Generación narrativa de sistemas solares en Unity
Narrative Generation of Solar Systems in Unity

Trabajo de Fin de Grado
Curso 2023–2024

Autor
Ángel López Benítez

Director
Carlos León Aznar

Grado en Desarrollo de videojuegos
Facultad de Informática
Universidad Complutense de Madrid

Generación narrativa de sistemas solares en Unity
Narrative Generation of Solar Systems in Unity

Trabajo de Fin de Grado en Desarrollo de videojuegos

Autor
Ángel López Benítez

Director
Carlos León Aznar

Convocatoria: *Junio 2024*

Grado en Desarrollo de videojuegos
Facultad de Informática
Universidad Complutense de Madrid

2024

Dedicatoria

A ese hombre que después de todo decidió intentarlo.

Agradecimientos

A Carlos, por su inigualable manera de dirigir el trabajo. A Daniel Illanes por su comentario crítico y comedido. A mis padres, por comprarme todos aquellos libros hace ya algún tiempo.

Y por supuesto a Eva, por asegurarme un espacio cómodo y limpio para el trabajo.

Resumen

Generación narrativa de sistemas solares en Unity

La inteligencia artificial se ha ganado un puesto protagonista en la vida de la mayoría de las personas, ni que decir tiene el nivel al que ha llegado en nuestro ámbito profesional. Un papel igual de importante tienen las historias, tanto en nuestra vida como en los videojuegos, es por eso que parece lógico intentar unir ambas con el objetivo de crear una herramienta que simplifique la labor de los diseñadores.

El usuario en concreto podrá conversar con el LLM para crear planetas con sus personajes, gobiernos y recursos. Con el objetivo de implementar una herramienta que ahorre tiempo y esfuerzo tanto a diseñadores como desarrolladores de videojuegos se ha implementado una interfaz que permite al usuario comunicarse con LLM desde el editor de Unity, junto con un módulo que se asegura de que la respuesta del LLM sea correcta. Una vez leida esta respuesta, la herramienta genera un entorno en tres dimensiones a través del cual el usuario puede viajar y explorar, aterrizando y despegando de los distintos planetas.

Para probar el funcionamiento de la herramienta se han realizado una serie de pruebas de usuario, en las cuales los usuarios han interactuado indirectamente con el LLM para generar un sistema solar sacado de su propia imaginación. Consideramos que los resultados del proyecto han sido satisfactorios y que la herramienta cumple su función en la mayoría de los casos.

Palabras clave

Narrativa, generación, inteligencia artificial, videojuego, ChatGPT, diseño de videojuegos, LLM.

Abstract

Narrative Generation of Solar Systems in Unity

Artificial intelligence has earned a prominent place in the lives of most people, not to mention the level it has reached in our professional field. Stories play an equally important role in our lives and in video games, which is why it seems logical to try to combine both with the goal of creating a tool that simplifies the work of designers.

Specifically, the user will be able to converse with the LLM to create planets with their characters, governments, and resources. To implement a tool that saves time and effort for both game designers and developers, an interface has been implemented that allows the user to communicate with the LLM from the Unity editor, along with a module that ensures the LLM's response is correct. Once this response is read, the tool generates a three-dimensional environment through which the user can travel and explore, landing and taking off from different planets.

To test the functionality of the tool, a series of user tests have been conducted, in which users have indirectly interacted with the LLM to generate a solar system from their own imagination. We consider the project's results to be satisfactory and believe that the tool fulfills its purpose in most cases.

Keywords

Narrative, generation, artificial intelligence, video game, ChatGPT, game design, LLM.

Índice

1. Introducción **1**
1.1. Motivación 2
1.2. Hipótesis 3
1.3. Objetivos 3
1.4. Metodología de trabajo 4
1.5. Plan de trabajo 4

2. Estado de la Cuestión **7**
2.1. Aprendizaje automático 9
2.2. Deep learning 10
2.2.1. Componentes básicos de una red neuronal 10
2.2.2. Proceso de entrenamiento 11
2.3. Transformers 11
2.4. Large Language Models (LLM) 13
2.5. Prompt engineering 15
2.6. GPT y ChatGPT 16
2.7. Aprendizaje por refuerzo 18
2.8. Videojuegos espaciales de exploración 19

3. Arquitectura general del generador y Comunicación con el LLM **21**
3.1. Descripción general del proyecto 21
3.2. Modulo de comunicación con el modelo de lenguaje 22
3.3. Método de validación del esquema de entrada 27
3.4. Validación narrativa del resultado del LLM 29

4. Generación del sistema solar con simulación física **31**
4.1. Generación del sistema 34
4.2. Simulación física y exploración del sistema 35

5. Experimentos y análisis resultados **49**
5.1. Descripción del experimento 49
5.2. Cuestiones planteadas a los usuarios después del experimento 50

5.3. Resultados de las encuestas 51
5.4. Análisis de resultados y alcance de la herramienta 58
5.4.1. Alcance técnico 58
5.4.2. Alcance narrativo 59

6. Discusión **61**

7. Conclusiones, y trabajo futuro **63**
7.1. Conclusiones 63
7.2. Trabajo futuro 64

Introduction **67**

Conclusions and Future Work **69**
7.3. Conclusions 69
7.4. Future Work 70

Bibliografía **73**

Índice de figuras

1.1. Diagrama de gant en el que se muestra la planificación incial del proyecto dividida por meses 6

2.1. Diagrama de la estructura de una red neuronal extraido del artículo Deep Learning in Diverse Intelligent Sensor Based Systems Zhu et al. (2022)) 11
2.2. Estrcutura de un transformer, figura extraida del artículo *Attention is all you need* (Vaswani et al. (2017)). 13
2.3. Ecuación Scaled Dot-Product Attetion 13
2.4. Ecuación de Multi-Head Attetion 14
2.5. Ejemplo de input tomado desde ChatGPT, en este prompt se le da menos contexto al modelo 14
2.6. Ejemplo de input tomado desde ChatGPT, en este prompt se le da más contexto al LLM. 15
2.7. Resultados extraidos del estudio Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs (Wang et al. (2024)) 16

3.1. Captura de pantalla de la modificación del editor de Unity de tal manera que podamos conversar con el ChatGPT 22

4.1. Diagrama en el que se muestra la generación para cada tipo de planeta 33
4.2. Diagrama de la simulación esperada. Dónde el objeto central reprensenta la estrella, los círculos rojo, azul marino, naranja y azul celeste representan los cuatro tipos de planetas, y los grises sus correspondientes satélites 35
4.3. Ejecución de ejemplo del programa 37
4.4. Ejemplo de la información que puede ver el usuario al acercarse a uno de los planetas de la simulación. 37
4.5. Captura de pantalla de la escena correspondiente a los planetas de fuego 38
4.6. Captura de pantalla de la escena correspondiente a los planetas de agua 38
4.7. Captura de pantalla correspondiente a los planetas de hielo 39

4.8. Ejemplo de la interfaz donde el usuario puede ver las características del sistema solar 39
4.9. Ejemplo de interfaz donde podemos ver la información del planeta . . 40
4.10. Ejemplo de la interfaz que muestra las razas del sistema solar 41
4.11. Ejemplo de la interfaz que muestra los personajes del sistema solar . . 41
4.12. Ejemplo de la interfaz que muestra las características de cada uno de los personajes. 42
4.13. Icono empleado para designar los planetas de fuego en la interfaz mostrada en la figura 4.8 42
4.14. Icono empleado para desginar los planetas de agua en la interfaz mostrada en la figura 4.8 42
4.15. Icono empleado para designar los planetas gaseosos en la interfaz mostrada en la figura 4.8 42
4.16. Icono empleado para designar los planetas de hielo en la interfaz mostrada en la figura 4.8 43
4.17. Icono empleado para designar cada uno de los satélites de cada planeta, por cada satélite que tenga el planeta aparecerá un icono al lado del texto satellites que aparece en la figura 4.9 43
4.18. Icono empleado para indicar que el planeta es de fuego, este icono aparece al lado del texto Planet Type que podemos ver en la figura 4.9 cuando el planeta es de fuego 43
4.19. Icono empleado para indicar que un planeta es de agua, este icono aparece al lado del texto Planet type que podemos ver en la figura 4.9 cuando el planeta es de agua 43
4.20. Icono empleado para indicar que el planeta es gaseoso, este icono aparece al lado del texto Planet type que podemos ver en la figura 4.9 cuando el planeta es gaseoso 43
4.21. Icono empleado para indicar que el planeta es de hielo, este icono aparece al lado del texto Planet Type que podemos ver en la figura 4.8 cuando el planeta es de hielo 43
4.22. Icono empleado para indicar que un planeta es del mismo territorio que el que está inspeccionando. Este icono aparece al lado del texto Relations en la posición correspondiente al planeta en la interfaz que aparece en la figura 4.9. Es decir, si estamos inspeccionando el planeta 0 y este icono se encuentra en la posición 2, eso significa que el planeta 0 y el planeta 2 pertenecen al mismo territorio. 44
4.23. Icono empleado para indicar que un planeta está en paz con otro. Este icono aparece al lado del texto Relations en la posición correspondiente al planeta en la interfaz que aparece en la figura 4.9. Es decir, si estamos inspeccionando el planeta 0 y este icono se encuentra en la posición 2, eso significa que el planeta 0 y el planeta 2 están en paz. 44
4.24. Icono empleado para indicar que un planeta está en tensión política con otro. Este icono aparece al lado del texto Relations en la posición correspondiente al planeta en la interfaz que aparece en la figura 4.9 . 44

4.25. Icono empleado para indicar que un planeta está en guerra con otro. Este icono aparece al lado del texto Relation en la posición correspondiente al planeta en la interfaz que aparece en la figura 4.9. Es decir, si estamos inspeccionando el planeta 0 y este icono aparece en la posición 2, eso significa que el planeta 0 y el planeta 2 están en guerra. 44

4.26. Icono empleado para indicar que el gobierno del planeta es una monarquía, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es una monarquía. 44

4.27. Icono empleado para indicar que el gobierno del planeta es una república, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es una república. 45

4.28. Icono empleado para indicar que el gobierno del planeta es una democracia, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es una democracia. 45

4.29. Icono empleado para indicar que el gobierno del planeta es autoriatario fascista, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es autoritario fascista. 45

4.30. Icono empleado para indicar que el gobierno del planeta es autoriatario comunista, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es autoritario comunista. 45

4.31. Icono empleado para indicar que el gobierno del planeta se basa en clanes guerreros, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta se basa en clanes guerreros. 45

4.32. Icono empleado para indicar que el planeta en concreto no está habitado y por lo tanto no tiene gobierno, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el planeta en concreto no está habitado. 45

4.33. Icono empleado para indicar que el planeta en concreto tiene comida como recurso. Este icono aparece al lado del texto Resources en la interfaz de la figura 4.9 cuando el planeta tiene comida como uno de sus recursos. 46

4.34. Icono empleado para indicar que el planeta en concreto tiene minerales como recurso. Este icono aparece al lado del texto Resource en la interfaz de la figura 4.9 cuando el planeta tiene minerales como uno de sus recursos 46

4.35. Icono empleado para indicar que el planeta en concreto tiene magia como recurso. Este icono aparece al lado del texto Resource en la interfaz de la figura 4.9 cuando el planeta tiene magia como uno de sus recursos 46
4.36. Icono empleado para indicar que el planeta en concreto tiene armas como recurso. Este icono aparece al lado del texto Resource en la interfaz de la figura 4.9 cuando el planeta tiene armas como uno de sus recursos 46
4.37. Icono empleado para indicar que el planeta en concreto tiene esclavos como recurso. Este icono aparece al lado del texto Resource en la interfaz de la figura 4.9 cuando el planeta tiene esclavos como uno de sus recursos 46

5.1. Gráfico que muestra los resultados obetenidos en la pregunta de edad de la encuesa demográfica 51
5.2. Gráfico que muestra los resultados obtenidos en la pregunta de sexo de la encuesta demográfica 51
5.3. Gráfico que muestra los resultados obtenidos en la pregunta de experiencia como programador de la encuesta demográfica 52
5.4. Gráfico que muestra los resultados obtenidos en la pregunta de experiencia como diseñador de videojuegos en la encuesta demográfica . . 52
5.5. Gráfico que muestra los resultados obtenidos en la pregunta de conocimientos sobre narrativa de la encuesta demográfica. 52
5.6. Gráfico que muestra los resultados obtenidos en la pregunta de conocimiento sobre Unity de la encuesta demográfica 53
5.7. Primera parte del resultado de la pregunta referente a estudios o trabajo actual de la encuesta demográfica. 53
5.8. Segunda parte del resultado de la pregunta referente a estudios o trabajo actual de la encuesta demográfica. 54
5.9. Gráfico que muestra los resultados de la pregunta referente al parecido que el usuario ha encontrado entre lo que tenía en mente y el resultado de la herramienta. Perteneciente a la encuesta de uso. 54
5.10. Gráfico que muestra los resultados de la pregunta referente a como de satisfecho ha estado el usuario con respecto al resultado de la herramienta. Perteneciente a la encuesta de uso. 54
5.11. Gráfico que muestra los resultados de la pregunta referente a como de útil ha encontrado el usario la herramienta para el desarrollo de videojuegos. Perteneciente a la encuesta de uso. 55
5.12. Gráfico que muestra los resultados de la pregunta referente a como de útil han encontrado los usuarios la herramienta para desarrollar un juego en una game jam. Perteneciente a la encuesta de uso 55
5.13. Gráfico que muestra los resultados obtenidos de la pregunta referente a como de útil han encontrado los usuarios la herramienta a sabiendas de que se mueden alterar las escenas de los planetas. Perteneciente a la encuesta de uso. 55

5.14. Gráfico que muestra los resultados obtenidos de la pregunta referente al tiempo que los usarios creen que hubiesen tardado en generar el contenido que la herramienta les ha facilitado. Perteneciente a la encuesta de uso. 56
5.15. Descripción de lo mejor y lo peor de la herramienta 1. 56
5.16. Descripción de lo mejor y lo peor de la herramienta 2. 56
5.17. Descripción de lo mejor y lo peor de la herramienta 3. 57
5.18. Descripción de lo mejor y lo peor de la herramienta 4. 57
5.19. Descripción de lo mejor y lo peor de la herramienta 5. 57
5.20. Descripción de lo mejor y lo peor de la herramienta 6. 58

Índice de tablas

Capítulo 1

Introducción

"Entre aquellas páginas debía estar yo mismo"
— Ángel de Miguel

Las primeras historias ya se contaban en las pinturas rupestres allá en Altamira y otros muchos sitios del mundo. La narrativa es tan antigua como la humanidad, pasando por Homero, Shakespeare o Tolkien por nombrar algunos. Y conforme evolucionamos como especie, las historias evolucionan a nuestro lado, creando mundos que no existen, contando historias sobre el nuestro propio e iluminándonos el camino conforme avanzamos en el tiempo. Desde la Ilíada y la Odisea al Señor de los anillos, inumerables autores han invertido incotable tiempo de sus vidas en crear historias que conmuevan, ayuden o enseñen al lector. Y ahora, milenios más tarde que el día en el que el primer hombre decidió erguirse para dibujarse a sí mismo canzando un ciervo o un bisonte, nosotros tenemos al alcance de nuestra mano, sin necesidad de erguirnos lo más mínimo, herramientas que nos permiten crear historias con solo una definición, unas cuantas palabras, o un puñado de indicaciones.

Algo antes que esta tecnología, surgió un nuevo medio, llamado el videojuego, y aunque nadie sabe específicar cocnretamente cuando nació, a día de hoy todos reconocen la importancia que ha tomado en la vida de las personas. Y en un mundo que compite constantemente por el tiempo del usuario, la narrativa en forma de manuscritos ha tenido que hacerse a un lado para dejar paso a los nuevos bardos de la narrativa, como pueden ser el cine o, desde hace relativamente poco tiempo, los videojuegos.

Aunque hasta hace más bien poco, por no decir que en parte sigue siendo así, los videojuegos se hayan visto como una pérdida de tiempo, o simplemente simuladores de matar marcianos y otros actos violentos, el medio ha demostrado una y otra vez su potencial para llevar las historias a un nuevo nivel que no habían conseguido ni los libros ni el cine. Esto se debe a que el videojuego no nos cuenta la historia de alguien externo, no vemos al personaje a través de una lente, ni conocemos sus palabras porque estén escritas en unas páginas. Nosotros somos el protagonista, nos metemos directamente en su carne, y en algunos casos, nosotros tomamos sus decisiones y escogemos sus palabras.

Títulos como Red Dead Redemption 2 (Rockstar Games (2018)) o algunos más parecidos a lo que nuestro proyecto propone como Outer Wilds (Annapurna Inter-

active (2019)) nos demuestran que estamos frente a un medio capaz de hacer mucho más que enseñarnos a matar marcianos, que tiene la capacidad de conmovernos y enseñarnos cosas que no se pueden aprender a base de estudiar.

