UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Análisis de la generación automática de animaciones para agentes virtuales (NPC) basada en respuestas de Generative Pre-trained Transformers.

Antonio Javier Altamirano Macias

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 19 de julio de 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Análisis de la generación automática de animaciones para agentes virtuales (NPC) basada en respuestas de Generative Pre-trained Transformers.

Antonio Javier Altamirano Macias

Nombre del profesor, Título académico **Diego Riofrio, PhD**

Quito, 19 de julio de 2024

© DERECHOS DE AUTOR

Nombres y apellidos: Antonio Javier Altamirano Macias

Código: 00210849

Cédula de identidad: 1718389024

Lugar y fecha: Quito, 19 de julio de 2024

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

Este trabajo investigó la generación de animaciones utilizando modelos de inteligencia artificial basados en Generative Pre-trained Transformers (GPT). Se evaluó la capacidad de un modelo orientado al procesamiento de texto para adaptarse a una tarea abstracta como generar animaciones 3D para agentes virtuales (NPC) a partir de descripciones textuales simples.

La creación de animaciones 3D es laboriosa y costosa, especialmente para proyectos de baja inversión, lo que invita a explorar alternativas automatizadas. Servicios similares como MOOTION utilizan modelos de difusión con este objetivo. Sin embargo, estudios de GPT-4 mostrando indicios de inteligencia artificial general (AGI) impulsaron este estudio a comprobar la viabilidad de utilizar GPT-4, a través de diferentes métodos de prompting, para generar animaciones coherentes. El desarrollo del proyecto se realizó utilizando custom GPT para la generación de animaciones, y la implementación de un plugin en Unity para la administración y comparación de dichas animaciones. Siendo GPT un modelo de procesamiento de texto, se decidió serializar los objetos de animación a JSON utilizando diferentes formatos para almacenar su información. Estos formatos se convirtieron en prompts con los que instruir a los custom GPT métodos generativos de animaciones. Se propuso un método cuantitativo de calificación para comparar estos formatos. Los resultados de estas versiones comprobaron que estadísticamente existieron prompts que ayudaron a GPT en la abstracción de su conocimiento para generar animaciones más que otros. Los resultados concluyeron que GPT-4 demuestra un nivel de abstracción sobre el movimiento humano, aunque sin llegar a la eficiencia de modelos de difusión. Se identificaron áreas de mejora en la compresión y se invita a explorar la habilidad de futuros modelos en tareas abstractas como un acercamiento a la creación de una AGI.

Palabras clave: Animación Generativa, GPT, Ruido Gaussiano, Difusión, Unity, Non Player Character, Generative Pre-trained Transformers, Animación 3D

ABSTRACT

This work investigated the generation of animations using artificial intelligence models based on Generative Pre-trained Transformers (GPT). The study evaluated the capability of a text-oriented model to adapt to an abstract task such as generating 3D animations for virtual agents (NPC) from simple textual descriptions.

Creating 3D animations is laborious and costly, especially for low-budget projects, prompting the exploration of automated alternatives. Similar services like MOOTION use diffusion models for this purpose. However, studies of GPT-4 showing signs of artificial general intelligence (AGI) motivated this study to test the viability of using GPT-4, through different prompting methods, to generate coherent animations. The project development involved using custom GPT for animation generation and implementing a plugin in Unity to manage and compare these animations. Since GPT is a text-processing model, animation objects were serialized to JSON using different formats to store their information. These formats were converted into prompts to instruct the custom GPT generative animation methods. A quantitative scoring method was proposed to compare these formats. The results of these versions showed that statistically, some prompts helped GPT abstract its knowledge to generate animations better than others. The results concluded that GPT-4 demonstrates a level of abstraction regarding human movement, although not reaching the efficiency of diffusion models. Areas for improvement in compression were identified, and it is suggested to explore the capabilities of future models in abstract tasks as an approach towards creating an AGI.

Keywords: Generative Animation, GPT, Gaussian Noise, Diffusion, Unity, Non Player Character, Generative Pre-trained Transformers, 3D Animation

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El presente trabajo expone el desarrollo de un proyecto de generación de animaciones utilizando modelos de inteligencia artificial basados en Generative Pre-trained Transformers (GPT). En concreto, se ha explorado la viabilidad y eficacia de utilizar GPT-4 para la creación de animaciones 3D para agentes virtuales (NPC) humanoides. Esta tarea es análoga a estudiar la habilidad de GPT para realizar cálculos matemáticos, algo para lo que no está diseñado [1]. Esto se debe a que la generación de animaciones como información es más afín a la generación de video, para lo que normalmente se utilizarían herramientas como modelos de difusión. El objetivo principal de este proyecto ha sido evaluar cómo un modelo principalmente orientado al procesamiento de texto puede ser adaptado para generar animaciones partiendo de una descripción textual simple y evaluando que tan coherentes son. De esta manera se analizó la habilidad de un Transformer como GPT4 para abstraer su conocimiento a valores numéricos representativos del movimiento humano.

Problemática

El origen de este proyecto yace en el uso de IA para agentes virtuales en el ámbito de la interacción conversacional. El servicio de Replika [2], una colaboración de OpenIA y Unreal Engine, utilizó transformes para crear NPC con los que tener conversaciones realistas y dinámicas dentro de un videojuego. En busca de mejorar este salto en la calidad de interacción con medios digitales nació la idea de proveer a estos NPC con movimientos corporales acordes con las conversaciones dinámicas que proyectos como Replika son capaces de generar. Esta idea toma la problemática de que una conversación sin movimientos corporales limita la inmersión de los usuarios. Sin embargo, también se extiende a una problemática relacionada con el uso y la creación de animaciones para aplicaciones variadas.

En el desarrollo de proyectos de bajo presupuesto, la implementación de animaciones puede ser un factor limitante debido al costo [3]. Paquetes de animaciones en páginas como MoCap[1], TurboSquid[2], SketchFab[3], o incluso el Asset store[4] de Unity[5] pueden llegar a los cientos de dólares, o hasta 5 dólares por animación individual. Esto viene bien para el desarrollo de un videojuego si se tiene presupuesto suficiente o si se busca una animación muy específica. Sin embargo, para proyectos de bajos recursos no es muy sostenible [4].

Justificación

Empresas en el mercado de la generación de animaciones por IA como sayMotion[6], Inworld IA[7] y MOOTION[8] utilizan modelos de difusión, una herramienta adecuada para la generación de imágenes y videos [5], los cuales son análogos a las animaciones corporales. Por otro lado, un transformer toma un prompt textual y trata de predecir cual es la palabra o token con mayor probabilidad de continuar al prompt que se le dio [6]. Se podría decir que un modelo de difusión es más adecuado que un transformer para la tarea de generación de animaciones.

Este proyecto se apalanca de estudios sobre la habilidad de GPT-4 para realizar tareas para las que no fue diseñado. Por ejemplo, Zvornicanin [7] estudia cómo GPT es capaz de realizar operaciones matemáticas con cierta exactitud a pesar de no estar entrenado para este propósito. Loeber [1] argumenta que al realizar cálculos es posible que GPT tome, por así decirlo, diferentes vías neuronales para intentar resolver estas operaciones. Esto indica que dentro de su entrenamiento se debieron formar caminos que resemblen la resolución de estas operaciones matemáticas, y que algunos caminos son más eficientes que otros. Expandiendo sobre esto,

[1] https://mocaponline.com/
[2] https://www.turbosquid.com/
[3] https://sketchfab.com/
[4] https://assetstore.unity.com/3d/animations
[5] Motor de videojuegos utilizado en el desarrollo de este proyecto, https://unity.com/
[6] https://www.deepmotion.com/saymotion-done
[7] https://inworld.ai/
[8] https://www.mootion.com/

Bubeck [8] explora la habilidad de GPT-4 en diferentes tareas para las que no fue diseñado como cálculo, visión, composición musical, generación de imágenes y programación. Todo esto con la idea de mostrar indicios de una inteligencia artificial general (AGI).

Si bien un transformer difícilmente genere resultados a la altura de un modelo de difusión, es posible que tenga una ventaja al ser capaz de cumplir el papel de animador y comunicador en un NPC simultáneamente. La ventaja de estudiar a GPT-4 como un AGI es que se espera una capacidad de realizar múltiples tareas de forma creativa y relacionada [9], mientras que un modelo de difusión es bueno en la realización de una tarea en específico y nada más [10]. Es en este contexto que se propuso estudiar la habilidad de GPT-4 en la generación de animaciones. Con este objetivo, se compararon diferentes métodos de prompting para indagar sobre caminos neuronales que tengan mejor habilidad de abstracción para esta tarea.

Objetivo general

- Analizar la capacidad que tienen los modelos de Generative Pre-trained Transformers (GPT) en la generación automática de animaciones realistas y coherentes en agentes virtuales (NPC), mejorando así la interacción y la inmersión en entornos virtuales.

Objetivos específicos

- Desarrollar un plugin para el editor de Unity que permita la gestión y evaluación de las animaciones generadas por GPT-4.
- Diseñar diferentes estructuras y formatos de animaciones, experimentando con la compresión y representación de datos para optimizar el rendimiento de GPT-4.
- Evaluar las animaciones generadas por GPT-4 en términos de coherencia y precisión del movimiento humano, identificando áreas de mejora y optimización.

ESTADO DEL ARTE

La generación de animaciones para NPC a través de IA está comenzando a entrar como un producto de la IA en el mercado, orientado a reducir costos e incrementar las posibilidades creativas de proyectos audiovisuales como video, medios interactivos, o videojuegos [11]. Empresas en este mercado como sayMotion y MOOTION se apalancan de capas de modelos de difusión y de físicas corporales propietarias que utilizan bases de datos propias generadas con Motion Tracking [12]. Con esto son capaces de proveer servicios de suscripción para brindar animaciones de calidad y hasta cierto punto configurables sobre sus parámetros de generación. Empresas inmersas en el mismo mercado de uso de animaciones como Unity, un motor de desarrollo de videojuegos, han implementado sus propios modelos de generación de animaciones como Unity Muse [13]. Además, su competencia en el mercado, Unreal Engine, también ha invertido en la implementación de IA en el desarrollo de NPC para videojuegos. El servicio Replika, una colaboración entre Unreal Engine y OpenIA, ha desarrollado agentes conversacionales que permiten desarrollar personajes con ciertas características configurables sobre quiénes son y cuál es su personalidad [2], de forma que usuarios pueden sostener conversaciones realistas con NPC.

