Limitación en cuanto a la cantidad de huesos del personaje. 12](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820716)

[Figura 2: Modelo de iteración. (Metodología iterativa, s. f.) 23](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820717)

[Figura 3: Set-up joints del modelo dentro del plug.in 26](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820718)

[Figura 4: Emparentamiento cuello 26](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820719)

[Figura 5: Animación por ejes 27](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820720)

[Figura 6: Función Sen(x) 27](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820721)

[Figura 7: Función 2\*Sen(x) 27](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820722)

[Figura 8: Función Sen(8x) 28](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820723)

[Figura 9: Código tiempo 29](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820724)

[Figura 10: Función Sen(x-10) 29](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820725)

[Figura 11: Función Sen(x)+5 30](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820726)

[Figura 12: Creación de floats 31](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820727)

[Figura 14: Función de rotación 31](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820728)

[Figura 13: Inserta función en el Update 31](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820729)

[Figura 15: Nombre de cada animación dentro del código 32](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820730)

[Figura 16: Función que permite el cambio de animación. 33](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820731)

[Figura 17: Script toma valores de la animación 33](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820732)

[Figura 18: Animación predeterminada (Walk) 34](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820733)

[Figura 19: Prueba de Usuario. Pregunta 1 36](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820734)

[Figura 20: Prueba de Usuario. Pregunta 2 37](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820735)

[Figura 21: Prueba de Usuario. Pregunta 3 37](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820736)

[Figura 21: Prueba de Usuario. Pregunta 4 38](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820737)

[Figura 22: Prueba de Usuario. Pregunta 5 38](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820738)

[Figura 23: Prueba de Usuario. Pregunta 6 39](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820739)

[Figura 24: Prueba de Usuario. Pregunta 7 39](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820740)

[Figura 25: Prueba de Usuario. Pregunta 8 40](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820741)

[Figura 26: Prueba de Usuario. Pregunta 9 41](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820742)

[Figura 26: Prueba de Usuario. Evidencia 1 42](#_Toc135820743)

[Figura 27: Prueba de Usuario. Evidencia 2 43](#_Toc135820744)

[Figura 28: Prueba de Usuario. Evidencia 3 43](#_Toc135820745)

[Figura 29: Prueba de Usuario. Evidencia 4 44](file:///D:\Docs\Descargas\Creación%20de%20Herramientas%20de%20Apoyo%20por%20Animación%20Procedural%20en%20Unity%20V.Final.docx#_Toc135820746)

# Lista de referencias

Animación. (s. f.). En *Real Academia de la Lengua Española*. Recuperado 30 de septiembre de 2022, de <https://dle.rae.es/animaci%C3%B3n>

Garza Mireles, D. (2014). *Animación digital y realidad virtual.* *Ciencia UANL*, *17*(67), 58-62.

Thomas, F., & Johnston, O. (1995). The Illusion of Life: Disney Animation. Disney Editions Deluxe.

Hernández Rielo, L. (s. f.). *Animación procedimental: Dando vida a través del código.* [Trabajo de fin de grado]. Universidad de Alicante.

Solarski, C. (2012). Drawing Basics and Video Game Art: Classic to Cutting-Edge Art Techniques for Winning Video Game Design. Watson-Guptill.

Short, T. X., & Adams, T. (2017). *Procedural Generation in Game Design* (T. Short & T. Adams, Eds.). A K Peters/CRC Press.

Biancard, C., Bona, R., & Hausheer, L. (2020). *LOCOMOCIÓN HUMANA: MODELOS Y VARIABLES BIOMECÁNICAS* [Ensayo]. Laboratorio de Investigación en Biomecánica y Análisis del Movimiento, Departamento de Ciencias Biológicas, Centro Universitario Paysandú, CENUR Litoral Norte, Universidad de la República de Uruguay, Paysandú, Uruguay Departamento de Kinesiología, Facultad de Medicina, Universidad de Concepción, Bío Bío, Chile.

*Metodología iteractiva*. (s. f.). BL Movil. <https://www.blmovil.com/wp-content/uploads/sites/3/2019/02/Scrum11.png>

*Mixamo*. (s. f.). [Software]. <https://www.mixamo.com/#/>

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot Modeling and Control*. https://www.control.isy.liu.se/en/student/graduate/robot/files/courseInfo.pdf

Menard, M. (2011). *Game Development with Unity*. http://ci.nii.ac.jp/ncid/BB06726189

Williams, R. A. (2001). *The Animator’s Survival Kit*. <http://ci.nii.ac.jp/ncid/BB19595990>

Unity. (s/f). Plataforma de desarrollo en tiempo real de Unity. Unity. Recuperado el 24 de mayo de 2023, de https://unity.com/es

Minaei Bidgoli, Behzad (2014). *Unity 3D Game Development: A Brief Overview* [Artículo]

Mullen, T. (2011). Introducing Character Animation.

Institut Henri Poincaré. (2014, November 17). *[Fête de la science 2014] De l’harmonie au Chaos - Aurélien Alvarez* [Video]. YouTube. <https://www.youtube.com/watch?v=7f5mB142qj8>

The Coding Train. (2017, March 17). *Coding Challenge #64.2: Inverse Kinematics* [Video]. YouTube. <https://www.youtube.com/watch?v=hbgDqyy8bIw>

Sneaky Kitty Game Dev. (2022, April 3). *Multiplayer FPS Procedural Animation Toolkit Plugin: Announcement!* [Video]. YouTube. <https://www.youtube.com/watch?v=33j8SKgrbV4>

Cleverpoly. (2022, September 20). *Unleash the power of procedural animation in blender* [Video]. YouTube. <https://www.youtube.com/watch?v=qvqRyRxAmM8>

game3Dover. (2016, January 29). *Cómo crear un árbol realista con la herramienta de Unity* [Video]. YouTube. <https://www.youtube.com/watch?v=75YssT--Rzc>

Maything. (2020, December 22). *Unity3D Circular Oscillation Effect* [Video]. YouTube. <https://www.youtube.com/watch?v=gu9AGLS-zQc>

Codeer. (2020, March 28). *Unity procedural animation tutorial (10 steps)* [Video]. YouTube. <https://www.youtube.com/watch?v=e6Gjhr1IP6w>

Margison, L. (2022). *Procedural animation for humans in Unreal Engine 5*. Udemy Academy. <https://www.udemy.com/share/108qJK/>

SebLague. (n.d.). *GitHub - SebLague/Intro-to-Gamedev: Tutorial series for gamedev with C# in Unity*. GitHub. <https://github.com/SebLague/Intro-to-Gamedev>