Sin embargo, estas historias también fueron escritas por personas de carne y hueso, escritas en un tiempo dónde pedirle a una máquina que escribiese ella una historia parecía un prodcuto de la imaginación y la cienca ficción. Y ahora, escasos años después del lanzamiento de estos títulos, estamos frente a una tecnología capaz de replicar voces, rostros e incluso de programar ella misma. Y por supuesto, también es capaz de escribir historias.

Por supuesto, como pasó cuándo se inventó el tractor, o cuando Henry Ford presentó el primer coche con un motor de combustión, hubo gente asustada, gente emocionada y los más perjudicados de todos, aquellos cuyo miedo se reducía a perder su trabajo. La inteligencia artificial no ha sido para menos, pues diseñadores gráficos, programadores y compositores musicales, se desbandan en pánico cada vez que esta tecnología hace un avance, y no es para menos, pues lo que alguien conseguía producir en días o semanas de trabajo, hay algoritmos que lo producen en minutos, si no segundos.

Así pues, este trabajo pretende echar un vistazo a esto mismo, pero en el ámbito de la narrativa, intentando atisbar que alcance puede tener una inteligencia artificial como ChatGPT, y si es que finalmente cualquier persona con una trabajo creativo se verá sustituida por un algoritmo, o quizás estemos frente al nacimiento de una herramienta que nos hará la vida más fácil, como fue el motor de combustión, o el tractor.

1.1. Motivación

En el mundo del videojuego, las historias son cada vez más frecuentes y profundas, aunque también se busca cada vez más la satisfacción inmediata, Los videojuegos cada vez se producen con mayor velocidad, los desarollos se acortan y las horas de trabajo se acumulan. En este contexto, tiene sentido tratar de hacer una herramienta que agilice ese trabajo. La generación automática de contenido es un tema ampliamente explorado, desde la generación de niveles en The Binding Of Isaac (Edmun McMillen y Florian Himsl (2011)) o Hades (Super Giant Games (2018)) hasta la creación de mundos completos como Minecraft (Mojang (2009)). Y algunos de los proyectos más complejos, como Dwarf Fortress (Tarn y Zach Adams (2006)).

Sin embargo, la generación de historias o personajes es algo que no se ha explorado en tanta profundidad, y ahora que la tecnología de la inteligencia artificial está en auge, parece el momento idóneo para indagar en las posibilidades que estas nuevas herramientas nos ofrecen.

Otro de los motores del proyecto es la profunidad y calidad narrativa que es capaz de alcanzar un LLM como ChatGPT, se pretende analizar la complejidad de sus historias y la profundidad que alcanzan sus personajes.

Con la premisa de la importancia que la generación aleatoria y procedural han tenido en el mundo del videojuego, además del protagonismo de la inteligencia artifi-

cial en el mundo de hoy en día, parece apropiado elaborar un proyecto cuyo propósito sea crear una herramienta de generación que trate de hacer más sencilla la labor de diseñadores y desarrolladores de videojuegos.

1.2. Hipótesis

La hipótesis del proyecto es que el sistema implementado permitirá ahorrar tiempo y esfuerzo tanto a desarrolladores como a diseñadores de videojuegos, evitándoles trabajo tedioso y que puede ayudar a equipos pequeños y sin presupuesto a buscar inspiración narrativa o a generar un entorno que puede serles útil en videojuegos de exploración espacial.

1.3. Objetivos

El objetivo del proyecto es crear una herramienta que reduzca la carga de trabajo tanto de desarrolladores como de diseñadores de videojuegos, generando un sistema solar con sus propias físicas funcionales, planetas con recursos, sistemas de gobierno, facciones, razas, y perfiles de personajes.

La inteligencia artificial se encarga de leer la descripción que el usuario aporta del sistema solar concreto que quiere, además se le añade un promt(Qué se muestra más adelante) explicándole como debe generar un archivo Json, que será lo que nos devuelva y que utilicemos para parsear la información y generar la simulación física. Es en este punto cuando el LLM genera tanto la historia del sistema solar como los personajes que lo habitan.

A continuación nombraremos y desarrollaremos en profundidad los objetivos del proyecto.

- **Simulación física del modelo:** utilizamos Unity (*Unity Engine*) para los cálculos físicos. Se utilizará la ecuación universal de la gravedad para atribuirles a los objetos su velocidad de rotación en torno a la estrella. Se busca una simulación que replique de forma realista la gravitación de los cuerpos. Sin embargo, a su vez buscamos un modelo algo más simplificado para facilitar el resultado que se le proporcione al usuario. Entre estas simplificaciones está pensado limitar la rotación de los planetas a un solo eje de la estrella. También se dividirán los planetas en cuatro tipos, con la intención de simplificar su generación. Semejante a la rotación de los planetas respecto a su estrella madre, queremos una rotación de los satélites en torno a sus planetas. Con estas simplificaciones, esperamos conseguir un modelo que se asemeje al diagrama de la figura 4.2
- **Comunicación con ChatGPT:** en este apartado buscamos tanto poder comunicar nuestro programa con el LLM como asegurarnos que sus datos son válidos a nivel estructural. Nos encargaremos de comprobar cual ha sido la respuesta, analizar a nivel estructural para asegurar que el programa no sufra errores graves o fatales durante su ejecución, y comunicarle al LLM cual ha sido el problema y que hay que hacer para solucionarlo.

- **Generación de una historia coherente e interesante:** dónde entendemos por coherente que no haya contradicciones narrativas en la generación, como por ejemplo que si un planeta está en guerra con otro, este otro no esté en paz con el anterior o que un personaje no pueda ser su propio enemigo. Y entedemos por interesante, por ejemplo, que el sistema solar no esté completamente en paz y que los personajes tengan conflictos entre sí.

1.4. Metodología de trabajo

Para llevar a cabo el proyecto, se utilizará el motor de videojuegos gratuito Unity[1], en concreto en su versión 2021.3.17. Como entorno de desarrollo utilizaremos visual studio 2022. Como herramienta de control de versiones utilizaremos git hub a través de github desktop.

Para facilitar la conversión de la entrada a una estructura de datos se utilizará la librería Newtonsoft Json [2], para ser exactos en su versión 13.0.3. Esta librería se encarga de convertir la entrada en formato Json a una estructura de datos utilizable.

Para esto se programará un módulo que se encargue de enviar la petición del usuario junto con el prompt al LLM, otro que se asegurará de que el archivo de entrada recibido desde el LLM es correcto y no provocará un error fatal en el programa. También habrá un módulo encargado de comprobar narrativamente el LLM, explicaremos lo que significa esa comprobación narrativa más adelante.

Habrá un módulo encargado de la generación de los datos sobre los cuerpos del sistema solar, otro encargado de generarlos y colocarlos en su lugar y uno más que mantendrá la simulación física, calculando las fuerzas que debe tener cada cuerpo del sistema y que hará las funciones de game manager.

1.5. Plan de trabajo

A continuación vamos a explicar los puntos más importantes del desarrollo de la aplicación, además de mostrar una planificación a priori dividida por meses con los tiempos aproximados que calculamos que llevará implementar distitnos apartados.

- **Diseño de la aplicación.** El proyecto apunta principalmente a desarrollar una herramienta útil para desarrolladores y diseñadores de videojuegos, por lo que el primer paso debía ser tomar una decisión acerca de que ibamos a generar. En un principio se barajó la posibilidad de generar un único planeta con sus correspondientes accidentes geógraficos, pero se descartó debido a la importante cantidad de trabajos parecidos, por no mencionar su existencia en juegos ya publicados como Don´t Starve (Klei Enterteiment (2013)) por lo que se tomó la decisión de apuntar a la generación de planetas en vez de un solo mundo.

  Dentro de este contexto se pensó en la posibilidad de permitir más de una estrella por sistema solar, pero este hecho dificultaba de manera importante

[1] https://unity.com/releases/editor/whats-new/2021.3.17
[2] https://www.nuget.org/packages/Newtonsoft.Json/

la simulación física, ya que si apuntasemos solo a simular lo que ocurriría en esas condiciones, no habría problema, pero como pretendemos proporcionar al usuario una herramienta que le permita crear un entorno para un videojuego, además de una historia coherente que lo acompañe, hemos decidido reducir el número a un único astro, de tal manera que aseguramos prácticamente en todos los casos que los planetas solicitados por el usuario se mostrarán de forma correcta y tendrán coherencia con la historia contada.

Decidimos utilizar Unity como entorno tanto por su facilidad para la simulación de la física, como por su importancia en el mundo del videojuego actual.

- **Simulación y generación física del entorno:** En el ámbito narrativo, es poco atractivo que si deseas, por ejemplo, cinco planetas no los encuentres a la hora de correr el programa. Por tanto es importante asegurar que los cuerpos se generen en puntos suficientemente cercanos a la estrella como para ser alcanzables por el jugador, pero lo bastante lejanos para darle coherencia a la narrativa y evitar que los planetas choquen entre sí.

- **Comunicación con ChatGPT** A día de hoy existen una infinidad de inteligencias artificiales parecidas a ChatGPT. Sin embargo, pocas o ninguna son tan potentes como la susodicha, por lo que parece razonable escoger esta para medir las limitaciones que pueden tener estos modelos y para elaborar una herramienta lo más potente posible.

  Para el proyecto es esencial que el usuario pueda comunicarse directamente con el LLM, de manera que necesitaremos modificiar el editor de Unity para permitirle al usuario comunicarse de tú a tú con la I.A.

- **Análisis lógico de los resultados de la I.A** A pesar de la potencia que ha demostrado ChatGPT, es necesario asegurarnos de que la información devuelta sea válida para nuestra estructura de datos.

  Una vez hayamos analizado la información, bien se enviará al sistema de generación físico, o bien le pediremos una correción al LLM. Para ello, necesitamos identificar cuál ha sido el problema, por lo que tendremos que leer los datos al completo, y una vez hecho mandar un mensaje acorde al problema a la inteligencia artificial.

- **Análisis narrativo del resultado** Sería muy fácil generar una historia sin sustancia y sin intriga, de manera que debemos hacer un análisis narrativo de lo que la inteligencia artificial nos proporciona.

  La narrativa es algo subjetivo, aunque hay algunos aspectos que podemos contabilizar y analiar mediante código, estos son, entre otros, el número de personajes, las relaciones que tienen entre ellos, el estado político de la sociedad o el número de distintas facciones dentro del sistema.

  Sin embargo, lo que cada usuario puede considerar una buena narrativa es algo imposible de medir o asegurar, por lo que le permitiremos modificar los parámetros anteriormente mencionados para que se adapte a sus preferencias o necesidades.

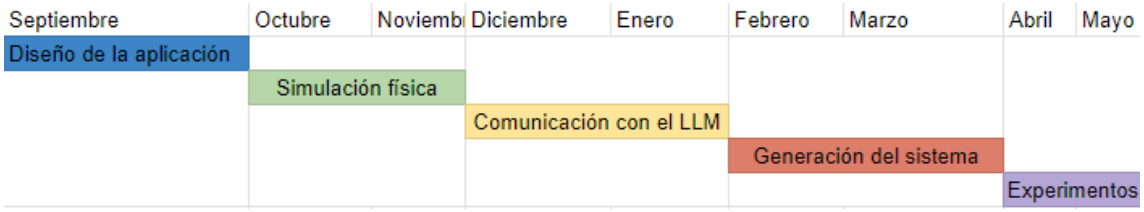

Figura 1.1: Diagrama de gant en el que se muestra la planificación incial del proyecto dividida por meses

Una vez obtengaos los datos, se empleará el mismo proceso que con el análisis lógico, de tal manera que si no se cumplen los requisistos pedidos a la inteligencia artificial, se detectará el error y se le devolverá un mensaje pidiendo que haga las correciones propicias, se repetirá este proceso hasta que obtengamos el resultado deseado.

En la figura1.1 podemos ver la planificación inicial pensada para el proyecto.

Capítulo 2

Estado de la Cuestión

En este capítulo vamos a explorar el estado actual de la generación de contenido para videojuegos. Del estado de la narrativa y sus funciones, además de mencionar otros estudios realizados sobre el ámbito.

Es difícil de precisar, aunque muchos denominan Adventure (Warren Robinett y Atari (1979)) como el primer juego narrativo. Y desde ese momento ha habido inumerables instancias de videojuegos cuyo objetivo ha sido contar una hisoria, por nombrar algunos: Zelda ocarina of time (Nintendo (1998)), Monkey island (Lucasfilms Games (1990)) o Fallout 3 (Bethesda (2008)). La narrativa ha tomado una parte tan importante en los video juegos que es casi imposible encontrar uno que no contenga algo de historia desde hace muchos años. Incluso juegos competitivos como Valorant (Riot Games (2020)) tienen ahora su parte narrativa aunque su jugabilidad no tenga nada que ver con ella. En este tipo de productos es espeiclamente importante la narrativa corporal o visual, cosas que se explican en el libro *Videogame Narrative and Criticism Playing the Story* (Thabet (2015)).

La historia se ha convertido en una parte prácticamente esencial de los videojuegos, casi tan necesario como los efectos de sonido o el apartado gráfico. Sin embargo, en contraste con estos dos ámbitos, la narrativa es el único que de no estar, o no ser demasiado claro, es elaborado por los usuarios, hasta tal punto que hay comunidades de usuarios dedicadas por entero a teorizar sobre los trasfondos de los personajes, los eventos pasados del juego o las posibbles causas de los eventos que vemos en el juego. Este es el caso de Elden Ring (FromSoftware (2022)) o incluso Clash Royale (Supercell (2016)).

Pero, ¿qué hace tan interesante la historia de un videojuego? Pues bien, como se menciona en el libro *Storyplaying: Agency and Narrative in Video Games* (Domsch (2013)), la historia de un videojuego es nodal, término que implica que para un punto cualquiera de la historia permite más de una continuación. Y aunque esto no es cierto en todos los videojuegos, ya que algunas historias, los eventos que sufren los personajes o los acontecimientos globales del videojuego están predefinidos y no son evitables, si es cierto que el jugador puede llegar a ellos de muchas maneras distintas, lo que convierte la historia en única para cada usuario.

Por supuesto, los videojuegos también tienen sus problemas con respecto a la narrativa, siendo el más conocido de ellos, la disonancia ludonarrativa. Siguiendo la

definición que se da en el artículo *Ludonarrative Dissonance: Is Storytelling About Reaching Harmony?* (Seraphine (2016)) la disonancia ludonarrativa es la sensación de desapego con el mundo que nos presenta el videojuego, normalmente porque las acciones que realiza nuestro personaje no terminan de encajar con su perfil dentro de la historia o con el mundo en el que se encuentra. Pongamos como ejemplo el personaje de Nathan Drake de la saga Uncharted (Naughty Dog (2007)), Nathan Drake es un cazador de tesoros que se nos presenta como un hombre entrañable aunque algo avaricioso, obrando de manera correcta e intentando que nadie salga herido durante la mayor parte de la historia. Sin embargo, en las secciones de la historia en las que el jugador toma el control de Nathan, este se dedica a disparar y asesinar a incontables personas, y aunque se traten de enemigos que en su mayor parte son piratas y mercenarios, Nathan no presenta ninguna reacción humana al hecho de matar a un ser humano.

Este efecto se produce al querer mostrar al jugador un personaje justo y entrañable al cual pueda coger cariño, pero al ser un juego de disparos, acción y aventura, el producto requiere de estas secciones para darle la sensación de peligro y reto al jugador. Otro caso lo encontramos en el videojuego Bioshock (2K Games (2007)) como se explica en el artículo *ludonarrative dissonance in bioshock the problem of what the game is about* (Hocking (2009)).

Tal es la importancia de estos apartados que existen estudios que intentan teorizar por completo la narrativa de los videojuegos como puede ser *Theorising Video Game Narrative* (Majewski et al. (2003)). También encontramos estudios que recogen las estructuras narrativas emergentes que han nacido o resurgido con los videojuegos, como podemos ver en el artículo *Narrative Structures in Computer and Video Games: Part 1: Context, Definitions, and Initial Findings* (Ip (2011))

En las siguientes secciones se entrará más en detalle con respecto a los diferentes apartados relacionados con la cuestión.

Tal como se define en el libro Machine Learning (Alpaydin (2021)) el aprendizaje automático es una rama de la inteligencia artificial que se centra en la construcción y estudio de sistemas que pueden aprender a partir de datos. La idea del aprendizaje automático, o *machine learning* en inglés, parte de la premisa de que, por ejemplo, es más fácil explicarle a un niño la diferencia entre un coche y un coche de carreras mostrándole ejemplos de ambas cosas en vez de intentando explicarle cuales son las características de ambos. Así mismo, el aprendizaje automático apunta a que en vez de codificar el conocimiento en los ordenadores les permitamos aprender relaciones y patrones a través de ejemplos y observaciones (Janiesch et al. (2021)).

De esta manera, podemos enseñar a una máquina a reconocer lo que es, por ejemplo, un perro, mostrándole miles o millones de fotografías de lo que es un perro. Por supuesto, cuanto mayor sea la muestra que le damos a la máquina para que aprenda, mayor porcentaje de éxito tendrán los resultados.

Podemos clasificar el aprendizaje automático en tres tipos.