En los últimos años se ha visto un progreso de los medios digitales hacia una mayor inmersión. Graficas cada vez más indistinguibles de la realidad, historias inmersivas donde la misma puede ser alterada por la participación del usuario [14], o inclusive el nuevo boom en la realidad virtual empujado por empresas como Meta[9] y Apple[10] introduciendo nuevos productos en el mercado. Este desarrollo en la forma en la que interactuamos con el mundo digital va de la mano del desarrollo de la IA y cómo últimamente se ha vuelto parte de nuestro día a día. Y aun así la IA sigue evolucionando, en la actualidad, hacia el objetivo de lograr la inteligencia

[9] https://www.meta.com/quest/quest-3/
[10] https://www.apple.com/apple-vision-pro/

artificial general (AGI) [9]. Estudios como [8] sobre GPT-4 muestran que este modelo podría encontrarse en los inicios de una AGI. Siendo un modelo entrenado sobre grandes cantidades de texto web, es aun así capaz de generar información fuera de lo que fue entrenado. Bubeck [8] muestra como GPT4 se desenvuelve en tareas variadas de manera simultánea. Por ejemplo, se le pide que realice un código en Latex para graficar un unicornio, obteniendo un resultado similar a lo que realizaría un humano. Esto es sorprendente, ya que muestra una habilidad de visualizar e interpretar el código como una imagen y correctamente abstraer el concepto e información de un unicornio a figuras de forma creativa.

Siendo ChatGPT de OpenIA uno de los recursos más utilizados de IA, si no el más utilizado a la vez que uno de los más hábiles, mucho se ha estudiado sobre su habilidad de realizar tareas fuera de su diseño. Loeber [1] muestra sus resultados sobre las habilidades aritméticas de GPT4. Si bien el modelo es un predictor de texto, en su entrenamiento y basto conocimiento se deben haber formado caminos neuronales que a través de texto reflejan o aproximan el significado de lo que son las diferentes operaciones aritméticas, mostrando indicios de razonamiento similar al humano. Particularmente sobre la generación de animaciones, Lan [15] muestra que es posible generar animaciones de rostro utilizando GPT como fuente de información para otras herramientas que se encargan de la animación. Es en base a estos estudios que se propuso investigar la habilidad de comprensión respecto al movimiento humano que tiene un Transformer como GPT-4. Utilizando a GPT como fuente generativa de información y a Unity como herramienta para poder interpretarla como animación. Considerando la posibilidad de que, en la evolución de estos modelos, aparezca la posibilidad de crear NPC que sean capaces de interactuar tanto verbalmente como físicamente bajo un mismo modelo de IA o AGI.

METODOLOGIA

Estudio estructural de animaciones en Unity

En el ámbito de las animaciones, la estructuración y almacenamiento de su información juega un papel fundamental en la creación del formato textual con el cual GPT4 será capaz de generar animaciones. Se explora específicamente cómo se estructuran las animaciones dentro del entorno de Unity, uno de los motores de juego más populares y poderosos en la industria, y posteriormente, cómo esta estructura puede ser representada y almacenada utilizando formatos como JSON, CSV, XML, etc., facilitando así su manipulación y reutilización.

Unity maneja las animaciones mediante un modelo que puede compararse de manera simplificada con el funcionamiento de un video. Un video puede conceptualizarse como una secuencia de imágenes o frames, donde cada frame es una lista de píxeles con valores numéricos específicos para los canales RGB que varían con el tiempo. De manera análoga, en Unity, una animación se compone de una serie de keyframes. Estos keyframes son puntos en el tiempo que especifican el valor exacto de las propiedades a animar, como la posición, rotación o escala de un objeto [16].

Cada articulación del modelo animado tiene asociadas curvas de animación o animation curves, que son secuencias de pares tiempo-valor, representando los cambios en una propiedad específica de la articulación a lo largo del tiempo. Por ejemplo, se podría tener una curva de animación para la rotación alrededor del eje X del pie y otra curva separada para el eje Y. Esta descomposición permite una gran flexibilidad y control sobre la animación de cada parte y propiedad del modelo. Cabe destacar que tanto la escala como la posición se manejan usando los ejes XYZ como objetos de tipo Vector3, que viene a ser un arreglo de 3 valores decimales dentro de Unity. Sin embargo, las rotaciones se manejan internamente usando Quaternions como tipo de dato. Los Quaternions son un vector de 4 dimensiones que permiten manejar las

rotaciones de forma más eficiente en los motores gráficos. Para dar una explicación breve, estos definen un vector de dirección utilizando sus ejes XYZ y después utilizan su último eje W para expresar la rotación alrededor de este vector [17]. Además, los valores que conforman un Quaternion están normalizados entre -1 y 1 [18]. Esto difiere de los ángulos de Euler con los cuales normalmente las personas estamos acostumbradas a entender las rotaciones, esto es usar el sistema de 0 a 360 grados sobre los ejes XYZ para expresar rotación.

Unity estructura las articulaciones de los personajes animados en una jerarquía tipo RIG [19], donde cada parte del cuerpo está conectada a sus partes adyacentes de manera jerárquica, permitiendo que los movimientos de una parte influyan relacionalmente en las demás. Por ejemplo, la mano está conectada a la muñeca, que a su vez está conectada al codo y así sucesivamente, con las caderas frecuentemente sirviendo como el origen de la jerarquía del cuerpo.

Formulación de un formato textual para GPT-4 usando JSON

Sabiendo que GPT4 tiene la habilidad de realizar tareas fuera de su diseño original con cierta efectividad, se piensa estudiar la habilidad de GPT4 en la generación de animaciones corporales. Con este objetivo, es necesario primero definir un procedimiento por el cual se pueden convertir respuestas textuales en objetos de animación que se puedan aplicar a un modelo 3D.

El ambiente que se va a utilizar para el desarrollo de este proyecto es el editor de Unity, por lo que el primer paso fue conseguir un modelo 3D con un RIG que podamos modificar libremente en Unity. El RIG obtenido en la sección anterior permitió estudiar cómo se manejan las animaciones en el ambiente de Unity. En base a esto, se plantea definir una estructura textual que sea capaz de almacenar una animación de forma que pueda ser interpretada por GPT-4.

Se plantea el uso de formatos textuales como JSON, CSV, XML, etc. El formato escogido contendrá información principalmente numérica sobre los tiempos en los que hay cambios en los movimientos del cuerpo y los valores exactos a donde estos se mueven.

En este caso se optó por utilizar JSON (JavaScript Object Notation). JSON es un formato de texto ligero para el intercambio de datos, ideal para este propósito debido a su habilidad para serializar objetos de programación y mantener la estructura jerárquica que ya tienen las animaciones [20].

En la práctica, cada keyframe puede ser representado como un objeto en JSON, con propiedades que especifican los momentos exactos de tiempo y los valores asociados para cada curva de animación de cada articulación. Esta estructura se replicó utilizando el modelo de datos en Unity definido por las siguientes clases en C#:

Tabla 1. Versión inicial de las *clases* utilizadas para serialización JSON.

```
[Serializable]
public class AnimationData{
  public string animationName;
  public KeyframeDataLoad[]
keyframes;
}
```

```
[Serializable]
public class
KeyframeDataLoad{
    public string path;
    public string
propertyName;
    public CurvePoint[]
curve;
}
```

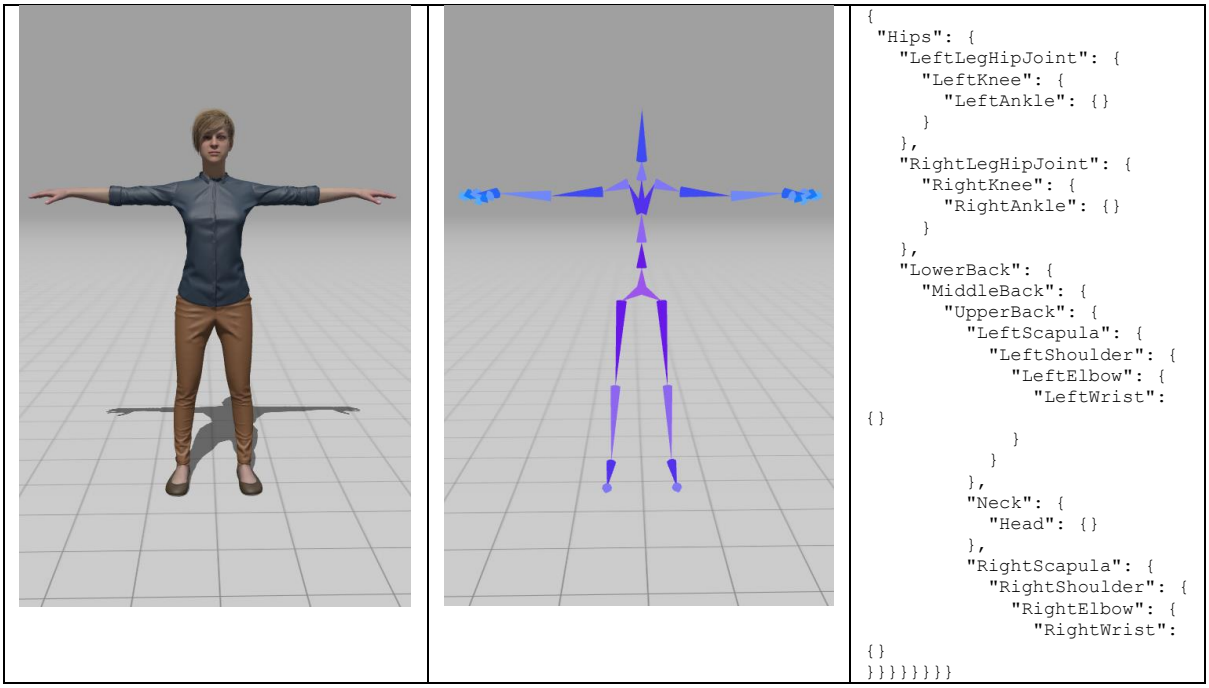
```
[Serializable]
public class CurvePoint{
    public float time;
    public float value;
}
```

Obtención del dataset de animaciones utilizando Mixamo

Existen plataformas que ofrecen animaciones de libre descarga. Sin embargo, cuando el objetivo es recopilar un compendio de animaciones aplicables para un mismo RIG, buscar animaciones entre diferentes sitios tiene sus desventajas. Muchos sitios que ofrecen descargas gratuitas tienen un máximo de descargas diario; después solo permiten animaciones que son pagadas. Con el propósito de recopilar una cantidad considerable de animaciones variadas, se

utilizó la librería de animaciones Mixamo de Adobe[11]. Esta es una plataforma que ofrece una extensa biblioteca de animaciones y modelos 3D. La ventaja de esta plataforma sobre otras páginas es que ofrece todas sus animaciones de forma gratuita al contar con una cuenta de Adobe, y permitió descargar sin límite sus animaciones aplicadas a un RIG en específico de su colección de modelos 3D con un código de automatización [21]. Con estas ventajas, se escogió un modelo 3D de su librería como el RIG seleccionado para este proyecto. Este RIG presenta la siguiente estructura:

Figura 1. Modelo 3D, RIG, y Estructura Corporal




```
{
 "Hips": {
   "LeftLegHipJoint": {
     "LeftKnee": {
       "LeftAnkle": {}
     }
   },
   "RightLegHipJoint": {
     "RightKnee": {
       "RightAnkle": {}
     }
   },
   "LowerBack": {
     "MiddleBack": {
       "UpperBack": {
         "LeftScapula": {
           "LeftShoulder": {
             "LeftElbow": {
               "LeftWrist":
{}
             }
           }
         },
         "Neck": {
           "Head": {}
         },
         "RightScapula": {
           "RightShoulder": {
             "RightElbow": {
               "RightWrist":
{}
}}}}}}}}
```


La estructura JSON que se muestra en la figura 1 representa la forma en que las diferentes partes del cuerpo se encuentran en el modelo 3D. Dentro de Unity un objeto puede tener hijos, de forma que su posición, escala, y rotación son afectadas por el padre. Es decir, los hijos varían de forma relativa al objeto padre. Con este contexto, la estructura JSON mostrada representa como cada objeto individual que construye el RIG se relaciona entre sí.