- Aprendizaje supervisado: El aprendizaje supervisado se caracteriza porque se le dan a la máquina los resultados que se esperan para cada entrada. Las técnicas más comunes de aprendizaje supervisado son, entre otros; la regresión lineal, la regresión logística o las máquinas de vectores de soporte o *support*

vector machines en inglés, como se describe en el artículo *Supervised Learning* (Cunningham et al. (2008)).

- Aprendizaje no supervisado: En el aprendizaje no supervisado el algoritmo intenta identificar relaciones naturales dentro de un grupo de datos sin ninguna referencia de cuales son las respuestas correctas. Estos modelos tratan de identificar subgrupos que compartan características. Algunos emplos de estos modelos son los k-means clusterings y los expectation-maximization clusterings utilizando modelos gaussianos mixtos, como se explica en el artículo *Unsupervised Learning* (Dayan et al. (1999)).

- Aprendizaje semisupervisado: Como su nombre indica, es una mezcla de los dos modelos anteriores, de tal manera que se le da al modelo algunos resultados para que sea capaz de encontrar los resultados correctos con el resto de casos, como se explica en el artículo *Semi-Supervised Learning* (Learning (2006))

Estas definiciones se han obtenido del artículo *What is Machine Learning? A Primer for the Epidemiologist* (Bi et al. (2019)).

Como se ha mencionado, el machine learning se inspira en las maneras de aprendizaje de los seres humanos. Es por esto que es entendible que algunos modelos estén basados incluso en la estructura cerebral del cerebro humano para codificar la información.

2.1. Aprendizaje automático

Entendemos aprendizaje automático como los algoritmos que a partir de grandes cantidades de datos son capaces de aprender una o varias tareas en concreto. Como se menciona en el artículo *Machine Learning from Theory to Algorithms: An Overview* (Alzubi et al. (2018)) aprender consiste en adquirir nuevos conocimientos, o modificar los existentes, y es precisamente en eso en lo que se basan este tipo de algoritmos.

El origen del aprendizaje automático y las redes neuronales artificiales nace de la ambición humana de construir una máquina que simule el comportamiento del cerebro humano. Contruir una máquina así requiere entender como funciona nuestro sistema cognitivo, problema que se remonta hasta el siglo tercero antes de Cristo con Aristóteles, como se indica en el artículo *On the Origin of Deep Learning* (Wang y Raj (2017)).

A raíz de esto han nacido diverso modelos que pretenden simular el funcionamiento de la mente humana, objetivo que parece cada día más cercano y realista. Antes se mencionó que este tipo de algoritmos se basan en la estrcutura del cerebro humano, pero también se basan en su funcionamiento, como se explica en el libro Aprendizaje automático (Moreno et al. (1994)), este tipo de algoritmos pretenden imitar la manera en que los animales aprenden, es decir, almacenando información para utilizarla en situaciones parecidas cuando sea necesario.

Otro formato de este tipo de algoritmo es *Lifelong machine* learning (Chen y Liu (2018)) el cual es similar al aprendizaje automático, pero este modelo acumula la

información que recibe, generando grandes cantidades de memoria de las que extraer datos.

2.2. Deep learning

El deep learning, como lo conocemos a día de hoy, es una rama del machine learning, o aprendizaje automático, cuyo funcionamiento se inspira en la estructura del cerebro humano, utilizando redes neuronales creadas de manera artificial. Estas redes constituyen diversas capas de unidades de procesamiento, que aprenden a reconocer patrones complejos en conjuntos de datos medianta la extracción de características en diferentes capas de abstacción.

El concepto de redes neuronales artificiales no es contemporáneo a nosotros, pues en 1957, Frank Rosenblatt presentaba uno de los primeros modelos de red neuronal basada en algoritmos matemáticos, llamada "Perceptrón", nombre que se le atribuye ahora a cada uno de las neuronas de la red neuronal.

Sin embargo, el concepto de deep learning es más moderno, datando sus comienzos a principios del 2000. Aunque no se popularizó hasta finales de la misma década, debido sobre todo a limitaciones computacionales y de datos. Después, con la aparición de computadoras más potentes, nuevas técnicas de entrenamiento y la disponibilidad de grandes conjuntos de datos, el deep learning dio el salto a lo que conocemos hoy en día.

Pero, ¿qué hace esta metodología tan distina al resto? Pues bien, los algoritmos que funcionan mediante deep learning, se distinguen del resto ya que aprenden mediante ejemplos, es decir, en vez de seguir una serie de normas predefinidas, podemos enseñar a nuestro programa a cosas como: reconocer imágenes, la compresión del lenguaje natural, o la creación de historias (Deng et al. (2014)).

2.2.1. Componentes básicos de una red neuronal

El deep learning se basa principalmente en Redes Neuronales Artificiales(ANNS), estas redes son el bloque de construcción fundamental del deep learning, consisten en capas de nodos interconectados, a los cuales llamamos neuronas. Cada neurona está conectada con otras dos, de manera y cada una de esas conexiones tiene un peso. Entendiendo por peso un número que determina la dirección que tomará la información en la red neuronal.

Otra parte fundamental de este tipo de algoritmos, son las capas ocultas, llamadas así porque no están conectadas ni a la entrada ni a la salida de la red neuronal. Estas capas son las responsable de realizar las transformaciones complejas en los datos de entrada a medida que estos se propagan a través de la red. Cada capa oculta aprende y extrae características relevantes y útiles de los datos de entrada, permitiendo que la red neuroanal capture patrones cada vez más abstractos a medida que acanza desde la capa de entrada.

El último componente del deep learning son las funciones de activación, estas funciones toman como parámetro la suma ponderada de las entradas de una neurona, y con esta suma deciden si la neurona debería activarse o no y en que medida.

Entendemos que una neuora está activada cuando su salida contribuye a la salida total del programa, y que está desactivada cuando el caso es inverso, al ser su salida ínfima o nula. Encontramos cuatro tipos distintos de funciones de activación: La función sigmoide, que transforma los valores de entrada en un rango entre 0 y 1. La función hiperbólica, similar a la función sigmoide, pero con un rango entre -1 y 1. La función ReLU, que devuelve cero para valores negativos y el mismo valor de la entrada para valores positivos. Y la función Softmax, utilizada en la capa de salida para problemas de clasificación multiclase (Zhu et al. (2022)).

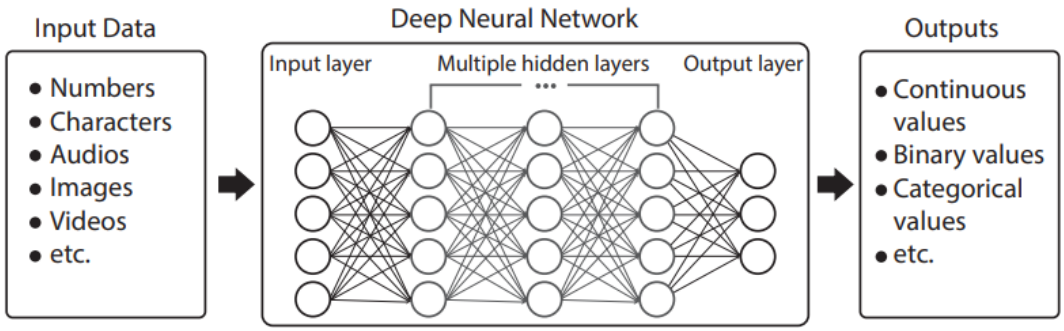

Figura 2.1: Diagrama de la estructura de una red neuronal extraido del artículo Deep Learning in Diverse Intelligent Sensor Based Systems Zhu et al. (2022))

2.2.2. Proceso de entrenamiento

A continuación, exploraremos como se entrena una red neuronal de tal manera que pueda aprender a partir de los datos. Primero, se inicializan los pesos de la red aleatoriamente. Después se hace Propagación hacia Adelante, o *Forward Propagation*, que como su propio nombre indica, consiste en propagar los datos desde la entrada, pasando por cada neurona y por las funciones de activación, hasta la salida final. Una vez tenemos la salida, se hace el cálculo del error, es decir, comparamos la salida de la red neuronal con la salida deseada. Acto seguido, pasamos a Retropropagación del error, o *Backpropagation*, este paso consiste en propagar el error hacia atrás en la red neuronal, calculando las derivadas parciales de la función de pérdida con respecto a los pesos de la red. Estas derivadas indican como cambiar los pesos para reducir la pérdida. Como es lógico, después se actualizan los pesos con los nuevos que hemos calculado. Y por último repetimos estos pasos hasta obtener resultados.

2.3. Transformers

Término expuesto por primera vez en *Attetion is all you need* (Vaswani et al. (2017)), un transformer es una arquitectura de modelo de aprendizaje automático que destaca por su capacidad para capturar relaciones a largo plazo en secuencias de datos de entrada, sin depender de estructuras recurrentes. Vamos a explorar sus componenentes.

- Primero, el mecanismo de atención, que es el proceso mediante el cuál el modelo determina cuales son las partes más relevantes de la entrada, por ejmeplo, las palabras clave en un texto.

- Las capas de codificación son la parte del Transformer que procesa la entrada y la transforma en una representación siginificativa. Cada capa de codificación consta de dos subcapas principales: Auto-atención, encargada de calcular la importancia de cada palabra en relación con todas las demás palabras en la secuencia de entrada. Permite al Transformer atender diferentes partes de la entrada en fucnión de su relevancia para el contexto general. La otra subcapa es la red neuronal de alimentación hacia adelante, cuyo objetivo es ponderar la importancia de cada palabra, esta subcapa procesa cada palabra indiviudalmente a través de una red neuronal.

- También encontramos las capas de decodificación, responsables de generar la salida secuencial basada en la represantición de la entrada. Cada capa de decodificación consta de tres subcapas principales: Auto-atención, similar a la de codificaicón, pero con una modificación, se impone una restricción adicional para evitar que las posiciones futuras en la secuencia se utilicen durante la atención. Auto-atención cruzada, esta subcapa permite que el descodificador atienda la representación de la entrada codificiada, ayuda al modelo a capturar la relación entre la entrada y la salida durante la generación del resultado. La última subcapa es una red neuronal similar a la de la capa de codificación

Entendemos atención como el mapeo de consultas y la asignación de parejas llave-valor al resultado, dónde las consultas, las llaves, los valores y el resultado son vectores. El resultado es la suma ponderada de los valores, donde el peso asignado a cada valor es procesado por un función compatibe de las consultas con la llave correspondiente.

En estos modelos, encontramos dos tipos de atención;

- Atención con Producto Puntual Escalado(Scaled Dot-Product Attention): La entrada consiste en consultas y llaves de dimension d_k, y valores de dimensión d_y. Se procesa el producto escalar the las consultas con todas las llaves, se divide cada una por $\sqrt{\$d_k\$}$, y se aplica una función softmax para obtener los pesos de los valores.

- Multi-Head Attention: En vez de utilizar una sola función de atención con d_model llaves dimensionales, valores y consultas, resulta beneficioso proyectar linealmente las consultas, llaves y valores h veces con proyecciones lineales diferentes de d_k, d_k y d_v dimensiones respectivamente. La multi-head attention permite al modelo prestar atención a información a distitnas posiciones (Cordonnier et al. (2020)).

Estos modelos han dado lugar a lo que conocemos como Large Language Models, lo cuales se basan en esta estructura para conseguir entender el lenguaje humano y simularlo (Irie et al. (2019)). Estos modelos dependen de memorizar la información con la que se entrenan, como se explica en el artículo *Memorizing Transformers* (Wu et al. (2022)).

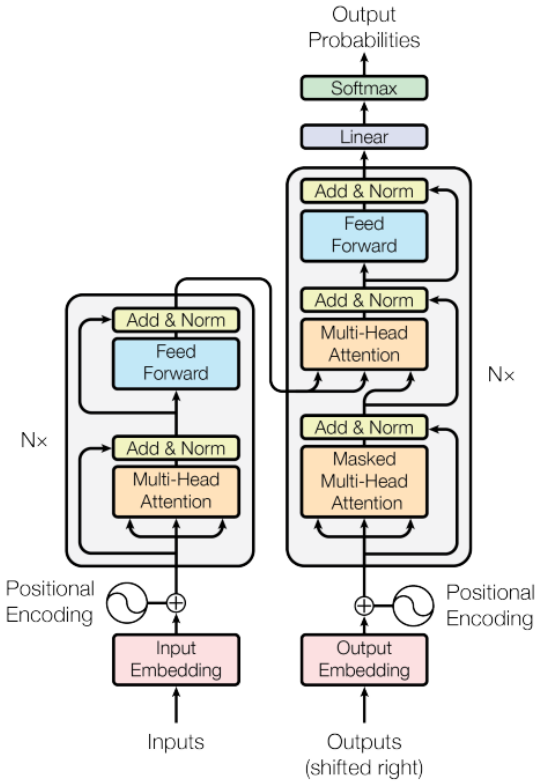

Figura 2.2: Estrcutura de un transformer, figura extraida del artículo *Attention is all you need* (Vaswani et al. (2017)).

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Figura 2.3: Ecuación Scaled Dot-Product Attetion

2.4. Large Language Models (LLM)

Uno de los logros más notables de la inteligencia artificial en los años recientes son los Modelos de Lenguaje Grandes(LLM, por sus siglas en inglés), o en otra palabras, enormes redes neuronales entrenadas específicamente para la predicción de palabras. Estos modelos destacan particularmente por su eficacia a la hora de generar texto fluido y coherente, a niveles que en ciertas ocasiones es indiferenciable del humano.(Pavlick (2023))

Este tipo de tecnología ya ha sido implementada en varios ámbitos como puede ser el sistema de respuestas inteligente de Gmail. Las empresas y los escritores individuales no han tardado en ver el potencial que este tipo de tecnología puede tener a la hora de cooperar con la mente humana en la generación de texto, como se explica en el artículo *Wordcraft: Story Writing With Large Language Models* (Yuan et al. (2022)).

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Figura 2.4: Ecuación de Multi-Head Attetion

No es para menos, pues este tipo de algoritmos está reduciendo el tiempo necesario para escribir artículos científicos, lo que por ende acelera el avance tecnológico en general, como podemos leer en el artículo *Science in the age of large language models* (Birhane et al. (2023)).

Liderando este tipo de modelos, ahora mismo se encuentra ChatGPT, lo que lo hace ideal para nuestro proyecto. Generative pre-trainer transformer, o GPT, por sus siglas en inglés, está basado en la arquitectura de los transformer, anteriormente descrita.

Por potente que sea esta tecnología, a día de hoy todavía tiene una serie de limitaciones que capan el alcance del proyecto. Sin ir más lejos, y como su propio nombre indica, Large, o grande en castellano, no es ninguna exageración. Estos modelos contienen millones de parametros, lo que aumenta mucho el espacio requerido para almacenarlos, por no hablar de las necesidades de hardware para su entrenamiento. Lo cual limita las posibilidades de uso y de entrenamiento a muchas compañias, o, como es nuestro caso, a personas individuales (Zangrossi et al. (2024)).

Aunque hay otras formas de implementar LLMs como puede ser sistemas basados en reglas o árboles de decisión (Egli (2023)), han sido los basados en transformers y redes neuronales los que han demostrado la mayor eficacia.

Por supuesto, la eficacia de estos modelos es inmensamente dependiente del input que el usuario proporcione, pudiendo obtener resultados totalmente distintos, en función de cuanto contexto se le proporcione al LLM. Se muestra un ejemplo en las figuras 2.5 y 2.6.

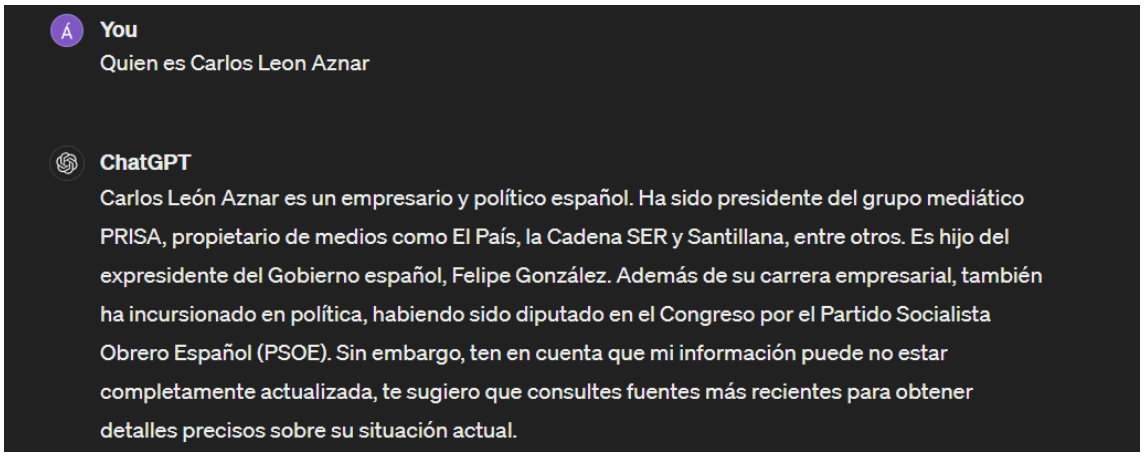

Figura 2.5: Ejemplo de input tomado desde ChatGPT, en este prompt se le da menos contexto al modelo

Por este motivo, en los últimos años ha surgido una disciplina que cada vez esta

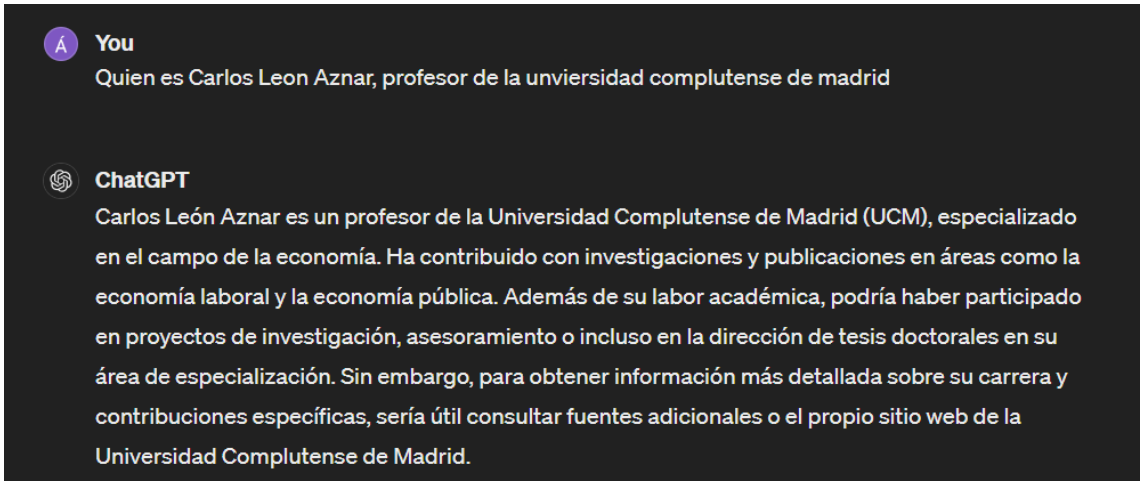

Figura 2.6: Ejemplo de input tomado desde ChatGPT, en este prompt se le da más contexto al LLM.

adquiriendo más y más importancia, de tal forma que incluso está empezando a ser una necesidad para los profesionales en el sector médico (Meskó (2023)).

2.5. Prompt engineering

El prompt engineering es el proceso mediante el cual se estructura un texto de tal forma que pueda ser comprendido por un modelo de inteligencia artificial generativa. Esta disciplina surge de la necesidad de dar contexto a la inteligencia artificial sobre el tema que se está tratando, y se ha convertido en una de las más importantes en lo referente a LLMs. Esta disciplina requiere de la capacidad de hipotetizar de cual pueden ser los problemas que sufre un prompt cuando el resultado no es el esperado, y la capacidad analítica como para identificar las hipótesis como correctar o erróneas y como solucionarlas, como se dice en el artículo *Prompt engineering a prompt engineer* (Ye et al. (2023)).

Y aunque hay otra serie de condiciones que pueden hacer variar la respuesta del modelo, como son; el tipo de arquitectura, los parámetros del modelo, los datos de entrenamiento o las técnicas de afinamiento, se ha demostrado que el prompt influye enormemente en el resultado del LLM.

Tomando como ejemplo el estudio realizado para el artículo *Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs* (Wang et al. (2024)) estudiado con pacientes que sufren de osteoarthritis, una enfermedad extensamente sufrida por la población, lo cual proporciona un número elevado de sujetos. Se ha investigado cuál era la credibilidad de un número determinado de LLMs, entiendo credibilidad como el número de veces que el diagnóstico del LLM coincide con el diágnostico correcto.

En este experimento, se ha utilizado cuatro tipos distintos de prompt, los cuales han conseguido una consistencia con el diágnostico real de entre el 50.6 % y el 63 % con ROT prompting sobre GPT-4, el que ha sido el resultado más alto. (Wang et al. (2024)) Viendo estos resultados, parece lógico utilizar la versión de ChatGPT más

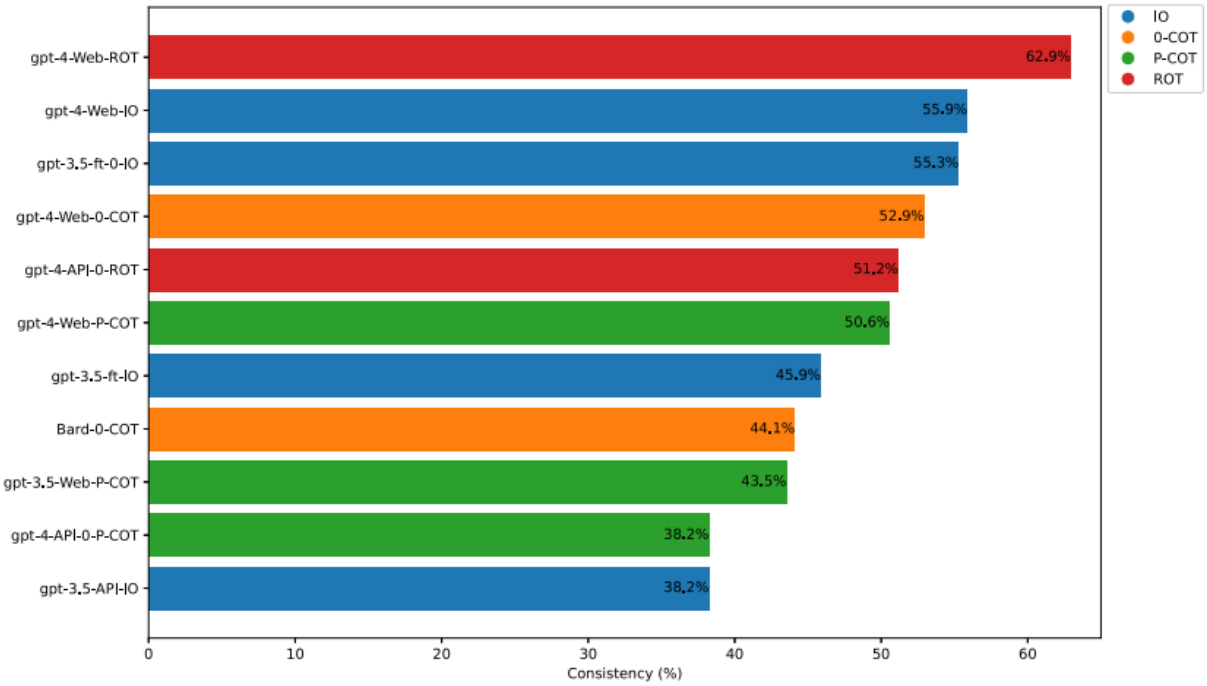

Figura 2.7: Resultados extraidos del estudio Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs (Wang et al. (2024))

avanzada a nuestro alcance para el proyecto. Además, ya hay estudios específicos enfocados en mejorar la relación de los prompts con ChatGPT, como por ejemplo *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT* (White et al. (2023)) que describe un catálogo de técnicas utilizadas para solventar determinados problemas que se dan al conversar con LLMs, o *Prompt Engineering with ChatGPT: A Guide for Academic Writers* (Giray (2023)) en el que se explica la necesidad de los escritores académicos de aprender esta disciplina.

2.6. GPT y ChatGPT

GPT, por sus siglas en inglés, Generative pretrained transformer, hace referncia a una familia de modelos neuronales que utilizan la arquitectura de los transformadores. ChatGPT por su parte, hace uso de esta base para implementar un bot capaz de conversar con el usuario.

Entre los numerosos modelos similares a GPT, fue este el propuesto por OpenAi en mayo de 2020 fue seleccionado entre el top diez de tecnologías revolucionarias por el MIT technology Review en 2021, atribuido por su extensa escala de parámetros, su habilidad de modelado excepcional, el rendimineto generalizado multitarea y la habilidad de aprendizaje con pocas muestras (Zhang y Li (2021)).

Esto es un claro indicativo del potencial de este modelo, sumado a su rápida evolución, GPT ofrece una cnatidad inumerable de oportunidades, y como se menciona en el artículo *Generative AI (gAI) in medical education: Chat-GPT and co* (Moritz et al. (2023)) el automovil necesitó un siglo para convertirse en la máquina que te-

nemos hoy en día, GPT-3.5 solo necesitó el año 2023 para convertirse en GPT 4, capaz de recibir imágenes como input.

Pero, ¿está GPT solo en este panorama? GPT está en lo que conocemos como la tercera generación de modelos de lenguaje pre entrenados, y como es de suponer, antes de esta tercera generación, esta tecnología tuvo predecesores. Empenzando por modelos como N-Gram, que utilizan un modelo estadístico discreto tradicional basado en frecuencia para calcular la probabilidad de ocurrencia de la siguiente palabra dado el contexto anterior (Zhang y Li (2021)).

Este tipo de modelo presenta tres limitaciones principales; el método de representación discreto por palabras tiene una habilidad de descripción muy pobre, el espacio de parámetros aumenta exponencialmente y los modelos basado en probabilidad estadística tienen poca capacidad de modelado, dando como resultado una capacidad de descripción pobre, poca robusted y poca precisión. Para solucionar estos problemas, los modelos de lenguaje pre entrenados han utilizado datos a gran escala, o incluso a nivel global. De esta manera, modelos como ELMo, BERT, y por supuesto GPT, han desarrollado, no solo el modelo clásico de probabiliad de los modelos antiguos, si no también una representación vectorial del segmento del lenguaje (Zhang y Li (2021)).

Con la introducción de ELMo, comienza la segunda generación de de modelos pre entrenados, ELMo es un modelo generador que utiliza LSTM (Long Short Term Memory) bidireccional como extractor de características y realiza modelado dinámico basado en el contexto, Comparado con la primera generación de modelos de lenguaje pre entrenados, ELMo es capaz de manejar mejor la polisemia y es experto en la generación natural de lenguaje. BERT es un modelo de lenguaje enmascarado que utiliza un codificador de transformador como extractor de características y es particularmente competente en analizar y entender el lenguaje natural. GPT es un modelo generativo que también utiliza un decodificador de transformador como extractor de características y muestra un rendimiento superior en tareas de generación de lenguaje natural (Zhang y Li (2021)).

Conociendo todo esto, se nos planeta la siguiente pregunta; ¿cuál es la diferencia entre GPT y ChatGPT? Mientras que GPT es la estrcutura basada en los transformers anteriormente descrita, ChatGPT es una implementación específica de dicha estrcutura, diseñada especificamente para tareas de conversación y diálogo. ChatGPT está optimizado para interactuar con humanos en una conversación natural.

Pero ChatGPT muestra potencial en ámbitos mucho más allá de la converasción, y no es para menos, puesto que como se menciona en el artículo *ChatGPT and its potential role in medicine* (Nazir et al. (2023)), se cree que ChatGPT ha aprendido todo lo que había en internet hasta Enero de 2022, por lo que puede dar respuestas precisas sobre prácticamente todo. Y, de nuevo, como se menciona en el artículo anterior, ChatGPT tiene el potencial de resolver dudas sencillas a pacientes con diferentes enfermedades, como por ejemplo cáncer, disminuyendo así el tiempo que los médicos y otros profesionales tuviesen que invertir en consultas menores, pudiendo enfocarlo ahora en consultas de mayor importancia.

Por supuesto, esta teconología también ha jugado un papel crucial en la capacidad de las personas de aprender por si mismas, como bien se explica en el artículo

How ChatGPT Can Transform Autodidactic Experiences and Open Education? (Firat (2023)).

Cabe mencionar que durante el desarrollo de la herramienta se hizo público ChatGPT 4, el cual hubiese sido mejor utilizar como indican artículos como *ChatGPT-3.5 and ChatGPT-4 dermatological knowledge level based on the Specialty Certificate Examination in Dermatology* (Lewandowski et al. (2023)) en el que se hace una comparación de ambas versiones con respecto a su conocimiento dermatológico. Sin embargo, como el desarrollo ya estaba más que avanzado, tuvimos que tomar la decisión de continuar con ChatGPT 3.5.

2.7. Aprendizaje por refuerzo

Otro método para entrenar o enseñar a estos modelos es el aprendizaje por refuerzo, una carácterística de los seres vivos que se ha introducido en los modelos de inteligencia artificial. Citando el artículo *Reinforcement learning and episodic memory in humans and animals: an integrative framework*, el aprendizaje por refuerzo es el proceso mediante el cual, por prueba y error, los organismos aprenden a predecir y obtener una recompensa.(Gershman y Daw (2017)) Por este motivo, y por su evidente eficacia a lo largo de la historia, resulta propicio intentar enseñar a una inteligencia mediante este proceso.

Como hemos dicho, el aprendizaje por refuerzo se basa en tomar las decisiones que llevan a una mejor recompensa, pero, ¿qué es una recompensa para una inteligencia artificial? Bien, pongamos el ejemplo de AlphaGo. AlphaGo es una inteligencia artificial implementada a través de deep learning en redes neuronales con el objetivo de jugar al juego de mesa Go. Go es un juego de mesa chino para dos jugadores cuyo objetivo es conseguir la mayor cantidad de tablero posilbe. AlphaGo funciona manteniendo una red de valores asignados a cada posición posible del tablero, esta red reduce la profundidad de la búsqueda truncando los árboles de decsión basándose en el valor dado, es decir, en la recompensa"numérica que el programa obtiene al evaluar esa posción. AlphaGo se entrena primero mediante aprendizaje supervisado, viendo jugar a jugadores profesionales de Go, y luego mediante aprendizaje por refuerzo, en el que juega contra sí mismo, y donde obtiene los valores de cada posición (Matsuo et al. (2022))

Según el artículo *Reinforcement learning: A survey* (Kaelbling et al. (1996)) hay dos estrategias principales para resolver problemas mediante aprendizaje por refuerzo. La primera consiste en buscar en el espacio de comportamiento yno que desempeñe de manera adecuada en el entorno propuesto. El segundo es utilizar técnicas estadísticas y métodos de programación dinámica para estimar la utilidad de determinadas acciones.

En el artículo *LLM-based Multi-Agent Reinforcement Learning: Current and Future Directions* (Sun et al. (2024)) se explica como este tipo de aprendizaje se aplica a los LLMs. Además, se citan ejemplos de experimentos realizados aplicando aprendizaje por refuerzo a LLMs, como puede ser el caso de el artículo *React: Synergizing reasoning and acting in language models* (Yao et al. (2022)), en el que se le pide al LLM que genere una solución para un problema dadas ciertas observaciones.

2.8. Videojuegos espaciales de exploración

Pocas cosas son más propias de la naturaleza huamana que la curiosisdad, un sentimiento que siempre ha llevado a las personas a intentar descubrir que se esconde más allá de sus horizontes. Y en nuestros tiempo, el siguiente horizontes que debemos traspasar es el espacio y sus secretos. Sabiendo esto, no es ninguna sorpresa que hayan nacido inumerables obras y teorías sobre lo que las estrellas nos deparan. Y como era de esperar, los videojuegos no se han quedado atrás, con incontables obras basadas en la exploración espacial.

Durante una gran parte de su historia, los videojuegos fueron historias o secuencia lineales, entendiendo por lineales que sus eventos estaban predefinidos de una manera concreta que el jugador no podía alterar. Esto cambia en 1980 con la aparición de Mystery House (On-Line Systems (1980)), un videojuego que permitía explorar una casa e ir resolviendo sus puzzles sin un orden específico.

Después de Mystery House, es prácticamente imposible nombrar todos y cada uno de los videojuegos de exploración, o como se les conoce a día de hoy, de mundo abierto. Algunos videojuegos de mundo abierto imitan espacios reales, lo cual puede llevar a disonancias con los entornos reales como se explica en el artículo *Geographical Aspects of Open-World Video Games* (Fraile-Jurado (2023)), algunos de estos títulos pueden ser Ghost of Tsunima (Sucker Punch Productions (2021)) o Far Cry 3 (Ubisoft (2012)), por nombrar algunos.

Sin embargo, los videojuegos ambientados en el espacio o en entornos imaginarios o fantásticos no tienen este tipo de limitaciones. A continuación echaremos un vistazo a algunos títulos de exploración espacial que pueden servirnos de base para inspirar las simulaciones y entornos de la herramienta.

- No man´s sky (Hello Games (2016)): Es un juego cuyos planetas se generan de manera procedural, permitiendo al usuario explorar una cantidad casi infinita de ellos. Es un videojuego en tercera persona y en tres dimensiones, cuyo objetivo es la recolección de recursos y la supervivencia.

- Starfield (Bethesda (2022)): Entrando más en temas de narrativa, Starfield es el perfecto ejemplo en cuanto a personajes e historia. Siguiendo las líneas de desarrollo de Bethesda, Starfield es una base perfecta desde la que partir a la hora de imaginar un videojuego de estas características.

- The outter wilds (Annapurna Interactive (2019)): The outter wilds representa un icono en el diseño, no solo de videojuegos cuya ambientación se encuentre en el espacio, si no de toda la industria. The outter wilds nos presenta un sistema solar cuyos secretos tendremos que desentrañar a través de la exploración.