[11] https://www.mixamo.com/#/

Traducción de la estructura JSON y testeo

Para procesar las animaciones obtenidas de Mixamo como archivos binarios ".anim", se transformaron estas animaciones en un formato textual que pueda ser interpretado y utilizado por GPT-4. Esto requirió la creación de código en Unity que implemente funcionalidad para traducir estos archivos binarios a JSON, utilizando la estructura de datos previamente establecida por las clases de la tabla (1). Esto facilitó la conversión de animaciones a un formato de texto que sirvió como material de entrenamiento para GPT-4.

Además, se implementó la funcionalidad inversa, asegurando que la traducción de archivos ".anim" a JSON mantenga la integridad de la animación original. Este paso fue esencial para asegurar que el proceso de conversión sea reversible y el modelo genere animaciones que se puedan después reproducir dentro de Unity.

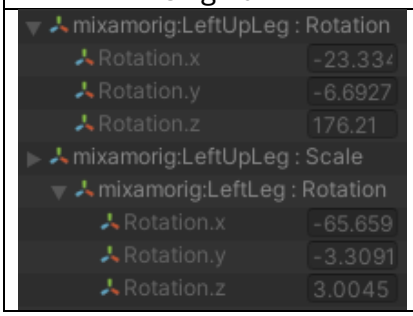
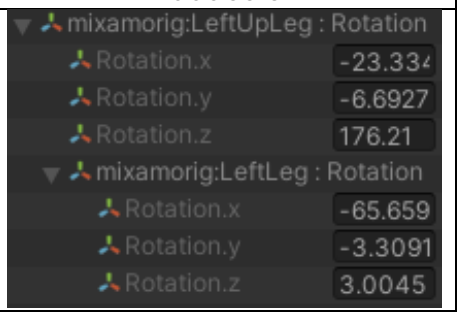
Unity maneja las animaciones como objetos AnimationClip, los cuales internamente albergan una lista de propiedades animadas. Estas propiedades contienen una lista de pares tiempo-valor conocida como curva de animación. Además, el nombre de la articulación y el eje al que pertenece esta curva. Por cada articulación del cuerpo existen 4 propiedades animadas, una por cada eje del vector que conforma un Quaternion. Es decir, se anima por separado cada eje XYZW de una misma articulación. La información sobre estas curvas no es accesible en runtime. Sin embargo, la librería Unity Editor permite acceder a esta información a través de scripts que corren únicamente dentro del ambiente de desarrollo. Por este motivo, se desarrolló una herramienta para automatizar la conversión de animaciones utilizando scripts de editor.

Esta herramienta implementó varias funcionalidades que son descritas más adelante. La primera funcionalidad implementada se encarga de la conversión de animaciones entre objeto y JSON de forma bidireccional. El proceso de conversión implicó cargar una animación a Unity

como un objeto AnimationClip y acceder a sus propiedades para almacenarlas en un objeto AnimationData utilizando las clases de la tabla (1). Este objeto se serializa a JSON para obtener las animaciones textuales. El proceso inverso consistió en convertir el texto JSON a un objeto AnimationData y utilizar su información para reconstruir un objeto AnimationClip. Esto se aplicó para todas las animaciones de Mixamo en un dataset.

Para probar la integridad de estas animaciones JSON se utilizaron dos copias iguales del RIG, a la primera se le aplicó la animación original y a la segunda la traducción. Cargar estas animaciones sobre dos RIG en runtime permitió comparar los ángulos de cada keyframe dentro de Unity, comprobando que los valores se mantuvieron iguales y se conservó la integridad de la animación. Por ejemplo, se muestra a continuación una comparación de keyframes para ambos RIG tomada directamente del editor de Unity.

Tabla 2. Comparación de keyframes entre animación original y traducción.

Original	Traducción
	

De estos primeros archivos JSON generados, se pudo destacar la cantidad de números decimales utilizados para almacenar los valores de cada keyframe. De aquí hubo indicios de una primera optimización. Esto se debe a que, si bien menos decimales representan menos exactitud, también tienen el beneficio de reducir el tamaño de los archivos JSON, lo cual probaría ser ventajoso.

Primera versión del template, estructura corporal y configuración del prompt de GPT

Con una funcionalidad de traducción implementada, se procedió a crear un custom GPT utilizando GPT-4. Los custom GPT, en comparación con un chat normal, tienen la ventaja de poder configurar el comportamiento del GPT a través de un setup prompt [22]. Además, son capaces de almacenar archivos textuales como una base extra de conocimiento que se puede proveer al momento de crearlo. Con la configuración correcta, es posible crear un chat personalizado capaz de generar las animaciones que buscamos en el formato correcto, simplificando la tarea de generación a un prompt explicando lo que se busca animar. La posibilidad de añadir archivos a su base de conocimiento también permitió proveer de ejemplos de animaciones para que GPT-4 pueda usar de referencia.

El setup prompt se redactó en base a la estructura JSON. Se instruyó sobre el proceso de generación de archivos JSON que se debe seguir, haciendo énfasis en el formato y en la manera en que debe procesar la petición del usuario. Se le pidió que genere valores adecuados utilizando su conocimiento sobre anatomía humana y animación. Además, se le instruyó a limitar su generación de respuesta únicamente al archivo JSON y evitar texto adicional. Así, maximizando el uso de los tokens permitidos de respuesta a la generación de la animación [23]. Se puede encontrar este primer setup prompt en el anexo (1).

A este setup prompt también se le añadieron dos bloques de estructuras JSON. El primero es la jerarquía que relaciona las diferentes partes del cuerpo en el RIG, el cual se muestra en la figura 1. La segunda es una simplificación de la estructura que debería utilizar para generar la animación. Este ejemplo muestra las propiedades del JSON y los diferentes valores que debe almacenar:

Figura 2. Ejemplo de la estructura JSON que GPT utilizó como plantilla.

```
{
    "animationName": "ExampleAnimation",
    "keyframes": [
        {
            "path": "Spine2/RightShoulder",
            "propertyName": "m_LocalRotation.z",
            "curve": [
                { "time": 0.0, "value": 0.0 },
                { "time": 1.0, "value": 45.0 }
            ]
        },
        {
            "path": "Spine2/RightShoulder/RightArm",
            "propertyName": "m_LocalRotation.z",
            "curve": [
                { "time": 0.0, "value": 0.0 },
                { "time": 1.0, "value": 90.0 }
            ]
        },
        {
            "path": "Spine2/RightShoulder/RightArm/RightForeArm",
            "propertyName": "m_LocalRotation.z",
            "curve": [
                { "time": 0.0, "value": 0.0 },
                { "time": 1.0, "value": 45.0 }
            ]
        }
    ]
}
```

Añadir la jerarquía del cuerpo y un template para la generación de la animación dentro del setup prompt nos asegura que en cada interacción con el custom GPT esta información será parte de su ventana de contexto. Además, esta información también se añadió como archivo en su librería de conocimiento. También, se añadieron animaciones del dataset que representen movimientos variados sobre distintas partes del cuerpo. Esto incluye animaciones como caminar, sentarse, aplaudir, mirar hacia arriba, y movimientos de brazos.

La base de conocimiento que se utilizó para el custom GPT consistió en: estructura jerárquica del cuerpo, template del formato JSON, 6 animaciones variadas, y finalmente un último archivo de ejemplos. Este archivo tuvo la función de utilizar el limitado espacio sobrante en la librería de conocimiento del custom GPT después de cargar los archivos anteriores. El proceso para la generación de compendios de animaciones fue el siguiente:

1. Durante la conversión del dataset de Mixamo a JSON, se obtiene una referencia a cada objeto de animación individual y su peso en bytes.
2. Se crea una lista de estos objetos de animación y un sumador para registrar el tamaño en bytes que tendría la suma de objetos en la lista.
3. Se define un límite de bytes a partir del cual la lista ya se considera llena y debe guardarse como archivo de compendio.
4. Mientras se itera sobre el dataset y se crean objetos de animación, se va poblando la lista de objetos hasta llegar al límite de bytes. Cuando esto ocurre, se serializa la lista a un archivo de compendio.
5. Después de generar un compendio, si existen más animaciones pendientes, se reinicia el sumador y vacía la lista de objetos para repetir el proceso.

Como resultado de este proceso se encontró que el tamaño restante en la librería de conocimiento se encontraba alrededor de los 2MB. Utilizando compendios de este tamaño, se contuvieron alrededor de 40 animaciones del dataset. Para la librería de conocimiento se escogió un compendio aleatorio.

Pruebas con GPT4 y limitaciones

El tamaño de las animaciones en este primer formato JSON fue considerablemente grande. Las animaciones más extensas dentro del dataset generado, de pasos de baile principalmente, resultaron en archivos de texto alrededor de los 5mb. De esta estructura y extensión se observó que al separar las rotaciones en cada eje se creó mucha repetición de información de 2 maneras. Primero, dado que las rotaciones se trabajan en Quaternions, cada rotación, aunque sea en un solo eje, depende de todos los valores XYZW para un mismo keyframe. Esto significa que para cada eje existen valores de tiempo repetidos. Dado que cada parte animada consta de un

quaternion de 4 dimensiones, conteniendo pares tiempo-valor, podemos decir que 3 de esas listas de tiempos son información repetida.

Segundo, hay repetición de información a través del uso de "{,[,(" dada por la cantidad de separación de los datos y también por la estructura de los objetos. Por ejemplo, si cada keyframe es un objeto con dos variables "time" y "value", entonces dentro del archivo se va a repetir el uso de estas palabras en función de la cantidad de keyframes que tenga.

Tercero, es posible representar rotaciones como ángulos de Euler que solo utilizan 3 dimensiones a diferencia de un Quaternion. Utilizar este tipo de rotación podría reducir un cuarto de la extensión del archivo. Esto es relevante para evitar tokens interpretados por GPT sin aportar información relevante a la animación.

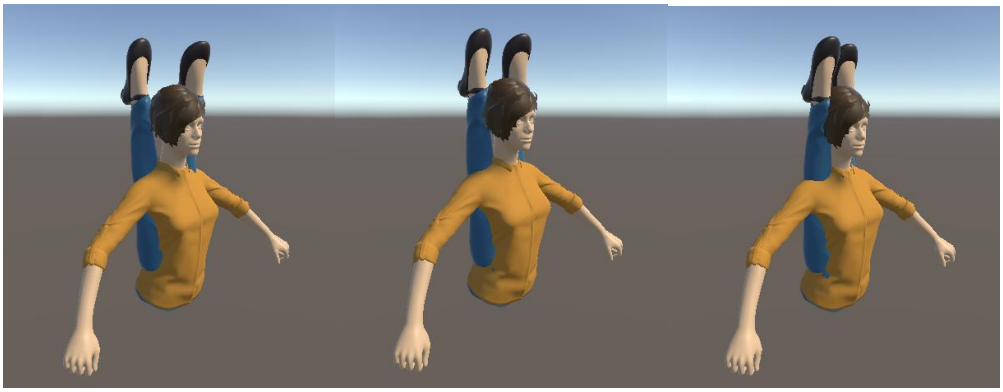
De este análisis se concluyó que se podría comprimir la información modificando la estructura de las clases utilizadas para serializar las animaciones a JSON. Estas modificaciones a su vez pueden ir de la mano con diferentes implementaciones del setup prompt. En base a los resultados de Loeber [1], es posible que estudiar diferentes versiones de setup prompt provoque mejores o peores resultados.

Una última fuente de extensión para el documento está dada por la cantidad de keyframes animados por parte del cuerpo. Se considera la posibilidad de simplificar la animación a una longitud de tiempo establecida, dividida en una cantidad de keyframes predefinida.

Pruebas iniciales con GPT-4 revelaron limitaciones significativas en la capacidad de generar animaciones competitivas con respecto a herramientas en el mercado como MOOTION o sayMotion que utilizan modelos de difusión. Sin embargo, estas animaciones si mostraron un nivel básico de entendimiento del prompt, demostrando cierto nivel de abstracción. Para un

prompt de caminata básica, el resultado fue que el RIG dobló las piernas 180 grados hacia arriba, como se muestra en la figura 2. Sin embargo, la flexión de las piernas una detrás de la otra a la par del movimiento de cadera sí se asemejaba a una caminata.

Figura 3. Keyframes de animación para una caminata para la primera versión



Con estas observaciones se propusieron variaciones al formato para estudiar la diferencia en resultados que pueden generar diferentes prompts [24]. Para esto se implementaron variaciones sobre el tipo de rotación utilizado y sobre la estructura de los objetos utilizados para almacenar la información de la animación, buscando hacerlo más eficiente.

Optimización y versiones

Compactación de información (vectores y reducción de decimales)

Para optimizar el almacenamiento de información en JSON se consideran modificaciones que reduzcan la cantidad de caracteres utilizados en el archivo. Primero, es posible integrar las curvas de animación de los ejes que conforman un Quaternion en una sola curva de vectores. Se generarían listas paralelas, una conteniendo valores de tiempo y otra con vectores. De esta forma, se simplifican 4 curvas de animación, se reduce el uso de caracteres como “{,[,(”, y también la repetición de nombres de variables en JSON por la clase CurvePoint.

Segundo, muchos caracteres se utilizan en el almacenamiento de decimales debido a cómo Unity serializa valores float. Reducir la cantidad de decimales podría ser una manera de reducir considerablemente el tamaño de los archivos, mientras esto no afecte la representación de los ángulos. Para decidir la cantidad de decimales adecuados se mapeó el rango de valores que pueden tomar los Quaternions a un rango de 360 grados. Los Quaternions manejan valores de 1 a -1, por lo que usando una simple regla de 3 se pudo estimar la significancia que tiene cada decimal sobre el ángulo resultante. Para esto se considera el tamaño del rango de valores permitido como la distancia de -1 a 1.

Tabla 3. Impacto de cantidad de decimales en conversión de Quaternions a ángulos de Euler

$tamaño\ del\ rango = V_f - V_o = 1 - (-1) = 2$		
$2 \rightarrow 360°$ $0{,}1 \rightarrow x$ $\quad x = 18°$	$2 \rightarrow 360°$ $0{,}01 \rightarrow x$ $\quad x = 1{,}8°$	$2 \rightarrow 360°$ $0{,}001 \rightarrow x$ $\quad x = 0{,}18°$

Movimientos en incrementos de aproximadamente 0,2 grados en un rango de 360 grados son prácticamente imperceptibles al aplicarlos al RIG. Tomando esto en cuenta, se redujeron los valores a 3 decimales. De forma similar se redujo los valores de tiempo a 2 decimales tomando en cuenta que el ojo humano puede procesar entre 30 y 60 keyframes por segundo [25]. Con 2 decimales se tienen 100 keyframes por segundo, que cubren la visión humana.

Quaternions a Euler para mejor comprensión por GPT

La empresa Anthropic ha realizado estudios sobre interpretabilidad en transformers [26] encontrando que, a partir de su entrenamiento, caminos neuronales pueden aprender a representar con cierta fidelidad conceptos abstractos. Podemos considerar en el entrenamiento de GPT-4 la cantidad de información que pudo formar su entendimiento sobre la relación de movimientos corporales con ángulos y con Quaternions. Es posible que, si hubo más información relacionando un tipo de rotación más que otro, uno de estos conceptos se haya

abstraído mejor. Por lo tanto, se explora la posibilidad de que, al generar animaciones utilizando ángulos de Euler o Quaternions, uno de estos arroje mejores resultados.

Para familiarizar a GPT con el RIG que debe animar, se proveyó a su librería de conocimiento con un archivo de rangos de rotación permitidos en base a la anatomía humana. Con esto, se espera mejorar su habilidad para relacionar valores numéricos con su efecto sobre el RIG. Estudios sobre el movimiento humano relacionado al campo de la visión por computadora [27] muestran formas de extrapolar poses tridimensionales de imágenes. Mientras este estudio utilizó motion tracking para analizar una variedad de movimientos. En este proyecto se cuenta con un modelo 3D con el que se encontraron estos rangos manualmente. Para esto se generaron poses directamente con el RIG y se registraron los ángulos máximos y mínimos de rotación por articulación que son coherentes. Esta información se almacenó en el archivo template.

Compresión utilizando longitud de tiempo y cantidad de keyframes predefinidos

Eliminar el tiempo de los keyframes en archivos JSON fue el método más brusco de compresión. En este caso, la idea fue forzar la animación dentro de una longitud de tiempo predefinida y dividida en intervalos iguales dependientes de la cantidad de keyframes. De esta forma se vuelve fácil reconstruir los puntos de tiempo para cada keyframe y es posible eliminar esta información del archivo. Si bien hay una gran reducción de tamaño, es un método brusco al perjudicar la fidelidad de la animación. En este caso, al introducir una animación existente a través del proceso de traducción, la animación resultante se vuelve una aproximación de la original que depende de la cantidad de keyframes utilizados. Mientras más keyframes, más se asemeja la animación resultante a la original.

Para implementar este método de compresión fue necesario modificar el código de traducción. Las curvas de animación normales siguen un patrón de tiempo definido. En este caso, es

necesario asumir una longitud y una cantidad de Keyframes por segundo predefinida. Por este motivo, en lugar de acceder a las curvas, se registró la rotación de cada parte del cuerpo en cada división de tiempo, tomando muestras en intervalos regulares.

Definición de 6 custom GPT basados en diferentes versiones del formato

En referencia al artículo de Anthropic [26], se espera encontrar un formato de generación que utilice un camino neuronal con mejor habilidad para relacionar valores numéricos con movimientos corporales. Por este motivo, se propone distribuir estas optimizaciones de compresión y diferentes tipos de rotación en 6 versiones de formatos para generación de animaciones. Con esto se espera comparar los resultados que estas diferentes variaciones pueden generar y encontrar una que sea estadísticamente mejor que las demás.

3 tipos de compresión se aplicaron utilizando ángulos en Euler y después Quaternions, creando 6 versiones en total. La estructura se mantuvo igual entre tipos de rotación, con la excepción de que Euler utilizó un eje menos y valores en un rango de 360 grados en vez de 1 a -1 en el caso de Quaternions.

Para cada versión se creó un custom GPT cuyo setup prompt reflejara estas diferencias. También se generaron templates distintos para cada formato, y se reutilizó la misma estructura corporal. El objetivo de estos custom GPT es crear diferentes chats con los que interactuar. Las animaciones generadas se almacenaron para después ser estudiadas y comparadas entre versiones, con lo cual encontrar un formato ganador. Referencias a los custom GPT creados para este proyecto se pueden encontrar en los Anexos (B). Los 3 tipos de compresión aplicados se describen a continuación:

- Full: Versión original del formato con ejes separados y listas de CurvePoints. Es el formato más extenso, aunque más detallado sobre la descripción de la animación.
- Compresión en vectores (Compressed): Se aplican diferentes estructuras de datos para evitar la repetición de información y de caracteres. Cada parte animada del cuerpo cuenta de una sola curva de animación que consta de listas paralelas tiempo-vector.
- Compresión en tiempo (Tokens): Versión con mayor compresión. Prioriza el almacenamiento de información en la menor cantidad posible de tokens. Para esto se modifica mucho la estructura JSON anterior. El archivo JSON almacena todas las partes del cuerpo que pueden ser animadas como listas de vectores de igual longitud. La longitud de la lista es igual a la cantidad de keyframes por segundo establecida. Para que estas animaciones tengan una granularidad similar al de los otros formatos, se utilizaron 5 keyframes por segundo.

Código de traducción para diferentes versiones

Cada versión del modelo utilizó una estructura de datos distinta, por lo que fue necesario definir métodos de traducción adaptados a cada una de las 6 versiones. Entre estos métodos se tuvieron que implementar 3 variaciones notables a estas funciones. Primero, al integrar los ejes dentro de un solo vector, se tuvo que modificar la estructura de las curvas. En lugar de utilizar CurvePoints se utilizaron listas paralelas para almacenar por un lado tiempos como una lista de floats, y por otro lado vectores como un arreglo bidimensional de floats cuyo tamaño varía entre 3 y 4 dependiendo del tipo de rotación. Segundo, para comprimir en intervalos de tiempos predefinidos se utilizó una estructura de datos completamente diferente. Esta consistió en una lista de las 20 partes del cuerpo conformando al RIG, donde cada variable tenía el nombre de la parte del cuerpo a la que se refería y representaba una lista de vectores. En la reconstrucción

del objeto de animación se relacionó el nombre de cada lista con la parte del cuerpo correspondiente, utilizando un diccionario.

Tabla 4. Comparación de estructuras JSON con ejemplo de 2 keyframes.