- Astroneer (System Era Softworks (2016)): Astroneer es un videojuego que aleja de la narrativa para centrarse en la exploración y en la generación de mundos y del sistema solar. Un videojuego centrado en el entorno y la recolección de recursos, aunque en una escala más reducida que No man´s sky.

Con estos ejemplos en mente, tenemos una base en la que asentar nuestro proyecto, tomando como referencias tanto las capacidades narrativas de dichos videojuegos

como sus propiedades físicas y ambientales.

Capítulo 3

Arquitectura general del generador y Comunicación con el LLM

3.1. Descripción general del proyecto

El proyecto consiste en una herramienta de desarrollo que comunica ChatGPT con el editor de Unity, de tal manera que el usuario pueda obtener una respuesta del LLM sin necesidad de salir del editor. De esta manera el usuario podrá hacer correcciones al modelo a medida que vaya probando las diferentes versiones de la generación que le otorga el LLM. Así, el usuario puede ir refinando el resultado del LLM a medida que observa lo que el LLM ha generado.

El programa también hará ciertas comprobaciones antes de generar el sistema solar, comunicándose automáticamente con el LLM sin necesidad de la intervención del usuario, asegurándose así de no sufrir errores fatales de ejecución o errores narrativos que pudiesen empeorar el resultado obtenido por el usuario. En este capítulo especificaremos que consideramos errores narrativos además de todas las comprobaciones lógicas que hacemos para asegurarnos de que el archivo de entrada sigue la estructura esperada y que no producirá un fallo fatal en el programa.

Una vez hechas estas comprobaciones, comienza la generación física de la simulación, en la que a los planetas se les asignan tamaños, distancias con respecto a la estrella y masas. A su vez, se generan los satélites con las mismas carácterísticas que los planetas, es decir, distancia con respecto a su planeta, tamaños y masas para la simulación física. En dicha simulación, se le da una velocidad inicial a cada cuerpo, comenzando la rotación en torno a la estrella y en el caso de los satélites, también a su planeta. Después durante cada iteración se le da la fuerza correspondiente según a la ley de la gravitación universal de Newton a cada cuerpo, continuando con sus rotaciones.

Una vez generado el sistema solar, el usuario puede moverse como si estuviese dentro de una nave espacial, acercándose a los planetas para inspeccionar sus propiedades. El usario también dispone de una interfaz a través de la cual puede inspeccionar todos los apartados de los personajes y de los planetas.

En las secciones siguientes explicaremos todos estos apartados relacionados con la comunicación con el LLM y en el siguiente capítulo todo lo relacionado con la

generación del sistema y la simulación física.

3.2. Modulo de comunicación con el modelo de lenguaje

Para que el usuario pueda comunicarse con ChatGPT se ha implementado un script que modifica el editor de unity como aparece en la figura 3.1.

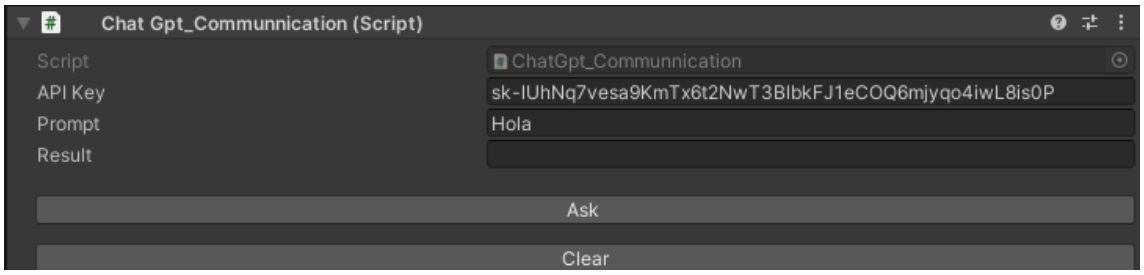

Figura 3.1: Captura de pantalla de la modificación del editor de Unity de tal manera que podamos conversar con el ChatGPT

De esta manera el usuario puede comunicarse con el LLM sin necesidad de salir del editor de Unity. En el apartado de API Key el usuario debe insertar una clave adecuada para la API de ChatGPT. En el apartado de prompt el usuario introduce su petición y en el apartado de result obtiene la respuesta que le da la inteligencia artificial. Aquí el usuario también podrá conversar con el modelo para corregir errores o hacer cambios de tal manera que podrá cambiar cualquier apartado del resultado sin salir del editor.

Es importante mencionar que este apartado no se ha podido desarrollar por completo debido a la falta de una clave adecuada para ChatGPT. Sin embargo, el código está implmentado, pero no se ha podido probar de la manera que hubiesemos querido.

Al mensaje que se le envía al LLM se le añade un prompt que contiene la estructura de la clase que queremos crear a partir de la respuesta del LLM, añadido esta a una explicación de las reglas que seguimos para parsear la información y con una serie de condiciones narrativas y estructurales que simplifican el parseo. A continuación se detallan dichas condiciones y se especifica el prompt completo que se le envía al modelo.

Es importante saber que todos los parámetros que más tarde se parsearán a tipos enumerados están representados mediante secuencias de números crecientes.

- Medidas de la estrella: se le explica al LLM que el número 0 significa una estrella enana, el número 1 tamaño normal y el número 2 tamaño gigante.

- Tipo de los planetas: Se le especifica al LLM que el número 0 significa que el planeta es de fuego, el número 1 significa planeta de agua, el 2 planeta gaseoso y el 3 planeta helado.

- Tipos de relaciones: se especifican las relaciones que los planetas pueden tener entre sí. Siendo las posibilidades las siguientes; el mismo planeta, es decir, él mismo(0), Paz(1), Tensión política(2), guerra(3).

- Tipo de gobierno en cada planeta: los tipos de gobierno que puede tener un planeta son los siguientes; monarquía(0), república(1), democracia(2), fascismo autoritario(3), comunismo autoritario(4), clanes guerreros(5) y planeta inhabitado(6).

- Propósito de los personajes: cada personaje tendrá un propósito, el cual podrá ser uno de los siguientes; asesinato(0), conseguir un objeto(1), rescatar a alguien(2), obtener información(3) y explorar una ubicación(4).

- Recursos del planeta: a cada planeta se le asigna uno, varios o ningún recurso. En caso de que el planeta no tenga nada, esta categoría aparecerá vacía. En caso contrario existen las siguientes posibilidades; comida(0), minerales(1), magia(2), armas(3) o esclavos(4). Esta información se representa como un array de números enteros.

- Nombres: se le especifica al LLM que los personajes deben tener un nombre y un diálogo. También se le indica que los planetas deben tener nombre.

- Planetas no habitados: Se especifíca que los planetas cuyo booleano Inhabite sea igual a falso deben tener el gobierno inhabitado. También se especifica que no pueden tener armas o esclavos como recursos. Además, se requeire que estos planetas no estén en guerra con nadie.

- Nombres de los planetas, razas o personajes: se le exige al LLM que invente nombres reales para los planetas, razas o personajes. Se le indica que no es válido llamarlos Planet1, Planet2, etc.

- Nomenclatura de los enemigos: se indica al LLM que los enemigos deben estar indicados mediante números, poniéndole por ejemplo que si el personaje número tres es enemigo con el personaje número cero, su array debe contener el número cero.

- Los personajes no pueden ser enemigos de sí mismos: se le indica a la inteligencia artificial que un personaje no puede ser enemigo de sí mismo.

- Clase para la serialización: se le indica al LLM cual será la clase en la que serializaremos el archivo Json que nos devuelva.

- Condiciones finales: por último se le pide que genere un archivo Json sin nada más de texto que siga la descripción del usuario.

Una vez enviado el prompt al LLM debemos asegurarnos de que todo está listo para su correcta utilización, para ello tomamos las medidas especificiadas en la siguiente sección. A continuación se presenta el prompt al completo.

```
Knowing that:

-Star_Size = 0 equals Dawrf star, Star_Size = 1 equals Normal star
and Star_Size = 2 equals giant star

-Type = 0 equals FirePlanet, Type = 1 equals Water planet,
Type = 2 equals Gas planet and Type = 3 equals frozen planet

-Relations = 0 means its the same planet, Relations = 1 means peace,
Relations = 2 means politic tension and Relations = 3 means war.
Relations array must be same size as N_Planets,
as every planet must have a relations with each other.

 -Goverment = 0 means Monarchy, Goverment = 1 means Republic,
 Goverment = 2 means Democracy, Goverment = 3 means Autoritarian_Fascist,
 Goverment = 4 means Autoritarian_Comunist,
 Goverment = 5 means War_Clans, Goverment = 6 means Inhabited

-Purpose = 0 means Assasination, Purpose = 1 means Collect an item,
Purpose = 2 means Rescue somebody, Purpose = 3 means Obtain_information,
Purpose = 4 means Exploration

-Resources = 0 means Food, Resources = 1 means Minerals,
Resources = 2 means Magic, Resources = 3 means Weapons,
Resources = 4 means slaves

-Charactes must have dialogue and name, planets must have names too

-Planets with Inhabited = false must have the Not_habited goverment,
wich means Goverment must equal 6, and they cant have Weapons,
nor Slaves as resources, they also cant be in war with anyone

-You cant name the planets: Planet1, Planet2... etc.
Same with characters and races, u must come up with actual names

-Enemies must come determined by a number,
meaning that if Character number 3 is enemy with character number 0,
its array must contain number 0, if it is enemy with number 3
it must contain number 3

 -Characters cant be enemies with themselves

 -This is the class im going to serialize the Json into

[System.Serializable]
```

```
public class SolarSystem
{
    public int Number_of_Planets { get; set; }
    public int Star_Size { get; set; }
    public Planet[] Planets { get; set; }
    public string History { get; set; }

    public string[] Factions { get; set; }

    public string[] Races { get; set; }
    public Character[] Characters { get; set; }
}
[System.Serializable]
public class Planet
{
    public string Name { get; set; }
    public PlanetCharacteristics Characteristics { get; set; }
    public int Satellites { get; set; }

    //Parses into enum
    public int Type { get; set; }
    public bool Giant { get; set; }

    public bool Inhabited { get; set; }

    //Determines relations with each planet, including itself(always 0)
    public int[] Relations { get; set; }

    //Parses into enum
    public int Goverment { get; set; }
}
[System.Serializable]
public class PlanetCharacteristics
{
    public string Surface { get; set; }
    public string Temperature { get; set; }

    //Parses into enum
    public int[] Resources { get; set; }

}
[System.Serializable]
public class Character
{
    public string Name { get; set; }
    public string Faction { get; set; }
```

```
    public string Dialogue { get; set; }

    //Parses into enum
    public int Purpose { get; set; }

    //Other characters
    public Character[] Enemies { get; set; }

    public string Race { get; set; }

    public int Planet { get; set; }

    public string Description { get; set; }

}
Give me a Json, and only the Json please, dont add anymore text,
that follows the following description,
And give it to me in a Json format that I can copy paste please
Generame un sistema solar tal que
```

Al que le sigue la descripción que da el usuario. De tal forma que se espera un archivo de entrada con la siguiente estructura.

```
  {
"Number_of_Planets": Integer,
"Star_Size": Integer between 0 and 2,
"Planets": [
  {
    "Name": "String",
    "Characteristics": {
      "Surface": "String",
      "Temperature": "String",
      "Resources": [Array of numbers between 0 and 4]
    },
    "Satellites": Integer,
    "Type": Integer between 0 and 3,
    "Giant": boolean,
    "Inhabited": boolean,
    "Relations": [array of numbers of size Number_of_Planets - 1,
    and numbers ranging between 0 and 3],
    "Goverment": Integer between 0 and 6
  },
```

```
Number_of_Planets with same structure...
  ],
  "History": "String",
  "Factions": ["Array of strings"],
  "Races": ["Array of strings"],
  "Characters": [
    {
      "Name": "String",
      "Faction": "String from the Factions array",
      "Dialogue": "String",
      "Purpose": Integer between 0 and 4,
      "Enemies": [Array of numbers of size Number of Characters - 1,
      with numbers ranging between 0 and Number of Characters - 1],
      "Race": "String from the array of races",
      "Planet": Integer representing in which planet we can
      find this character, ranging from 0 to Number_of_Planets - 1,
      "Description": "String"
    },
N characters with same strcuture
}
```

3.3. Método de validación del esquema de entrada

Una vez recibida la respuesta adecuada del LLM, se comprueba que todos los parámetros estén dentro de los rangos permitidos de tal manera que no se produzca un fallo de ejecución, siendo dichos rangos los siguientes.

- Valores de los recursos del planeta: se comprueba que cada recurso de cada planeta esté dentro del rango válido, es decir, que ninguno supere el número máximo que permite el enumerado. En la versión actual del proyecto dicho número es cuatro. Si se encuentra un error se devuelve el siguiente mensaje.

  ```
  In planet: planets[i].Name Resource: j is out of bounds and
  must be a lower number. Donde planets[i].Name es el nombre
  del planeta dónde se ha encontrado el error y j el número
  que ha dado el error en concreto.
  ```

  Este error indica que uno de los elementos del array de recursos del planeta está fuera del rango y hay que modificarlo.

- Valores de las relaciones entre los planetas: se comprueba que cada relación de cada planeta esté dentro del rango del enumerado, cuyo máximo en esta versión del proyecto es tres. Si se encuentra un error, se devuelve el siguiente mensaje.

  ```
  In planet: planets[i].Name Its relation with planet number:
  j is out of bounds and must be a lower number. Siendo planets[i].Name
  ```

```
el nombre del planeta donde se ha encontrado el error y j
el número de planeta que ha dado lugar al error.
```

Este error indica que uno de los elemntos del array de relaciones está fueran del rango y hay que modificarlo.

- Valores de los gobiernos de los planetas: se comprueba que el valor del gobierno de cada planeta está dentro de los rangos del enumerado. Siendo el número máximo en este caso el seis. Si se encuentra un error se devuelve el siguiente mensaje.

  ```
  In planet: planets[i].Name Its goverment its out of bounds
  and must be a lower number. Siendo planets[i].Name el nombre
  del planeta dónde se ha encontrado el error.
  ```

  Esto error indica que el valor del parametro de gobierno del planeta está fuera del rango y hay que modificarlo.

Después de estas comprobaciones se pasa a los valores dentro de la lista de personajes.

- Existencia de la facción de cada personaje: se comprueba que la facción que se le ha asignado al personaje existe en la lista de facciones del sistema solar. En caso de error se le devolverá el mensaje.

  ```
  In Character: characters[i].Name you added a faction that
  doesnt exist, set the characters faction to one of the factions
  array. Siendo characters[i].Name el nombre del personaje en
  concreto.
  ```

  Este mensaje indica que el string de la facción del personaje en concreto no existe dentro del array de facciones del sistema solar, por lo que hay que modificarlo.

- Propósito del personaje: se comprueba que el propósito del personaje está dentro del rango del enumerado. Siendo el número máximo en esta versión el tres. Si se detecta un error se le devuelve el siguiente mensaje al LLM.

  ```
  In Character: characters[i].Name you set a purpose that is
  our of bounds and need to be a lower number between 0 and
  purposes. Siendo characters[i].Name el nombre del personaje
  en concreto y purposes el número máximo que puede tomar el
  parámetro.
  ```

  Este mensaje indica que el propósito del personaje en concreto está fuera del rango del enumerado y por lo tanto hay que cambiarlo.

- Valores de los enemigos del personaje: para cada personaje se comprueba que sus enemigos están representados por un número correcto, entendiendo por correcto que esté entre el cero y el número máximo de personajes existentes. Si se encuentra un error se devuelve el siguiente mensaje.

```
In Character: characters[i].Name its Enemy number j it´s
our of bounds and must be a number between 0 and + maxCharacter.
Donde characters[i].Name es el nombre del personaje en el
que se ha encontrado el error, j es el número del enemigo
que se ha identificado como erróneo y maxcharacter el número
máximo que puede adoptar.
```

Este mensaje indica que uno de los enemigos de un personaje en concreto no existe, es decir, si por ejemplo hubiese cuatro personajes en el sistema, el enemigo vendría indicado por un número inferior a cero o superior a tres y por lo tanto hay que cambiarlo.

- Existencia de la raza del personaje: se comprueba que la raza del personaje existe en la lista de razas del sistema solar. En caso de error se devolverá el siguiente mensaje.

```
In Character: characters[i].Name you added a race that doesnt
exist, set the characters race to one race of the races array.
Donde characters[i].Name es el nombre del personaje al que
se le ha encontrado dicho error.
```

Este mensaje indica que la raza del personaje en concreto no aparece en el array de razas del sistema solar y por lo tanto hay que sustituirla por una de las que aparecen en el array.

Si la generación pasa todas estas condiciones se devuelve un mensaje con OK y se procede a la validación narrativa del resultado.

3.4. Validación narrativa del resultado del LLM

En este apartado vamos a detallar la base de lo que le pedimos al LLM como una buena narrativa, enumerando los motivos por los que la consideramos insuficiente.

- Número de personajes: los personajes son el motor de la historia, pues son los que perpetran las acciones que llevan al cambio en una narrativa. Por este motivo nos aseguramos de que el resultado del LLM contenga al menos dos personajes.

- Número de planetas: un sistema solar sin planetas no sería distinto, en este contexto, a una hoguera sin nadie que cuente historias a su alrededor, es por esto que nos aseguramos de que el resultado contenga por lo menos un planeta.

- Número de facciones: la división política e ideológica es un tema tan antiguo como la raza humana, además de ser un potente motor para la trama de muchas historias, como puede ser Star Wars(LucasFilms (1977)) con la batalla entre el imperio y la república. Así pues, hemos decidido asegurar un mínimo de dos facciones en nuestra generación.

- Número de planetas habitados: sin vida inteligente, los planetas se reducen a una serie de reacciones químicas y meteorológicas que desde el aspecto físico pueden ser increiblemente interesantes. Sin embargo, desde un punto de vista narrativo puede resultar aburrido, pues como dice Christopher Vogler "En el corazón de toda gran historia hay un cambio", cosa que sin seres que actuen de forma consciente, se vuelve casi imposible de conseguir. Con esta premisa, aseguramos que nuestra generación contenga al menos dos planetas habitados.
- Relaciones entre los distintos planetas: la paz y la tranquilidad son dos cosas que el ser humano busca a diario en su vida, y aunque en muchos casos pueden parecer elementos de la ficción, cuando agarramos un libro o nos sentamos a ver una película, estos dos elementos son lo último que buscamos. Como dijo una vez Alfred Hitchcock "La tensión es lo que mantiene a la audiencia pegada a sus asientos... es el ingrediente esencial de cualquier historia memorable". A sabiendas de esto, hemos decidido asegurarnos que por lo menos dos planetas mantengan tensión política entre ellos.
- Número de enemistades: al igual que con el lado oscuro de la fuerza, todas las historias necesitan su lado oscuro para poder brillar, la contraposición de ideas ha sido el catalizador de la narrativa desde que Aquiles se embarcó hacia la guerra de Troya. Según dice Javier Marías "El villano no es quien hace el mal, sino quien nos hace pensar", por lo que hemos decidido asegurar que al menos dos personajes estén enemistados entre ellos dentro de nuestra narrativa.

Para cada uno de los casos se devuelve un elemento del siguiente enumerado.

```
public enum Reasons {
    Interesting_Story,
    Not_Enough_Planets,
    Not_Enough_Factions,
    Not_Enough_Habited,
    Not_Enough_Characters,
    Boring_Relations,
    Not_Enough_Enemies
}
```

Con los que enviamos los siguientes mensajes al LLM, respectivamente. Excluyendo el valor de InterestingStory, el cual indica que la entrada es apropiada y podemos proceder con ella.

- You need to add more planets to the system.
- You need to add more factions to the system
- More planets need to habited
- You need to add more characters to the system.
- You need to make some planets have tension or war between them.
- More characters should be enemies between them.

Capítulo 4

Generación del sistema solar con simulación física

Una vez nos aseguramos de que el archivo de entrada es válido, se escogen las masas, radios, y posiciones de los cuerpos dentro del sistema solar. Para esto utilizamos una serie de constantes que hacen las funciones de máximos y mínimos, además de constantes basadas en las masas de nuestro sol y nuestra luna, de tal manera que podamos calcular los radios y masas de las estrellas a través de reglas de tres sencillas, utilizando como referencia nuestros propios astros. Seguimos los siguientes pasos para determinar los parámetros de cada uno de los cuerpos dentro del sistema solar. A través de estar variables, generamos un sistema solar funcional y autónomo que funciona mediante físicas y no mediante trayectorias predefinidas, funcionando de tal manera que si el programa se deja corriendo durante el tiempo necesario todos los planetas del sistema acaban internándose en sus estrella. Cada cuerpo afecta sobre los demás, por lo que cada uno importa en las trayectorias del resto.

Para la generación de un sistema solar funcional, seguimos los pasos que se detallan a continuación.

- Generación del radio de la estrella: para escoger el radio de la estrella del sistema solar, se utiliza como entrada el parámetro `Star_Size` del esquema de entrada. De tal forma que la entrada se convierte al siguiente enumerado.

  ```
  public enum StarSize  Dawrf, Normal, Giant;
  ```

  Si se ha generado una estrella gigante, el tamaño de esta variará entre el tamaño máximo de una estrella, el cuál está asignado a 2000 unidades y este mismo número menos 300 unidades. Es decir un rango entre 1700 y 2000.

  Si se genera una estrella enana, el rango variará entre el tamaño mínimo, que en esta versión son 200 unidades y este mismo número más quinientas unidades. Es decir, un rango entre 200 y 700.

  Si se genera una estrella de tamaño normal, el rango estará comprendido entre 800 unidades y 1500 unidades.

  Dondes entendemos unidades siempre como las unidades de escala de Unity.

- Generación de la masa de la estrella: para asignarle una masa a la estrella se utiliza una regla de tres con la masa del sol de la siguiente manera.

$$F = \frac{rad \times sunMass}{sunRadious}$$

Donde rad es el radio de la estrella, sunMass es la masa del sol, lo que equivale a 333000 en unidades de Unity y sunRadious el radio del sol, que equivale a 696 en unidades de Unity.

- Distancia de los planetas con respecto a su estrella: escogemos la distancia a la que estará el planeta de la estrella, para ello nos basamos en el parámetro planetType de cada uno de los planetas, el cual convertimos al siguiente tipo enumerado.

```
public enum PlanetType  Fire, Water, Gas, Frozen;
```

De tal manera que si el tipo del planeta es Fire se situará a una distancia máxima de minDis, la cual se calcula de la siguiente manera.

```
int minDist = (starRad * sunMinHabDistance) / sunRadious;
```

Donde starRad es el radio de la estrella, sunMinHabDistance es la distancia mínima habitable para nuestro sol, la cual equivale a 1200 unidades de Unity, y sunRadious que es el radio de nuestro sol, el cual equivale a 696 unidades de Unity.

Y la distancia mínima del planeta vendrá determinada por la variable minFire, la cual se calcula de la siguiente manera.

```
int minFire = minDist - starRad/2;
```

Donde minDist es la variable calculada anteriormente y starRad es el radio de la estrella, igual que en la variable anterior.

Si el tipo del planeta es Water, la distancia variará en un rango comprendido entre miDist, que se calcula de la misma manera que antes y maxDist, la cual se calcula de la siguiente manera.

```
int maxDist = (starRad * sunMaxHabDistance) / sunRadious;
```

Donde starRad es el radio de la estrella, sunMaxHabDistance es la distancia máxima a la que un planeta puede estar del sol para ser habitable, la cual equivale a 2000 en unidades de Unity y sunRadious es el radio del sol, el cual equivale a 696 unidades de Unity.

Si el planeta es de tipo Gas la distancia estará entre maxDist y maxDist * 2, que se calcula de la misma manera que en el caso anterior.

Si el planeta es de tipo Frozen la distancia estará entre maxDist * 2 y maxDist * 4, que se calcula igual que en el caso de Water.

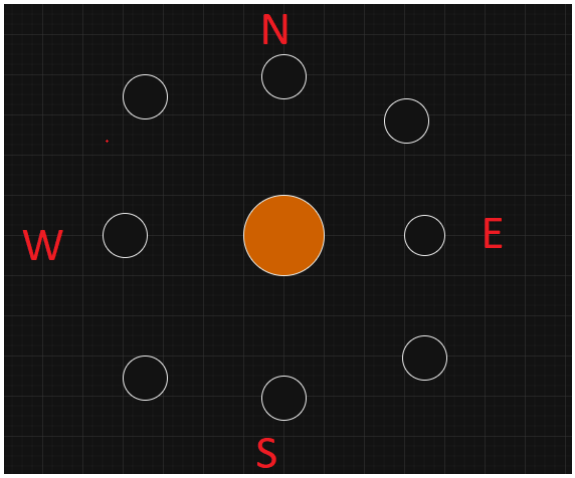

Figura 4.1: Diagrama en el que se muestra la generación para cada tipo de planeta

- Posición incial de los planetas con respecto a la estrella: para que los planetas tengan posiciones coherentes y no choquen entre si en la mayoría de los casos, los planetas del mismo tipo se posicionan en torno a la estrella siguiendo sus puntos cardinales, colocando los cuatro primeros en lo que serían las posiciones norte, este, sur y oeste. Y los cuatro siguientes en noreste, sureste, suroeste y noroeste. Consiguiendo como resultado final que; habiendo ocho planetas del mismo tipo, los planetas estén separados entre sí en torno al planeta como se muestra en la figura 4.1 Como se ha mencionado, esto ocurrirá para cada tipo de planeta, en caso de que hubiese ocho planetas de cada tipo, obtendriamos cuatro formaciones similares a la de la figura, solo que cada vez más lejanas a la estrella.

- Radio de los planetas: para asignar el radio de cada planeta utilizamos como entrada el parámetro Giant de cada planeta. En función de este parámetro contemplamos dos posibilidades. En caso de que Giant sea true y por lo tanto el planeta sea gigante, su tamaño se calculará en un rango aleatorio entre 500 unidades de Unity y 1000 unidades de unity. En caso de que no sea un planeta gigante, el rango estará entre 10 unidades de Unity y 500 unidades de Unity.

- Masa de los planetas: para calcular la masa de los planetas dividimos el radio del planeta en concreto entre el radio de la tierra, que en unidades de Unity equivale a 6.3. Es suficiente con hacer esta división porque la masa de la tierra equivale a 1 en unidades de Unity, ya que la hemos tomado como unidad de referencia.

- Distancia de los satélites con respecto a sus planetas: Para calcular la distancia a la que colocamos los satélites de sus respectivos planetas se utiliza un rango aleatorio, dónde el minimo viene expresado por la siguiente suma.

  ```
  planetPos + (planetRad * 2)
  ```

  Donde planetPos es la posición del planeta al cual orbita el satélite y planetRad es el radio de dicho planeta. Y cuyo máximo viene expresado por la siguiente

suma.

```
planetPos + (planetRad * 4)
```

Donde las variables representan lo mismo que para el mínimo.

- Tamño de los satélites: para asignar los tamaños de los satélites de los planetas, a cada uno se le asigna un número de un rango aleatorio comprendido entre las siguientes expresiones. Donde el mínimo viene representado por la siguiente división.

  ```
  planetRad / 15
  ```

  Donde planetRad es el radio del planeta que orbita el satélite. El máximo viene dado por una expresión similar.

  ```
  planetRad / 2
  ```

- Masas de los satélites: la masa de cada uno de los satélites viene dada por la siguiente expresión.

  ```
  (satelliteRad * moonMass) / moonRad;
  ```

  Donde satelliteRad es el radio del satélite cuyo masa estamos calculando, moonMass es la masa de la luna, que en unidades de Unity equivale a 0.01. Y moonRad es el radio de la luna, que en unidades de Unity equivale a 1.7.

- Posicionamiento de los satélites: se sigue la misma norma que con los planetas, de tal manera que podemos esperar una generación similar a la imagen 4.1 tomando el centro de la imagen como el planeta y los círculos de alrededor como sus satélites.

Una vez asignadas todas estas variables, se puede comenzar la generación física.

4.1. Generación del sistema

Para la generación del sistema tomamos los valores descritos en el apartado anterior y seguimos los siguientes pasos con el objetivo de crear la estrella, los planetas y sus satélites. A los cuerpos del planeta los denominaremos celestiales, estos serán los objetos que tendrá en cuenta el sistema más adelante a la hora de hacer los cálculos físicos.

- Generación de las estrellas: se ha tomado la decisión de limitar la generación de estrellas a solo una, ya que al hacer pruebas con más los planetas salen despedidos fuera de sus órbitas o se meten dentro de las estrellas, haciendo que la narración no sea coherente y por lo tanto imposibilitando el uso de la herramienta en estos casos. Por tanto, se crea el objeto de la estrella, un prefab de Unity cuyas propiedades son las del sol. A este objeto se asigna el tamaño de la estrella en todos los ejes de su escala, después le asignamos su masa y lo reportamos al sistema como celestial.

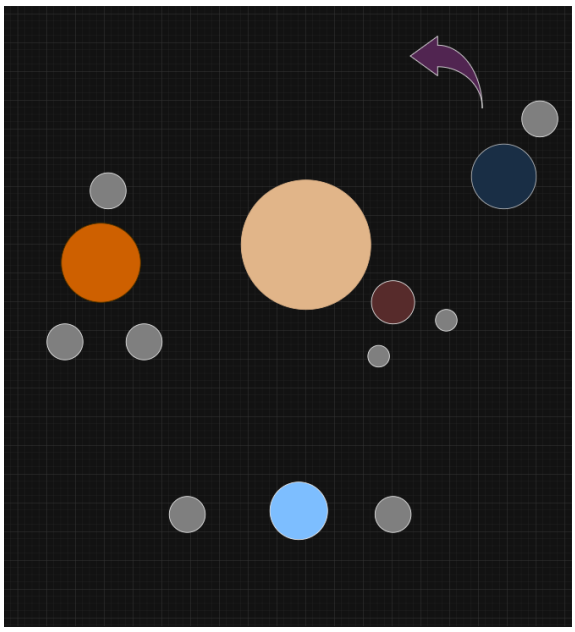

Figura 4.2: Diagrama de la simulación esperada. Dónde el objeto central reprensenta la estrella, los círculos rojo, azul marino, naranja y azul celeste representan los cuatro tipos de planetas, y los grises sus correspondientes satélites

- Generación de los planetas: creamos un objeto prefab de Unity cuyos valores son los de la tierra. Se reporta el objeto como celestial y se le asigna su radio. Se asigna el valor de este radio a todos los ejes de la escala del objeto y se asigna su masa. Acto seguido se le asigna su información, que es una clase que contiene tanto la información física del planeta(radio, masa, etc) como su información narrativa(gobierno, recursos, etc). Luego añadimos los personajes a esta misma clase y le asignamos su material en función de su tipo de planeta.

- Generación de los satélites: para la generación de los satélites se utiliza un módulo adicional, al cual se le asignan los valores de las posiciones, radios y masas de los satélites del planeta en concreto antes de comenzar la generación. Una vez hecho esto, el módulo itera sobre los satélites, creando un objeto prefab con los valores de la luna para cada uno. Luego lo reporta como celestial al sistema, le asigna su radio, le asigna dicho radio a los valores de la escala y le asigna su masa.

Una vez creados los objetos celestiales del sistema, podemos dar comienzo a la simulación física.

4.2. Simulación física y exploración del sistema

Una vez creados todos los cuerpos de la simulación, les otorgamos una velocidad incial para que comiencen a rotar en torno a su estrella madre y en el caso de los satélites a su planeta. Para lograr esto, iteramos dos veces sobre los cuerpos del

sistema para aplicarle a cada cuerpo la fuerza que los demás ejercen sobre él. Esta fuerza la calculamos mediante la siguiente ecuación.

$$\sqrt{\frac{G * mass2}{r}}$$

Siendo:

- G la constante de la gravitación universal, $\approx 6{,}674 \times 10^{-11}\, m^3\, kg^{-1}\, s^{-2}$
- mass2 la masa del segundo cuerpo, es decir el que ejerce la fuerza. r la distancia entre ambos cuerpos.

Esta fuerza se le suma a la velocidad de cada cuerpo para que comiencen a rotar.

Tras comenzar la rotación empleamos la siguiente fórmula a cada iteración de la simulación.

$$F = G\frac{m_1 \times m_2}{d^2}$$

Siendo:

- G la constante de la gravitación universal, G $\approx 6{,}674 \times 10^{-11}\, m^3\, kg^{-1}\, s^{-2}$
- m_1 y m_2 representan las masas de ambos cuerpos.
- Y d^2 la distancia entre ambos cuerpos al cuadrado.

Con esta fórmula mantenemos la velocidad de rotación de los cuerpos durante toda la simulación. Mientras los cuerpos rotan dejarán una estela morada tras de ellos para que el usuario pueda seguir de manera sencilla su trayectoria. De esta manera y tras haber seguido los pasos anteriormente descritos obtendriamos un resultado como el mostrado en la figura 4.3.

Una vez en el sistema solar el jugador puede desplazarse con las teclas WASD y girar la cámara moviendo el ratón. Al acercarse a un planeta el texto de la parte superior izquierda de la pantalla en el que en la figura 4.3 pone Planet Information, se mostrarán algunos de los parámetros del planeta, en concreto la temperatura, los recursos y la superficie. También muestra que pulsando la tecla L el usuario puede aterrizar en el planeta. Un ejemplo de esto se muestra en la figura 4.4

Si el usario aterriza en uno de los planetas se muestra una escena sencilla en la que en esta versión del proyecto lo único que puede hacer es caminar. Esta escena es distinta para cada tipo de planeta, salvo para los planetas gaseosos, en los que no se puede aterrizar. En la escena también se crea un rectángulo de color gris que representa la nave espacial, si el usuario se acerca y pulsa la tecla T volverá a la escena del sistema solar. A continuación se muestra una imagen de cada una de las escenas de los planetas.