Full	Compressed	Tokens

<pre>{
"animationName": "AnimName",
"keyframes": [
{
"path": "BodyPart",
"propertyName":
"m_LocalRotation.x",
"curve": [
{"time": 0.00, "value": 0},
{"time": 0.50, "value": 15}
]
},
{
"path": " BodyPart ",
"propertyName":
"m_LocalRotation.y",
"curve": [
{"time": 0.00, "value": 0},
{"time": 0.50, "value": 30}
]
},
{
"path": " BodyPart ",
"propertyName":
"m_LocalRotation.z",
"curve": [
{"time": 0.00, "value": 0},
{"time": 0.50, "value": 20}
]
}
]
}</pre>
	<pre>{
	"animationName": "AnimName",
	"keyframes": [
	{
	"path": "Neck",
	"propertyName": "m_LocalRotation",
	"time": [0.00,0.50],
	"value": [
	[0, 0, 0],
	[15, 30, 20]
	]
	}
	]
	}</pre>
		<pre>{
		"animationName": "AnimName",
		" BodyPart ": [
		[0, 0, 0],
		[15, 30, 20]
		]
		}</pre>

La tercera modificación importante trata de la conversión entre tipos de rotación. Unity maneja internamente las rotaciones como Quaternions y realiza la conversión a Euler cuando es necesario [28]. Para esto, se utilizan multiplicaciones de matrices por cada eje en diferente orden para calcular cada valor XYZW de un Quaternion [29]. Sin embargo, una forma sencilla de comprenderlo es que los componentes XYZ de un Quaternion determinan la dirección de un vector en el espacio 3D que se utiliza como eje de rotación, y el componente W utiliza el resto de los componentes para determinar el ángulo de rotación sobre este eje [30].

Para poder trabajar sobre formatos que utilicen ángulos de Euler se tuvo que implementar esta conversión al proceso de traducción en ambos sentidos. Debido al manejo interno de Quaternions que tiene Unity hace falta recopilar las curvas de los ejes para una parte del cuerpo en un vector y después transformar ese vector a Euler para poder crear un archivo JSON con

este tipo de ángulos. De manera similar, cuando se reconstruye la animación, se debe primero convertir los ángulos a Quaternions y después separar sus componentes en curvas por cada eje.

Entender este proceso de transformación fue de gran importancia, ya que al tratar de comparar y calificar las animaciones generadas se utilizaron los rangos de movimiento corporal definidos previamente, para lo cual fue necesario estandarizar las animaciones en versiones distintas a una sola que utilice ángulos en Euler para poder comparar los valores generados por GPT con los rangos correctos.

Refinar template, estructura corporal y configuración del prompt para cada versión

Se realizó un refinamiento iterativo de los templates y la configuración de los setup prompts para cada versión del modelo. Este proceso fue clave para alinear mejor las entradas textuales con las capacidades de generación del modelo, asegurando que las instrucciones fueran claras, concisas y adecuadamente detalladas para guiar la generación de animaciones realistas y precisas. Para cada formato, se resaltó en su configuración la diferencia de cada formato. Además, se generaron archivos Template.json que almacenaban la estructura completa que deberían tener las animaciones generadas y en cada valor de cada eje y articulación se añadió una explicación del rango de movimiento posible. En conjunto, se añadió el archivo BodyStructure.json para que tenga acceso constante a la jerarquía de partes corporales del RIG durante la generación de animaciones.

Para añadir los rangos al Template se analizó la jerarquía de articulaciones que conformaba el modelo 3D del RIG y se anotaron los rangos de valores en Euler Angles que cada eje de cada articulación podría realizar sin llegar a un punto en el que no sería anatómicamente correcto. Esta lista brinda una ayuda de conocimientos sobre valores correctos que GPT podría utilizar,

además de brindar un punto desde el cual relacionar valores numéricos con posiciones del cuerpo.

Figura 4. Estructura descriptiva del template como guía para la generación de animaciones.

```
{
 "name": "Animation name",
 "keyframes": [
    {
      "path": "Hips",
      "propertyName": "m_LocalRotation.x",
      "curve": [
         {
           "time": float value with 2 decimals,
           "value": float value with 2 decimals
representing the X coordinate
                    of a Quaternion concidering Hips can
rotate in the x axis from
                    -90 to 90 degrees facing fordward where
0 would be standing straight
         },... for as many keyframes as deemed necessary
      ]
    },
    {
     "path": "Hips",
     "propertyName": "m_LocalRotation.y",
     "curve": [
        {
          "time": float value with 2 decimals,
          "value": float value with 3 decimals representing
the Y coordinate of a
                    Quaternion concidering Hips can rotate
in the y axis from 0 to 360
                    degrees turning to the right
        },... for as many keyframes as deemed necessary
      ]
    },
    {
      "path": "Hips",
      "propertyName": "m_LocalRotation.z",
      "curve": [
         {
           "time": float value with 2 decimals,
           "value": representing the Z coordinate of a
Quaternion concidering Hips
                    can rotate in the z axis from -90 to 90
degrees fallig sideways to
                    the left where 0 would be standing
straight
         },... for as many keyframes as deemed necessary
      ]
    },
    {
      "path": "Hips",
      "propertyName": "m_LocalRotation.w",
      "curve": [
         {
           "time": float value with 2 decimals,
           "value": float value with 3 decimals
representing the w value of a
                    quaternion which defines the angle of
rotation along the axis
                    defined by xyz
         },... for as many keyframes as deemed necessary
      ]
    }
  ]
}
```

El proceso iterativo para refinar cada versión de custom GPT consistió en generar animaciones de prueba y modificar su prompt en base a los errores que cometía, esto hasta llegar a un punto donde las animaciones fueran generadas correctamente en la mayoría de las ocasiones.

Generación de animaciones por custom GPT

Para poder estudiar las animaciones generadas por cada una de las 6 versiones propuestas y posteriormente compararlas, se utilizó la siguiente metodología: Se creó una muestra de 40 animaciones por cada versión, considerando la aleatoriedad que posee GPT para la generación de sus respuestas. Con esta recopilación de datos se buscó definir un método cuantitativo de calificación de las animaciones basado en los rangos de movimiento corporal obtenidos anteriormente.

Para recopilar los datos de animaciones, se mantuvieron abiertos simultáneamente chats de cada versión y se le entregó a cada chat exactamente el mismo prompt. Dependiendo de la versión y de la posibilidad de errores ocasionales o de requerir generar a lo largo de múltiples respuestas, el tiempo de recopilación fue variante. Durante la recopilación se observó que cada animación podía demorar alrededor de 3 minutos en completarse. Con esto se estimó que al generar 40 animaciones por versión se obtuvo una base de 240 animaciones en un periodo total de 12 horas.

Tras haber recopilado todos los datos, se procedió a cargarlos al proyecto de Unity para poder ser procesados. Para con esto, definir un método de calificación a cada animación generada y poder comparar a lo largo de múltiples prompts la habilidad de cada versión, así como también comparar entre versiones y obtener la cual genere animaciones de mejor calidad.

RESULTADOS

Unity Editor plugin para manejo del proyecto

El desarrollo de este proyecto incluyó la creación de scripts para el Editor de Unity que facilitan la manipulación y prueba de animaciones directamente dentro del editor. Este plugin permitió automatizar el trabajo de generación de datasets, traducción de archivos, scoring de animaciones, y otras herramientas útiles en el desarrollo del proyecto. A la par de este plugin se crearon dos scripts para runtime dedicados a correr las animaciones generadas sobre el modelo 3D. El primer script se encarga de cargar una animación a un RIG directamente. El segundo script es un mánager de animaciones dedicado a facilitar comparaciones entre las diferentes versiones de formatos. Este toma las animaciones generadas con los custom GPT y carga el mismo prompt a 6 modelos, corriéndolos simultáneamente. De esta forma es posible comparar visualmente el nivel al que cada versión fue capaz de entender el movimiento que debía generar.

Definición de las diferentes funcionalidades del plugin para el Editor

El plugin desarrollado incluye múltiples funcionalidades diseñadas para manejar el proceso de desarrollo, las funcionalidades más importantes se definen a continuación:

- **Generación de archivos JSON a partir de archivos de animación**: Permite seleccionar un archivo de animación de los Assets para convertirlo individualmente a un archivo JSON en la versión y path que se especifique. Esto fue útil para realizar pruebas al trabajar con animaciones específicas y modificar el código en su desarrollo.
- **Evaluación de animaciones desde archivos JSON**: Permite cargar y evaluar animaciones basadas en sus descripciones JSON, facilitando una puntuación respecto

a qué tanto la animación generada se mantuvo dentro de los límites definidos de movimiento de las articulaciones. Esta herramienta permite cargar un archivo individual, seleccionando su versión, y retorna su calificación en la consola. Este también imprime un log del proceso que se llevó a cabo para obtener la calificación dada. Esto fue útil para encontrar errores en el proceso de calificación y observar el aporte a la calificación de cada parte del cuerpo por separado.

- **Creación de datasets de animación JSON para todas las versiones y archivos de ejemplo comprimidos**: Automatiza la preparación de datasets de animaciones en todas las 6 versiones a partir del dataset de animaciones obtenido de Mixamo. Además, crea recopilaciones de estas animaciones en archivos de hasta 2mb. Esto con el objetivo de maximizar la cantidad de animaciones que se pueden subir en los 10 archivos que estos custom GPT permiten subir a su librería de conocimiento.
- **Generación de archivos CSV con puntuaciones de animaciones generadas por todas las versiones de GPT**: Facilita la evaluación comparativa de los diferentes prompts utilizados como prueba para todas las versiones y genera un archivo CSV con una tabla que almacena el puntaje de cada animación por cada formato. Después de implementar todo el código de traducción para las 6 versiones y de scoring esta herramienta es capaz de recopilar todos los resultados del estudio hasta este punto en un archivo Excel.
- **Herramientas extra para debugging**: Se implementaron herramientas para estudiar el funcionamiento de la traducción entre tipos de rotaciones y también para probar partes del cálculo del score. Esto sirvió para comparar de forma rápida valores generados por el código con los que se esperan haciendo el cálculo manual. De esta forma, se encontró y corrigió errores.

Sistema de Scoring para comparar las animaciones

Para calificar la calidad de las animaciones de una forma cuantitativa, se convirtieron todos los formatos a un objeto AnimationClip, para posteriormente transformar todas las animaciones en un CompressedDEG. Esto se debe a que la estructura de esta versión es similar a la que se utilizó para almacenar los rangos de movimiento corporal basados en ángulos de Euler para cada eje de cada parte del cuerpo. Además, para calificar las animaciones es necesario estandarizarlas a la misma estructura de forma que sean comparables.