Al ser una herramienta pensada principalmente para diseñadores, el usuario también cuenta con una interfaz en la que podrá ver todas las características del sistema solar. Esta interfaz se abre y se cierra pulsando la tecla TAB. Se muestra un ejemplo de esta interfaz en la imagen 4.8

Dentro de esta interfaz se muestran todos los planetas del sistema solar por orden de proximidad a la estrella. Dentro de cada planeta encontraremos una interfaz

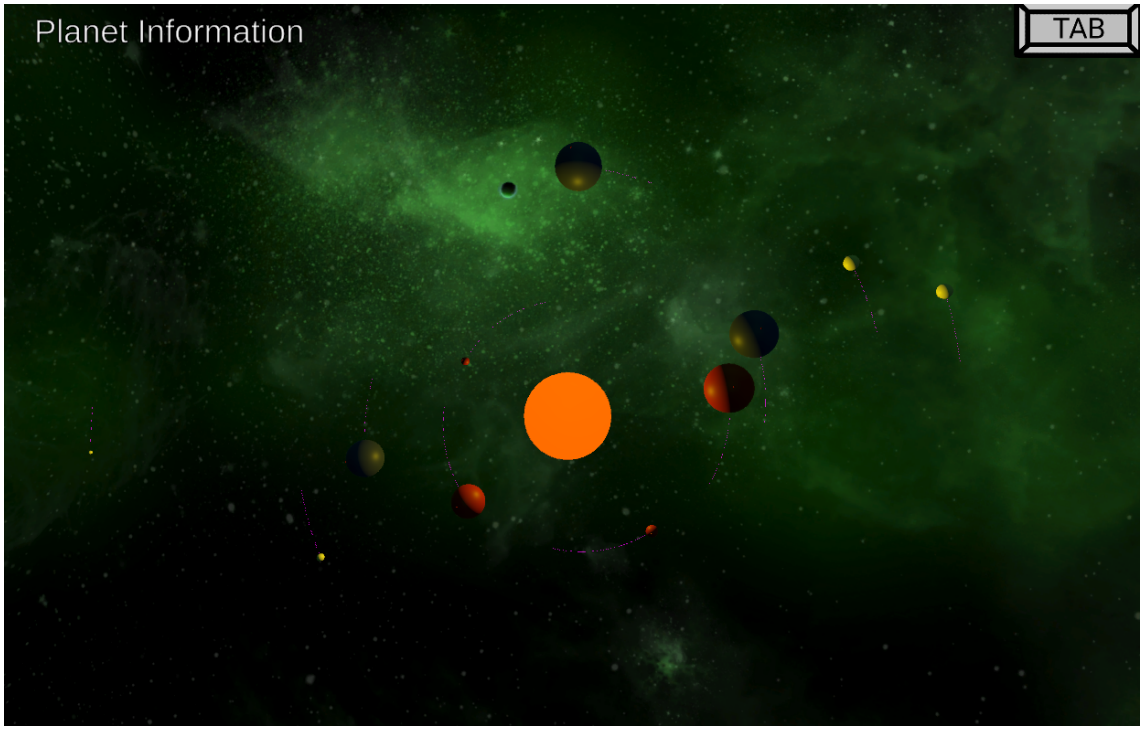

Figura 4.3: Ejecución de ejemplo del programa

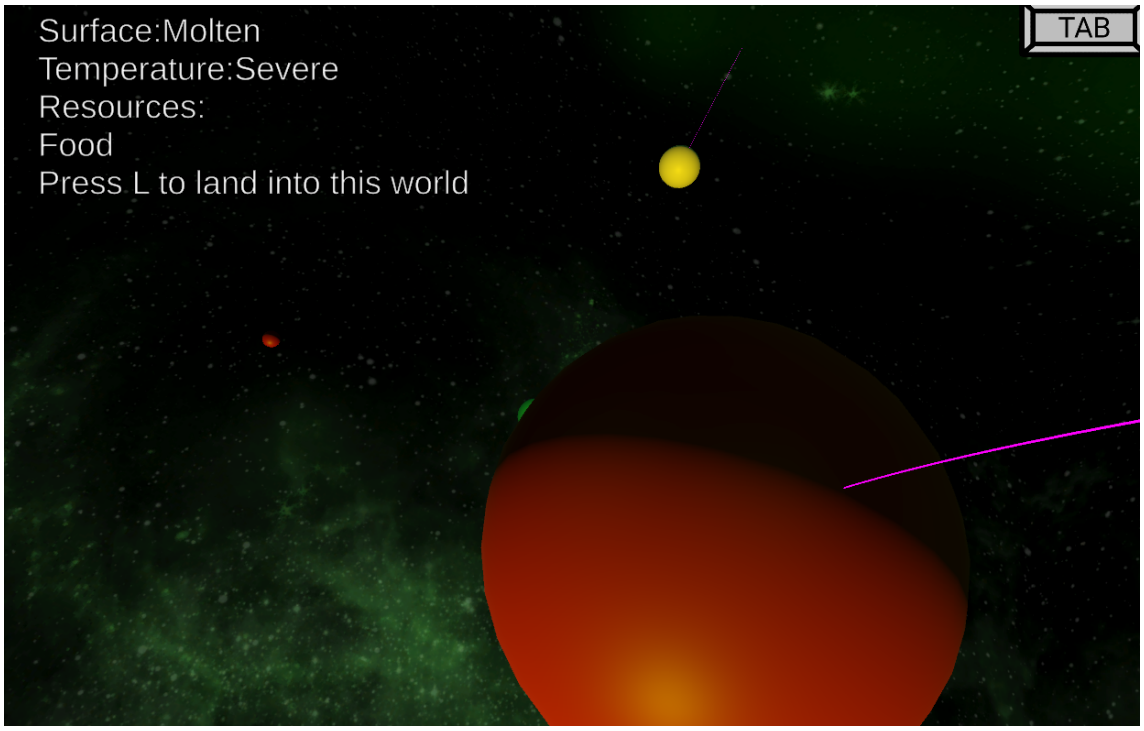

Figura 4.4: Ejemplo de la información que puede ver el usuario al acercarse a uno de los planetas de la simulación.

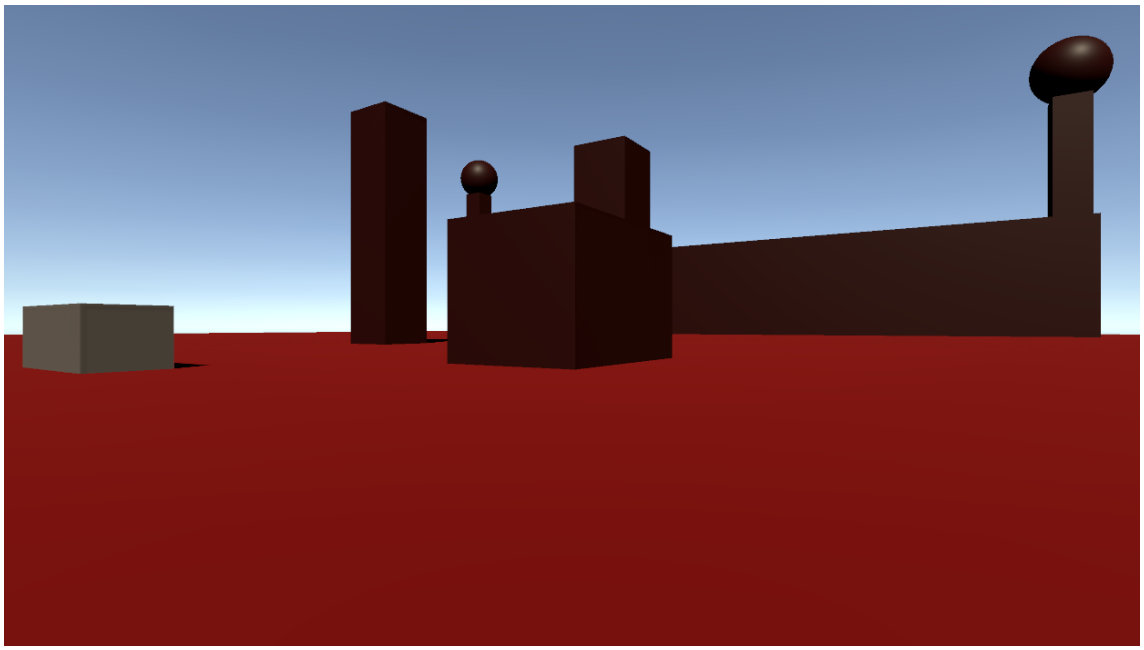

Figura 4.5: Captura de pantalla de la escena correspondiente a los planetas de fuego

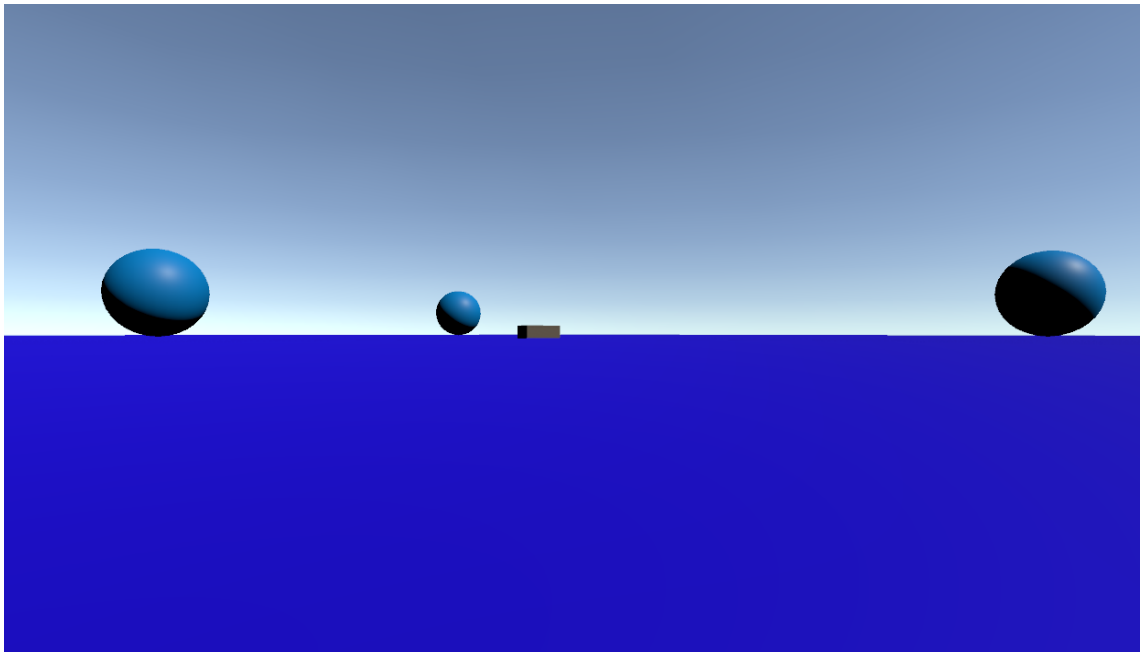

Figura 4.6: Captura de pantalla de la escena correspondiente a los planetas de agua

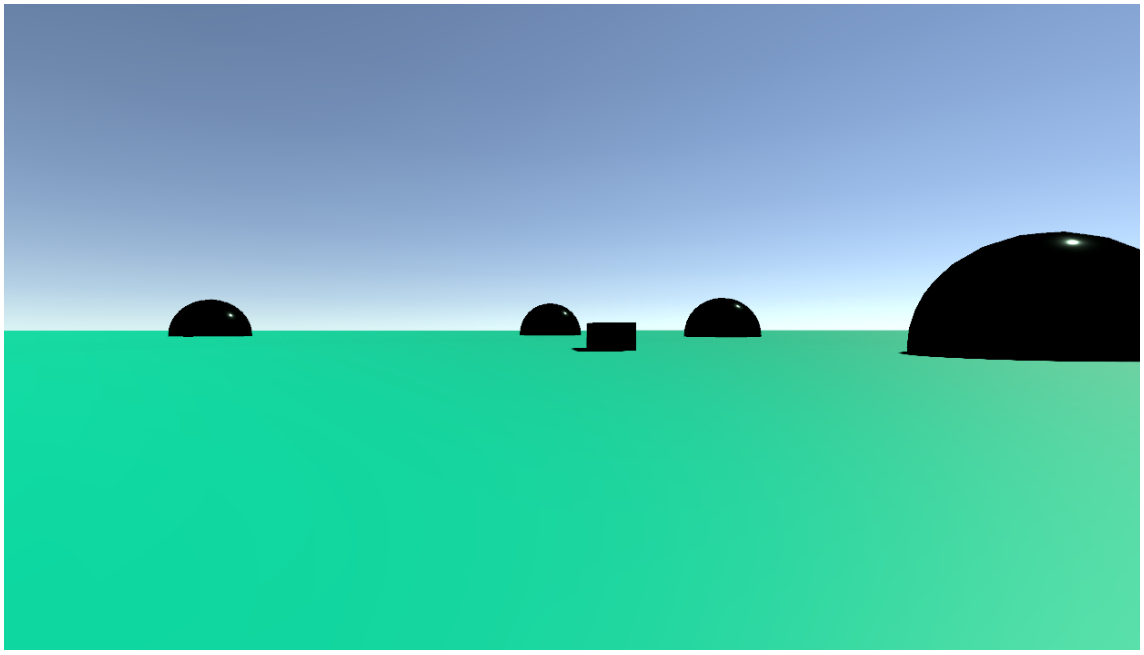

Figura 4.7: Captura de pantalla correspondiente a los planetas de hielo

Figura 4.8: Ejemplo de la interfaz donde el usuario puede ver las características del sistema solar

Figura 4.9: Ejemplo de interfaz donde podemos ver la información del planeta

como la que se muestra en la figura 4.9. Los botones de la parte inferior de la pantalla que se ve en la figura 4.8 funcionan al igual que los planetas, pero muestran, respectivamente, las razas del sistema solar, las facciones y los personajes. Un ejemplo de la interfaz que muestra las razas aparece en la figura 4.10. Un ejemplo de la interfaz que muestra las facciones aparece en la figura 4.10. Por último tenemos la interaz dónde se muestran los personajes, la cual podemos ver en la figura 4.11. La interfaz que muestra los personajes funciona de manera similar a la de los planetas, lo que significa que podemos hacer clic sobre cada uno de ellos para mostrar sus características concretras. Podemos encontrar un ejemplo de esto en la figura 4.12.

Para minimizar la cantidad de texto empleada se utilizan iconos para identificar diferentes características de los planetas. A continuación especificaremos que significa cada uno de esos iconos.

En las figuras 4.13, 4.14, 4.15 y 4.16 podemos ver los iconos empleados para desginar los tipos de planetas. En la figura 4.17 podemos ver el icono empleado para indicar el número de satélites que tiene cada planeta. En las figuras 4.18, 4.19, 4.20 y 4.21 aparecen los iconos utilizados en la interfaz de la figura 4.9 para indicar el tipo de planeta que estamos inspeccionando.En las figuras 4.22, 4.23, 4.24 y 4.25 podemos ver los iconos empleados para designar las relaciones que los planetas tienen entre sí. En las figuras 4.26, 4.28, 4.27, 4.29,4.30, 4.31 y 4.32 podemos ver los iconos utilizados para indicar el tipo de gobierno que hay en un planeta en concreto. Por último, en las figuras 4.33, 4.34, 4.35, 4.36 y 4.37 podemos ver los iconos utilizados para indicar los recursos que podemos encontrar en un planeta en concreto.

Una vez presentado el trabajo realizado podemos dar paso a los experimentos y

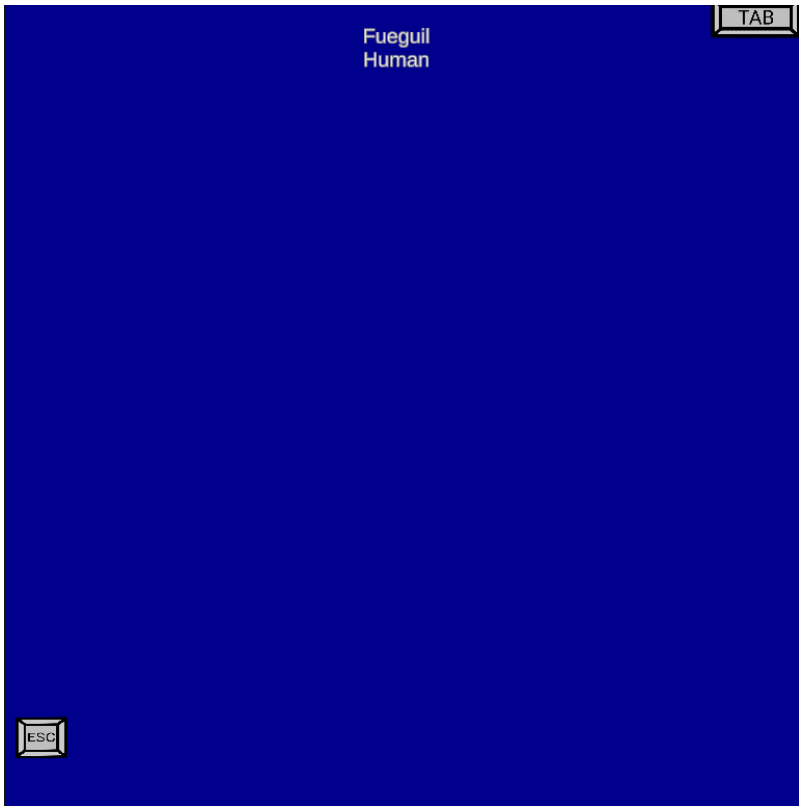

Figura 4.10: Ejemplo de la interfaz que muestra las razas del sistema solar

Figura 4.11: Ejemplo de la interfaz que muestra los personajes del sistema solar

Figura 4.12: Ejemplo de la interfaz que muestra las características de cada uno de los personajes.