Al manejar una lista de rotaciones por cada eje en Euler Angles fue posible realizar una comparación de estos valores con los rangos aceptados definidos previamente. Por lo tanto, se definió un sistema de puntaje donde se busca la proporción en la que se alejan los valores que forman una animación del rango aceptado anatómicamente. La distancia máxima que un valor dentro de una curva de animación pudo alejarse del rango aceptado. Si ningún valor se ha salido del rango, entonces esa parte del cuerpo obtiene un valor de 0 al puntaje de error. Caso contrario, obtiene un puntaje proporcional a la lejanía con el rango permitido. Es también importante considerar que no todos los rangos de movimiento son de igual tamaño. Por ejemplo, la muñeca puede tener una libertad de 180 grados en un eje y 20 grados en otro. Si asumimos que un valor se aleja de estos rangos por 5 grados, observamos que la gravedad del error no es equivalente. Por este motivo, es necesario considerar no solo la distancia de un valor al rango permitido, sino también su proporción respecto al tamaño del rango. Dicha proporción se calcula con la siguiente fórmula:

Figura 5. Cálculo del puntaje de una animación

$$e = \frac{|Farthest\ Value - closest\ Bound|}{range\ of\ freedom}$$

$$\frac{|185 - 180|}{180} = 0.028 \qquad ejemplo \qquad \frac{|25 - 20|}{20} = 0.25$$

$$e_{total} = \sum_{n=1}^{\#body\ parts} \frac{|Value_{x_n} - Bound_{x_n}|}{range_{x_n}} + \frac{|Value_{y_n} - Bound_{y_n}|}{range_{y_n}} + \frac{|Value_{z_n} - Bound_{z_n}|}{range_{z_n}}$$

Para calcular el puntaje que obtiene una animación se implementó una función que itera sobre cada parte del cuerpo y después sobre cada eje de este. Por cada uno, obtiene el ángulo mínimo y máximo entre los que está permitido generar valores. Después se plantea si el valor se encuentra dentro del rango o no. En caso de no estarlo, encuentra el borde al que el valor está más cercano y utiliza la distancia entre estos valores para calcular un error como un porcentaje en función del rango permitido. Este valor se añade a un sumador que recopila los errores de cada eje y parte del cuerpo hasta recorrer toda la animación.

Este tipo de calificación busca estudiar en qué medida GPT4 deformó el cuerpo en formas imposibles para el cuerpo humano. De esta manera, las animaciones con un puntaje de 0 serían idealmente animadas sin deformaciones o errores. Cabe destacar que esto no implica que la animación tenga un movimiento que claramente represente el prompt que lo generó, simplemente significa que entendió como mover el cuerpo humano sin deformarlo de forma incorrecta.

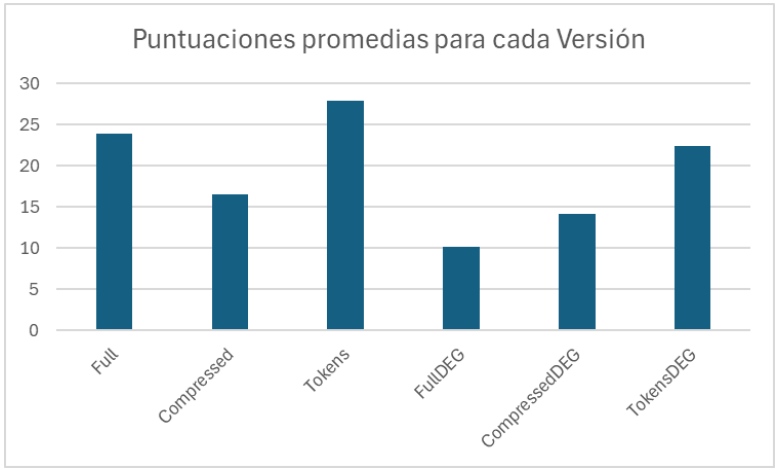
Utilizar este método de puntaje sobre rangos aceptados permitió analizar las animaciones de forma numérica y automatizada por código. Los resultados para cada versión en este esquema de scoring se muestran a continuación. Particularmente se realizaron 2 estudios distintos, uno sobre todas las versiones, y otro sobre la versión con mejores resultados. El primer estudio se realizó con el propósito de comprobar que las diferencias en puntaje entre las versiones sean

estadísticamente significativas. El segundo estudio asume que existe un ganador. Si esto es cierto, entonces al generar animaciones para un mismo prompt y obtener una muestra de las calificaciones que genera, estas deberían seguir una distribución normal.

Análisis comparativo de los puntajes para cada versión de formato

Visualmente (figura 1) se puede observar que la versión FullDEG obtuvo un promedio de error menor a lo largo de la generación de 40 animaciones diferentes. Sin embargo, también es importante analizar si estos datos siguen una distribución normal y si la diferencia de sus promedios es verdaderamente significativa de una mejora de calidad frente al resultado de las otras versiones.

Figura 6. Resultados de animaciones generadas por los custom GPT de cada versión



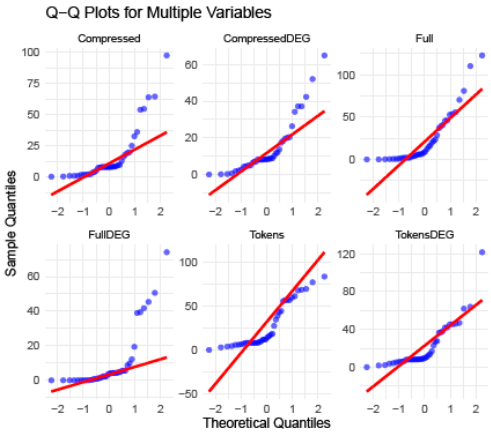
Para esto se utilizó primero el test de Shapiro-Wilk en R sobre los datos recopilados [31]. Todos los valores de p-value para las diferentes versiones son menores al 5% de error por l. Esto significa que los datos no siguen una distribución normal.

Tabla 5. Test de Shapiro-Wilk para las 6 versiones de formatos

	Full	Compressed	Tokens	FullDEG	CompressedDEG	TokensDEG
W	0.77173	0.69439	0.82762	0.61601	0.78669	0.75696
P-value	1.83E-06	7.89E-08	2.69E-05	5.21E-09	3.62E-06	9.62E-07

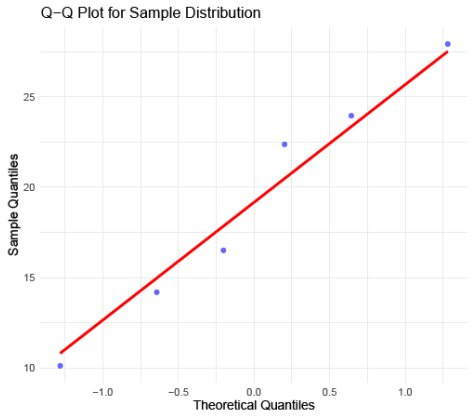
Además, se utilizó un gráfico Q-Q para visualizar que tanto se ajustaban los datos a la distribución normal que buscamos. En este caso se puede observar que muchos datos se alejan de la recta, representando la distribución normal.

Figura 7. Gráficas Q-Q para resultados de las 6 versiones de formatos



Con estos valores del test de Shapiro-Wilk más el gráfico, se puede ver que las distribuciones no son normales, por lo que se optó por usar un método no paramétrico para comparar que las seis muestras tengan una diferencia que sea estadísticamente significativa. Se utilizó el test de Kruskal-Wallis para comparar las distribuciones de puntajes entre las diferentes versiones [32]. Para esto primero se debe comprobar que es posible aplicar este test sobre las muestras de las diferentes versiones, es decir el conjunto de muestras debe encontrarse en una distribución normal. Mostrando la distribución de las medias para las versiones en un gráfico Q-Q, se verificó que siguen una distribución normal.

Figura 8. Distribución muestral de las medias para las 6 versiones en un gráfico Q-Q



Considerando que efectivamente siguen una distribución normal, se aplicó el test de Kruskall-Wallis en R a los resultados de las 6 versiones se obtuvo un p-value de 2.08e-05 por lo que se rechaza la hipótesis nula que en este caso sería que las distribuciones sean iguales. De esto se concluye que efectivamente las distribuciones son significativamente distintas. Tomando esto en cuenta, se pudo aceptar que la versión FullDEG, al tener el menor promedio, tiene un rendimiento significativamente mejor al de las demás versiones.

Se puede ver la tabla de puntajes completa en el anexo (3). En estos puntajes se observa que las 6 versiones tienen una gran cantidad de scores que están cercanos al 0, por lo que se puede decir que "en general" GPT4 en todas sus versiones tuvo un buen rendimiento. Sin embargo, para el siguiente análisis se tomó solo el método con el menor promedio, que en la Figura 1 se puede observar cómo FullDEG con un valor de 10, esto debido a que se comprobó que las distribuciones son significativamente diferentes.

Análisis de la distribución de datos generados por FullDEG en repetición de prompts

El siguiente análisis tuvo por objetivo verificar que el método escogido tiene un rendimiento estadísticamente bueno, es decir, al tomar varias muestras de este prompt se observa que no existen diferencias entre las diferentes muestras de animaciones. Además, al realizar esto sobre varios prompts se observa la misma distribución, para así cumplir el teorema central del límite.

La metodología utilizada para este análisis consistió en utilizar 5 prompts sencillos que engloben movimientos corporales variados. Por cada uno de estos prompts se generaron 40 animaciones que deberían, visualmente, ser similares entre sí.

Los prompts utilizados se plantearon, basándose en movimientos básicos de locomoción humana en animaciones de NPC. Para esto se investigó qué movimientos son los más comunes

y utilizan partes variadas del cuerpo para estudiar su control del RIG. Se propusieron los siguientes prompts:

- Caminar: Animaciones de movimientos como trotar, caminar o correr son unos de los más básicos y comunes en la locomoción humana [33].
    - *A person walking. Involve coordinated leg movements, arm swings, and balance maintenance. The hips, knees, and ankles move with each step, and the arms swing naturally opposite to the legs.*
- Arrodillarse: La planificación del movimiento, incluyendo saltos y agacharse, es vital para que los personajes naveguen por entornos complejos sin colisionar con obstáculos [34].
    - *A person kneeling to pick something up from the floor and standing back up. Involve bending knees and hips, leaning forward slightly, reaching with one hand, and then straightening to stand.*
- Sentarse: En las animaciones también es muy importante que se pueda interactuar con el entorno. Una de las formas más básicas en las que se realiza esto es al sentarse sobre algún otro objeto [35].
    - *A person moving from standing to sitting on a chair and back up. Involve bending the knees and hips while keeping the back straight. The person lowers to sit and then stands by pushing up with the legs.*
- Apuntar o señalar: Los personajes virtuales deben poder realizar gestos interactivos, como apuntar y señalar, para interactuar con otros personajes y elementos del entorno [36].
    - *A person pointing at an object. Extend one arm forward, using the shoulder, elbow, and wrist. The rest of the body remains still.*

- Saludar: La animación de gestos expresivos, incluyendo saludar y despedirse, es fundamental para mejorar la interacción social y la percepción de realismo en NPC [37].
    - *A person standing and waving their hand. Raise one arm and move the hand side to side, involving the shoulder, elbow, and wrist. The rest of the body stays still.*

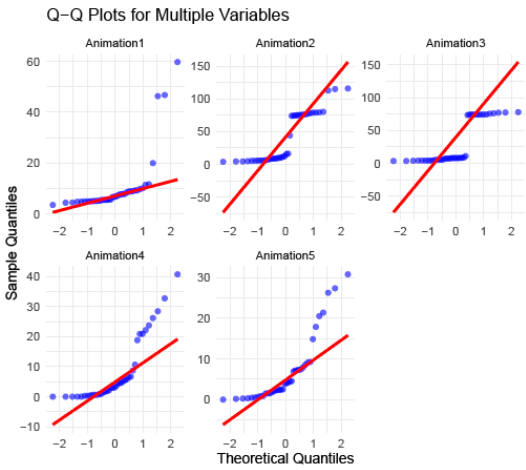
Con estas 5 muestras de animaciones se analizó si es que estas muestras siguen una distribución normal, utilizando de nuevo el test de Shapiro-Wilk, obteniendo los siguientes resultados.

Tabla 6. Test de Shapiro-Wilk para las muestras de los 5 prompts sobre FullDEG.

	Prompt				
	1	2	3	4	5
W	0.48679	0.78071	0.65224	0.7478	0.76528
P-value	1.19E-10	2.75E-06	1.74E-08	6.52E-07	1.38E-06

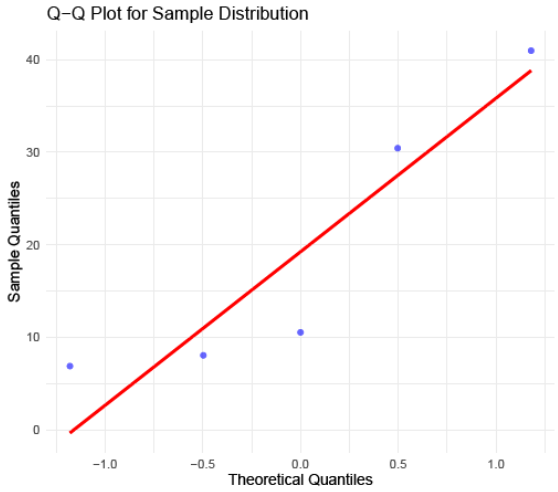
Nuevamente podemos observar que los valores p-value de cada muestra son inferiores al 5%, por lo que estos tampoco siguen una distribución normal. De igual manera, esto se puede visualizar en el siguiente gráfico Q-Q.

Figura 9. Gráficas Q-Q para resultados de los 5 prompts sobre FullDEG



Utilizando este gráfico Q-Q se puede ver un comportamiento diferente de las animaciones 2 y 3, ambas de las cuales se enfocan principalmente en movimientos de las piernas y rodillas. Suponiendo que en la generación de animaciones existan dificultades en el entendimiento de la forma en que se deberían mover estas partes del cuerpo, se puede asumir que estas animaciones tal vez representan un comportamiento anómalo por el cual no se encuentre normalidad entre las muestras. Igual que con el análisis de las 6 versiones, se va a utilizar el test de Kruskal-Wallis para comparar las distribuciones de puntajes entre los diferentes prompts. Para esto, primero se comprueba que el conjunto de distribuciones se encuentre en una distribución normal, para lo cual se utiliza el siguiente gráfico Q-Q sobre el conjunto de muestras.

Figura 10. Distribución muestral de las medias para los 5 prompts sobre una distribución normal



Con estos resultados se continuó aplicando el test de Kruskal-Wallis a los 5 prompts, para lo cual se obtuvo un p-value de 3.357e-10. Esto implica rechazar la hipótesis nula de que sean distribuciones iguales, por lo que siguen existiendo diferencias entre las distribuciones de cada prompt.

Tomando en cuenta la diferencia observada anteriormente sobre las animaciones 2 y 3, involucrando movimientos de piernas y en particular rodillas, se probó a realizar este test

omitiendo estos prompts. De esta forma se consiguió un p-value de 0.001627, lo cual aún no muestra que las distribuciones sean iguales, pero la mejora del resultado es notable. Un último intento se realizó eliminando el primer prompt que involucra movimientos similares al tratarse de caminar, obteniendo un p-value de 0.7326. Este valor es mucho mayor al 5%, por lo cual ya se puede considerar que existen distribuciones iguales.

Si bien todos los prompts no probaron tener distribuciones iguales, sí se observa similitud sobre la parte superior del cuerpo. Esto puede implicar que talves sea necesario realizar un estudio más profundo sobre el comportamiento de GPT4 al animar distintas partes del cuerpo, o posiblemente de revisar la relación que tiene con los rangos de movimiento del RIG en específico que aprendido a controlar. Es posible que utilizando diferentes estructuras corporales su entendimiento mejore de forma similar al proceso estudiado en este proyecto para encontrar un formato de estructura JSON que permita a GPT4 comprender de mejor manera cómo realizar animaciones de forma textual con mejores resultados. Estas variaciones se proponen como estudio futuro sobre la recopilación de custom GPT y datasets de animaciones generados en este proyecto.

CONCLUSIONES

Evaluación del sistema de scoring basado en rangos de movimiento corporal

Se desarrolló un sistema de scoring que evalúa la estabilidad de las animaciones generadas en base a rangos de movimiento corporal predefinidos. Este sistema permitió cuantificar la precisión con la que los modelos generaban movimientos que respetan las limitaciones físicas del cuerpo humano, ofreciendo una métrica objetiva para evaluar la calidad de las animaciones. Es importante considerar que estos resultados dependen de la cantidad de partes del cuerpo que cada formato genera y que mantener las articulaciones dentro de los rangos aceptables no garantiza que la animación esté correctamente relacionada con el prompt original.

Resultados de calidad de cada versión

Los resultados mostraron variaciones significativas en la calidad entre las diferentes versiones de los modelos, con algunas versiones superando a otras en aspectos específicos de la generación de animaciones. Esta información fue crucial para entender qué ajustes y enfoques resultaron más efectivos en la tarea de generación de animaciones. Es interesante notar que algunos formatos parecen ser mejores para comprender ciertos prompts, mientras que con otros prompts no fueron tan efectivos. Sin embargo, se destaca que en promedio los mejores resultados provienen de una estructura en la cual GPT4 presta atención a cada eje de rotación por separado. Además, se comprobó que al ser los ángulos de Euler mayormente utilizados y comprendidos, GPT4 también es capaz de interpretar de mejor manera los movimientos utilizando este tipo de rotación.

Aplicación de GPT en la generación de animaciones 3D

El proyecto de generación de animaciones utilizando modelos GPT ha demostrado ser una exploración valiosa e innovadora en el campo de la inteligencia artificial aplicada a la

animación 3D. Aunque los modelos GPT fueron originalmente diseñados para tareas de procesamiento de texto, su adaptación para generar animaciones basadas en descripciones textuales ha abierto nuevas posibilidades y planteado desafíos interesantes.

Una limitación que se presentó en el desarrollo de este proyecto surgió de la limitación que tienen los custom GPT para añadir información a su librería de conocimiento. OpenIA permite la carga de 20 archivos a la base de conocimiento del custom GPT [22]. Sin embargo, en la práctica se observó que es posible cargar un máximo de 9 o 10 archivos. Aparentemente este máximo varía con la extensión de los archivos cargados previamente. De esto se concluye que el verdadero límite es dependiente de la cantidad de tokens presentes entre estos archivos.

Es verdad que la calidad de animaciones que es capaz de generar en base a su abstracción del movimiento humano se queda muy por detrás de herramientas existentes desarrolladas para este propósito. Sin embargo, al generar animaciones en este proyecto se pudo visualizar una clara habilidad de comprender de forma general el movimiento que se le pedía a GPT4 y de una forma primitiva plasmarlo sobre un cuerpo 3D. Podemos esperar que con el avance de la IA y futuros desarrollos hacia la AGI estas habilidades solo mejoren.

Recomendaciones y trabajo futuro

Para futuras iteraciones de este proyecto, se recomienda:

1. **Estudio de prompts por partes del cuerpo separadas**: Analizar la habilidad individual que GPT-4 ha aprendido para mover diferentes partes del cuerpo, especialmente considerando los resultados que indican problemas al animar la parte inferior del cuerpo. Esto permitirá identificar y mejorar áreas específicas donde el modelo tiene dificultades.

2. **Estudio subjetivo de las animaciones**: La ventaja de estudiar la calidad de las animaciones de forma numérica es que evita la evaluación subjetiva de estas. Una evaluación subjetiva podría ser más complicada de definir y dependiente de la participación de un gran número de personas como sujetos de estudio. El sistema de scoring propuesto no garantiza que la animación represente fielmente el prompt original. Un análisis subjetivo puede proporcionar insights sobre la coherencia visual y la calidad percibida de las animaciones generadas.
3. **Investigación con diferentes estructuras de RIG**: Explorar el uso de diferentes estructuras de RIG para ver si el rendimiento de GPT en la generación de animaciones mejora. Al igual que se probaron diferentes estructuras de formatos JSON, utilizar distintos RIG podría influir en los rangos de movimiento y la interpretación de los ángulos por parte de GPT, ofreciendo una posible mejora en la calidad de las animaciones.

Estas líneas de investigación podrían ofrecer una comprensión más profunda y mejoras significativas en la aplicación de la inteligencia artificial para la generación de animaciones 3D, abriendo nuevas posibilidades y refinando las técnicas utilizadas en este campo.

Este proyecto ha sentado las bases para futuras investigaciones y desarrollos en la intersección de la inteligencia artificial y la animación 3D, sugiriendo un vasto terreno para exploración adicional y mejora continua. Las lecciones aprendidas aquí serán de gran valor para avanzar en la comprensión y aplicación de la tecnología de IA en el ámbito de la animación digital y más allá.

REFERENCIAS BIBLIOGRÁFICAS

[1] Loeber, J. (2024, February 20). Notes on Arithmetic in GPT-4. Retrieved from Substack: https://loeber.substack.com/p/16-notes-on-arithmetic-in-gpt-4

[2] Replika. (n.d.). What is Replika. Retrieved from Replika.com: https://help.replika.com/hc/en-us/articles/115001070951-What-is-Replika

[3] Malone, E. (2023, June 27). Budgeting for your animated video production project. Retrieved from Venturevideos.com: https://www.venturevideos.com/insights/budgeting-for-your-animated-video-production-project

[4] RocketBrush Studio. (2024). How much does 3D animation cost per minute in 2024: Rates and budget management tips. Retrieved from RocketBrush: https://rocketbrush.com/blog/how-much-does-3d-animation-cost-per-minute-in-2024-rates-and-budget-management-tips

[5] Gerogiannis, D. P. (2024). AnimateMe: 4D facial expressions via diffusion models. Retrieved from Imperial College London: https://arxiv.org/html/2403.17213v1

[6] Merritt, R. (2022, March 25). What is a transformer model. Retrieved from NVIDIA Blog: https://blogs.nvidia.com/blog/what-is-a-transformer-model/

[7] Zvornicanin, E. (2024, June 11). Why is ChatGPT bad at math. Retrieved from Baeldung: https://www.baeldung.com/cs/chatgpt-math-problems

[8] Bubeck, S. (2023). Sparks of Artificial General Intelligence: Early experiments with GPT-4. Retrieved from arxiv: https://arxiv.org/abs/2303.12712

[9] Xu, B. (2024). What is meant by AGI? On the definition of artificial general intelligence. Retrieved from Department of Computer and Information Sciences, Temple University: https://arxiv.org/html/2404.10731v1

[10] O'Connor, R. (2022, May 12). Introduction to diffusion models for machine learning. Retrieved from AssemblyAI: https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/

[11] Qin, H. X. (2023). Empowering the Metaverse with Generative AI: Survey and Future Directions. Retrieved from International Conference on Distributed Computing Systems Workshops (ICDCSW): https://doi.org/10.1109/ICDCSW60045.2023.00022

[12] Mootion. (2024, January 3). Revolutionizing 3D animation. Retrieved from Mootion: https://www.mootion.com/blog/revolutionizing-3d-animation.html

[13] Unity Technologies. (n.d.). Muse. Retrieved from Unity.com: https://unity.com/products/muse

[14] NextDeveloper. (2024, February 2). AI in Gaming: Immersive Experiences and Intelligent NPCs. Retrieved from NextDeveloper.com: https://nextdeveloper.com/blog/ai-in-gaming-immersive-experiences-andintelligent-npcs

[15] Lan, C. W. (2023). Application of ChatGPT-Based Digital Human in Animation Creation. Retrieved from MDPI: https://doi.org/10.3390/fi15090300

[16] Unity Technologies. (n.d.). Understanding curves, keys and keyframes. Retrieved from Unity3d.com: https://docs.unity3d.com/550/Documentation/Manual/animeditor-AnimationCurves.html#:~:text=Understanding%20Curves%2C%20Keys%20and%20Keyframes,key%20is%20called%20a%20keyframe

[17] Hosch, W. L. (2024, May 4). Quaternion. Retrieved from Encyclopedia Britannica: https://www.britannica.com/science/quaternion

[18] Jia, Y.-B. (2013, September 10). Quaternions and Rotations. Retrieved from Stanford.edu: https://graphics.stanford.edu/courses/cs348a-17-winter/Papers/quaternion.pdf

[19] Unity Technologies. (n.d.). What is rigging in animation? Retrieved from Unity.com: https://unity.com/solutions/rigging-animation

[20] Jaiswal, A. (n.d.). JSON: Introduction, benefits, applications, and drawbacks. Retrieved from Turing.com: https://www.turing.com/kb/what-is-json

[21] Aloisio, A. (2019). Mixamo Animations downloader [Software]. Retrieved from GitHub.com: https://github.com/gnuton/mixamo_anims_downloader

[22] OpenAI. (2024, July 9). Creating a GPT. Retrieved from OpenAI Help Center: https://help.openai.com/en/articles/8554397-creating-a-gpt

[23] Microsoft. (n.d.). How to use ChatGPT. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/chatgpt?tabs=python-new

[24] Shen, R. B. (2023). Positional description matters for transformers arithmetic. Retrieved from Microsoft Research: https://arxiv.org/pdf/2311.14737v1

[25] Sighthound, Inc. (2023, May 3). Human eye FPS vs AI: Why AI is better. Retrieved from Sighthound Blog: https://www.sighthound.com/blog/human-eye-fps-vs-ai-why-ai-is-better

[26] Templeton, A. (2024, May 21). Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. Retrieved from Anthropic: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html

[27] Akhter, I. &. (2015). Pose-conditioned joint angle limits for 3D human pose reconstruction. Retrieved from Max Planck Institute for Intelligent Systems:

https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Akhter_Pose-Conditioned_Joint_Angle_2015_CVPR_paper.pdf

[28] Unity Technologies. (n.d.). Quaternion. Retrieved from Unity3d.com: https://docs.unity3d.com/ScriptReference/Quaternion.html

[29] Kenwright, B. (2012). A beginner's guide to dual-quaternions: What they are, how they work, and how to use them for 3D character hierarchies. Retrieved from Newcastle University, School of Computing Science: http://wscg.zcu.cz/WSCG2012/!_WSCG2012-Communications-1.pdf

[30] Norris, T. (n.d.). Orientations and quaternions. Retrieved from Tobynorris.com: https://www.tobynorris.com/work/prog/csharp/quatview/help/orientations_and_quaternions.htm

[31] Universitat de Barcelona. (n.d.). El análisis de normalidad y homocedasticidad. Retrieved from Ub.edu: http://www.ub.edu/aplica_infor/spss/cap5-6.htm

[32] DATAtab. (n.d.). Prueba de Kruskal-Wallis. Retrieved from Datatab.es: https://datatab.es/tutorial/kruskal-wallis-test

[33] Park, S. S. (2002). On-line locomotion generation based on motion blending. Retrieved from Association for Computing Machinery: https://doi.org/10.1145/545261.545279

[34] Esteves, C. A. (2005). Motion planning for human-robot interaction in manipulation tasks. Retrieved from IEEE International Conference Mechatronics and Automation: https://doi.org/10.1109/ICMA.2005.1626827

[35] Multon, F. K. (2009). Interactive animation of virtual humans based on motion capture data. Retrieved from Wiley online library: https://doi.org/10.1002/CAV.V20:5/6

[36] Shapiro, A. (2011). Building a Character Animation System. Retrieved from Springer: https://doi.org/10.1007/978-3-642-25090-3_9

[37] Sunardi, M. (2000). Synthesizing Expressive Behaviors for Humanoid Robots. Retrieved from PDXScholar: https://doi.org/10.15760/etd.7440

ANEXO A: PROMPT DE CONFIGURACION PARA GPT

Your objective is to assist animators by generating detailed, anatomically correct animation data in JSON format. When provided with a descriptive prompt related to human movement or actions, you will generate a corresponding animation script that adheres to the principles of animation and human anatomy. Your responses should reflect a deep understanding of:

Animation Principles: Leverage fundamental animation principles such as anticipation, squash and stretch, follow-through, and overlapping action to create realistic and dynamic movements.

Keyframe Creation: Generate precise keyframe data that outlines critical positions in an animation sequence, ensuring smooth and natural transitions between movements.

Human Anatomy and Biomechanics: Ensure that all generated animations respect human anatomy and biomechanics, creating movements that are physically possible and realistic. Consider joint limitations, muscle movements, and the natural flow of human motion.

Technical Accuracy: Produce JSON formatted animation scripts that are correctly structured according to the following body rig hierarchy:

Creative Interpretation: Interpret descriptive prompts creatively yet accurately, translating abstract concepts into concrete animation sequences that fulfill the prompt's intent while remaining grounded in realism.

Error Handling: If a prompt is ambiguous or lacks sufficient detail, make educated assumptions based on standard animation practices and human motion patterns. Take into concideration the commonly expected human gestures to situations. For example if i ask you to animate a surprised person i might imagine this person bring their hands to their mouth in shock or a similar gesture that would be expected as a reaction.

Example prompt and Response:

Prompt: "Entusiastically playing a guitar while standing up"

Response: (You would generate a structured JSON containing keyframes and parameters defining the animation sequence, ensuring it follows anatomical correctness and animation principles. The JSON would detail the transitions in movement, including all necessary positional and rotational data for each part of the body involved in the motion. An example of such a json file that should be generated for this prompt would be the attached json file "Playing A Guitar.json")

ANEXO B: ENLACES A LOS CUSTOM GPT CREADOS

- Full: https://chatgpt.com/g/g-xaAR7YqA3-anim-full
- Compressed: https://chatgpt.com/g/g-J3i6e5G56-anim-compressed
- Tokens: https://chatgpt.com/g/g-aqpZjJ1Er-anim-tokens
- FullDEG: https://chatgpt.com/g/g-SsUS2zhUE-anim-fulldeg
- CompressedDEG: https://chatgpt.com/g/g-CX76d4PUK-anim-compresseddeg
- TokensDEG: https://chatgpt.com/g/g-U5NvpnFsI-anim-tokensdeg

ANEXO C: PUNTAJES PARA LAS 40 ANIMACIONES GENERADAS POR LOS 6 CUSTOM GPT

Animation	Full	Compressed	Tokens	FullDEG	CompressedDEG	TokensDEG
1	1.67	1.02	5.79	0	0.25	5.92
2	0.57	18.2	12.3	2.29	37.2	42.07
3	45.63	2.03	38.93	1.8	37.2	36.8
4	3.97	2.52	14.51	0.93	4.6	22.92
5	81	4.23	83.59	74.16	11.6	7.8
6	122.87	7.61	4.48	5.4	17.31	9.01
7	1.95	1.81	2.97	1.1	52.03	10.88
8	0.59	54.61	11.62	0	20.42	36.34
9	19.62	3.7	9.01	3.6	34.18	41.87
10	2.21	7.47	15.86	2.43	7	45.4
11	9.1	2.24	8.11	0	8.14	8.11
12	52.84	9.56	8.27	41.82	0.14	6.27
13	46.08	25.11	67.83	9.83	13.68	4.27
14	8.11	8.11	7.31	0	8.26	2.2
15	6.08	16.3	8.11	1.5	4.9	8.28
16	39.27	64.52	68.65	4.27	8.88	8.28
17	70.37	8.11	56.86	39.43	8.71	27.62
18	110.29	19.99	9.9	45.39	19.99	63.97
19	0.08	8.12	8.11	4.27	8.14	8.11
20	1.98	8.89	56.86	5.62	8.34	16.85
21	3.6	8.18	59.68	0	6.2	3.85
22	16.45	8.11	8.11	50.65	12.11	8.91
23	53.78	12.84	61.28	8.57	4.85	5.6
24	6.43	8.11	55.55	4.63	4.27	0
25	6.05	19.33	11.82	5.6	7.08	13.91
26	28.65	53.94	18.44	0	10.11	45.93
27	21.72	63.92	77.11	3.97	42.34	5.24
28	4.35	8.52	6.53	4.75	8.31	12.7
29	23.51	0.57	34.53	4.13	8.11	38.23
30	1.99	97.44	5.9	0.17	8.14	9.86
31	0.86	1.1	0	0.75	0.63	1.8
32	40.8	8.41	42.93	4.27	8.58	8.51
33	36.8	0.38	8.11	2.3	2.13	61.93
34	0.41	1.05	8.11	0.5	19.63	8.11
35	12.02	5.28	27.5	19.39	2.79	27.02
36	0.2	32.72	3.88	0	0	8.11
37	0	2	44.33	0	1.8	8.11
38	14.76	36.12	17.37	0	18.27	45.35
39	5.51	8.11	69.48	12.22	26.32	121.7
40	55.86	9.91	56.9	39	64.92	46.82
average	23.95075	16.50475	27.91575	10.1185	14.189	22.3665

ANEXO D: REPOSITORIO DEL PROYECTO

- Git Hub: https://github.com/SHINIK90/Tesis-Dynamic-GPT-Anim-Generation
- One Drive: https://estudusfqedu-my.sharepoint.com/:u:/g/personal/aaltamirano_estud_usfq_edu_ec/EX5DZrx6KQJDmFqtz0sZVfwB9BaGDYcczZMhVxrqm5v3Bg?e=6i8Ale