Figura 4.13: Icono empleado para designar los planetas de fuego en la interfaz mostrada en la figura 4.8

Figura 4.14: Icono empleado para desginar los planetas de agua en la interfaz mostrada en la figura 4.8

Figura 4.15: Icono empleado para designar los planetas gaseosos en la interfaz mostrada en la figura 4.8

Figura 4.16: Icono empleado para designar los planetas de hielo en la interfaz mostrada en la figura 4.8

Figura 4.17: Icono empleado para designar cada uno de los satélites de cada planeta, por cada satélite que tenga el planeta aparecerá un icono al lado del texto satellites que aparece en la figura 4.9

Figura 4.18: Icono empleado para indicar que el planeta es de fuego, este icono aparece al lado del texto Planet Type que podemos ver en la figura 4.9 cuando el planeta es de fuego

Figura 4.19: Icono empleado para indicar que un planeta es de agua, este icono aparece al lado del texto Planet type que podemos ver en la figura 4.9 cuando el planeta es de agua

Figura 4.20: Icono empleado para indicar que el planeta es gaseoso, este icono aparece al lado del texto Planet type que podemos ver en la figura 4.9 cuando el planeta es gaseoso

Figura 4.21: Icono empleado para indicar que el planeta es de hielo, este icono aparece al lado del texto Planet Type que podemos ver en la figura 4.8 cuando el planeta es de hielo

Figura 4.22: Icono empleado para indicar que un planeta es del mismo territorio que el que está inspeccionando. Este icono aparece al lado del texto Relations en la posición correspondiente al planeta en la interfaz que aparece en la figura 4.9. Es decir, si estamos inspeccionando el planeta 0 y este icono se encuentra en la posición 2, eso significa que el planeta 0 y el planeta 2 pertenecen al mismo territorio.

Figura 4.23: Icono empleado para indicar que un planeta está en paz con otro. Este icono aparece al lado del texto Relations en la posición correspondiente al planeta en la interfaz que aparece en la figura 4.9. Es decir, si estamos inspeccionando el planeta 0 y este icono se encuentra en la posición 2, eso significa que el planeta 0 y el planeta 2 están en paz.

Figura 4.24: Icono empleado para indicar que un planeta está en tensión política con otro. Este icono aparece al lado del texto Relations en la posición correspondiente al planeta en la interfaz que aparece en la figura 4.9
. Es decir, si estamos inspeccionando el planeta 0 y este icono aparece en la posición 2, eso significa que el planeta 0 y el planeta 2 están bajo tensión política.

Figura 4.25: Icono empleado para indicar que un planeta está en guerra con otro. Este icono aparece al lado del texto Relation en la posición correspondiente al planeta en la interfaz que aparece en la figura 4.9. Es decir, si estamos inspeccionando el planeta 0 y este icono aparece en la posición 2, eso significa que el planeta 0 y el planeta 2 están en guerra.

Figura 4.26: Icono empleado para indicar que el gobierno del planeta es una monarquía, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es una monarquía.

Figura 4.27: Icono empleado para indicar que el gobierno del planeta es una república, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es una república.

Figura 4.28: Icono empleado para indicar que el gobierno del planeta es una democracia, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es una democracia.

Figura 4.29: Icono empleado para indicar que el gobierno del planeta es autoriatario fascista, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es autoritario fascista.

Figura 4.30: Icono empleado para indicar que el gobierno del planeta es autoriatario comunista, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta es autoritario comunista.

Figura 4.31: Icono empleado para indicar que el gobierno del planeta se basa en clanes guerreros, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el gobierno de dicho planeta se basa en clanes guerreros.

Figura 4.32: Icono empleado para indicar que el planeta en concreto no está habitado y por lo tanto no tiene gobierno, este icono aparece al lado del texto Goverment que aparece en la interfaz que se muestra en la figura 4.9 cuando el planeta en concreto no está habitado.

Figura 4.33: Icono empleado para indicar que el planeta en concreto tiene comida como recurso. Este icono aparece al lado del texto Resources en la interfaz de la figura 4.9 cuando el planeta tiene comida como uno de sus recursos.

Figura 4.34: Icono empleado para indicar que el planeta en concreto tiene minerales como recurso. Este icono aparece al lado del texto Resource en la interfaz de la figura 4.9 cuando el planeta tiene minerales como uno de sus recursos

Figura 4.35: Icono empleado para indicar que el planeta en concreto tiene magia como recurso. Este icono aparece al lado del texto Resource en la interfaz de la figura 4.9 cuando el planeta tiene magia como uno de sus recursos

Figura 4.36: Icono empleado para indicar que el planeta en concreto tiene armas como recurso. Este icono aparece al lado del texto Resource en la interfaz de la figura 4.9 cuando el planeta tiene armas como uno de sus recursos

Figura 4.37: Icono empleado para indicar que el planeta en concreto tiene esclavos como recurso. Este icono aparece al lado del texto Resource en la interfaz de la figura 4.9 cuando el planeta tiene esclavos como uno de sus recursos

las conclusiones sacadas de estos.

Capítulo 5

Experimentos y análisis resultados

En este capítulo explicaremos con detalle los experimentos llevados a cabo con la herramienta y profundizaremos en los resultados de los mismos. Se detallarán los pasos que se han seguido a la hora de hacer los experimentos, el número de usuarios que ha participado y se describirán los resultados observados además de los obetnidos mediante encuestas.

5.1. Descripción del experimento

Durante la prueba se le pide al usuario que se tome tres minutos para desarrollar una historia de carácter ficticio, en concreto, basada en el género de la ciencia ficción. Se le dan como ejemplos Star Wars (LucasFilms (1977)) y Dune (Frank Herbert (1965)). También se le especifica al sujeto que no tiene porque terminarla, además de que puede usar la aplicación que mejor le parezca para escribir, o que puede incluso hacerlo a papel.

Es importante mencionar que lo ideal para el experimento hubiese sido pedir a cada sujeto que se tomase por lo menos un día entero para escribir, pero por falta de presupuesto y de disponibilidad de los sujetos no ha sido posible. También es importante que, como se mencionó en capítulos anteriores, no disponíamos de una clave para el API ChatGPT, por lo que se ha tenido que simular el experimento mendiante un chat privado como se describe más adelante.

Una vez el sujeto ha escrito su historia, le pedimos que nos la escriba como si fuese una petición al LLM por chat privado. Una vez tenemos el mensaje lo añadimos al prompt descrito en el capítulo anterior y se lo enviamos al LLM. Una vez recibida la respuesta, se la enviamos al sujeto a través de un documento de texto, le indicamos que lo copie y lo pegue en el apartado Json Text del script Solar SystemPhysics del objeto SolarSystem de la escena.

Después se le pide que ejecute el programa y se le explican los controles y lo que puede hacer en el entorno. Una vez ha examinado el entorno y los elementos de la interfaz narrativa, pasando por los planetas y los personajes, se le pregunta si hay algo que no coincide. De ser la respuesta afirmativa, se le pide al sujeto que haga una corrección en el chat privado anteriormente mencionado como si se tratase de

ChatGPT. Este proceso se repite hasta que el sujeto está satisfecho con el resultado, contando el número de veces que ha sido necesario corregir al LLM para obtener un resultado satisfactorio.

Una vez el sujeto declara que está satisfecho con lo que ha producido la herramienta se da por concluido el experimento y se le hace responder dos encuestas. Una demográfica y una sobre su experiencia con la herramienta, a continuación detallaremos exactamente que se pregunta en cada una de ellas y cuales han sido los resultados.

5.2. Cuestiones planteadas a los usuarios después del experimento

Comenzando por el formulario demográfico, vamos a enumerar las preguntas que realizamos a los usuarios una vez han terminado el experimento con la prueba.

- Edad, cuya media es de 22.625 y cuya desviación estándar es de 0.599. Resultados en la figura 5.1
- Sexo, cuya mayoría ha sido de un 75 % de hombresm un 12.5 % de mujeres, un 6.25 % que prefieren no contestar y un 6.25 % que han contestado otro. Resultados en la figura 5.2
- Del 1 al 5 ¿Cómo calificarías tu experiencia como programador? Cuya media es de 3.875 y cuya desviación estándar es de 0.875. Resultados en la figura 5.3
- Del 1 al 5 ¿Cómo calificarías tu experiencia como diseñador de videojuegos? Cuya media es de 2.875 y cuya desviación estándar es de 1.14. Resultados en la figura 5.4
- Del 1 al 5 ¿Cómo calificarías tu conocimiento sobre narrativa? Cuya media es de 2.375 y cuya desviación estándar es de 1.06 Resultados en la figura 5.5
- Del 1 al 5 ¿Cómo calificarías tus conocimientos sobre Unity? Cuya media es de 3.975 y cuya desviación estándar es de 0.56 Resultados en la figura 5.6
- Estudios o trabajo actual. Resultados en las figuras 5.7 y 5.8.

En cuanto a la encuesta referente al uso de la herramienta, se plantearon las siguientes preguntas, en las que cuanto más alto es el número más de acuerdo se encuentra el usuario con la pregunta que se está planteando.

- Del 1 al 5 ¿Cómo de parecido ha sido el resultado con respecto a lo que esperabas? Cuya media es de 4.125 y cuya desviación estándar es de 1.79. Resultados en la figura 5.9
- Del 1 al 5 ¿Cómo de satisfecho estás con el resultado? Cuya media es de 3.8125 y cuya desviacioón estándar es de 0.99. Resultados en la figura 5.10

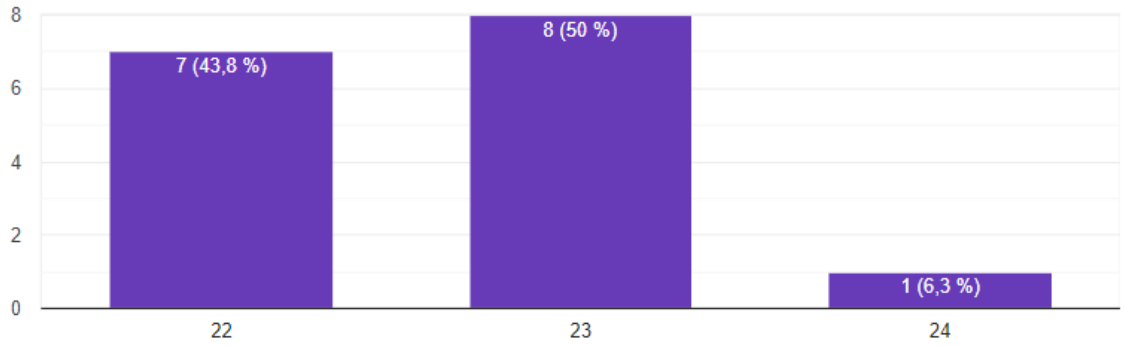

Figura 5.1: Gráfico que muestra los resultados obetenidos en la pregunta de edad de la encuesa demográfica

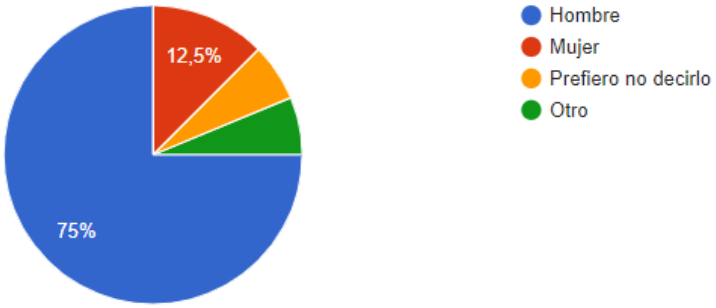

Figura 5.2: Gráfico que muestra los resultados obtenidos en la pregunta de sexo de la encuesta demográfica

- Del 1 al 5 ¿Cómo de útil te parece esta herramienta para el desarrollo de videojuegos? Cuya media es de 4.375 y cuya desviación estándar es de 0.88. Resultados en la figura 5.11
- ¿Usarías este sistema para un juego de jam? Resultados en la figura 5.12
- Sabiendo que se pueden modificar las escenas de los planetas en los que aterrizas. ¿Usarías esta herramienta para el desarrollo completo de un videojuego? Resultados en la figura5.13
- ¿Cuánto tiempo crees que hubieses tardado en generar y almacenar la información qué te ha aportado la herramienta? Resultados en la figura 5.14
- Describe tu experiencia con lo mejor y lo peor de la herramienta. Resultados en las figuras 5.15, 5.16, 5.17, 5.18, 5.19 y 5.20.

5.3. Resultados de las encuestas

En cuanto a las correcciones necesarias para que el LLM genere un sistema que se asemeje a lo que pedía el usuario, cuya media ha resultado ser de 1.6 correcciones,

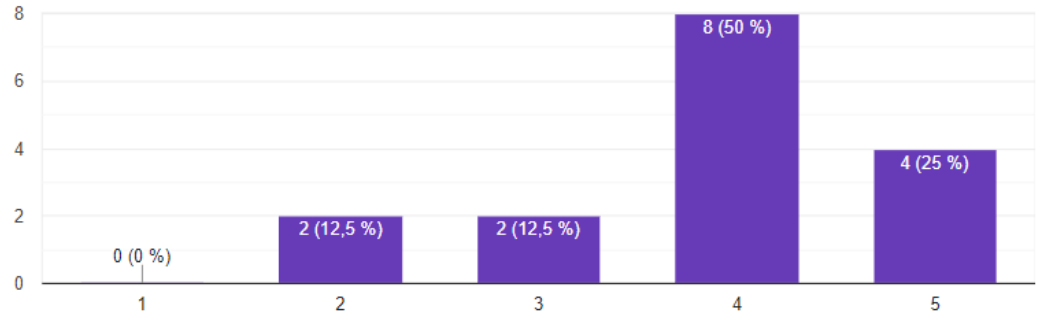

Figura 5.3: Gráfico que muestra los resultados obtenidos en la pregunta de experiencia como programador de la encuesta demográfica

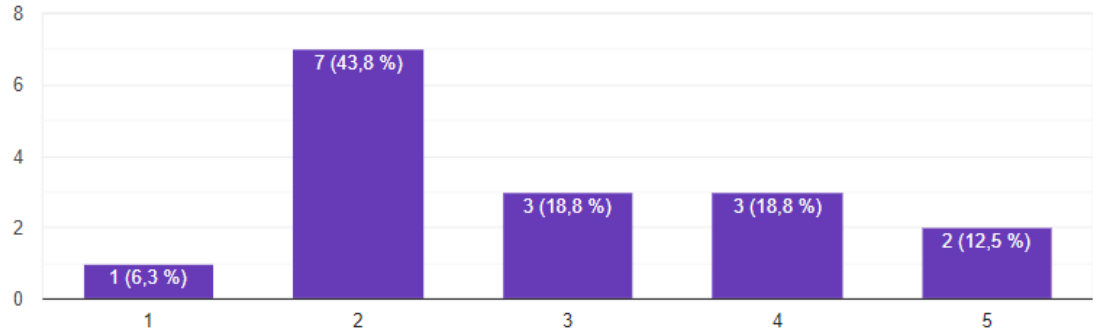

Figura 5.4: Gráfico que muestra los resultados obtenidos en la pregunta de experiencia como diseñador de videojuegos en la encuesta demográfica

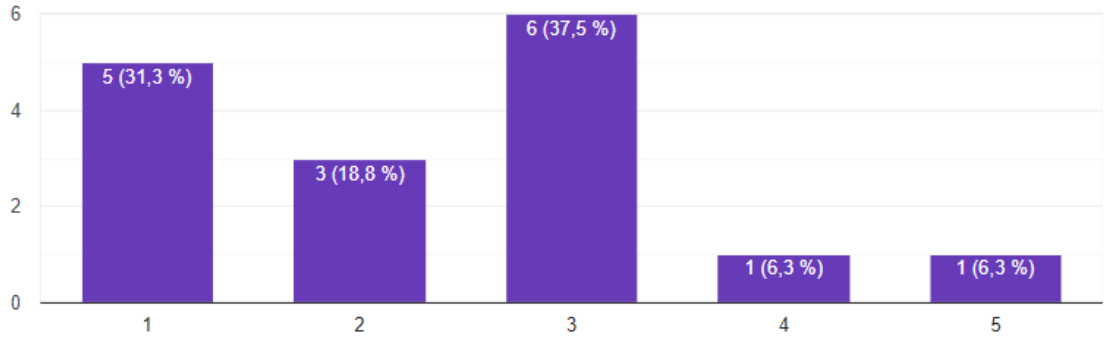

Figura 5.5: Gráfico que muestra los resultados obtenidos en la pregunta de conocimientos sobre narrativa de la encuesta demográfica.

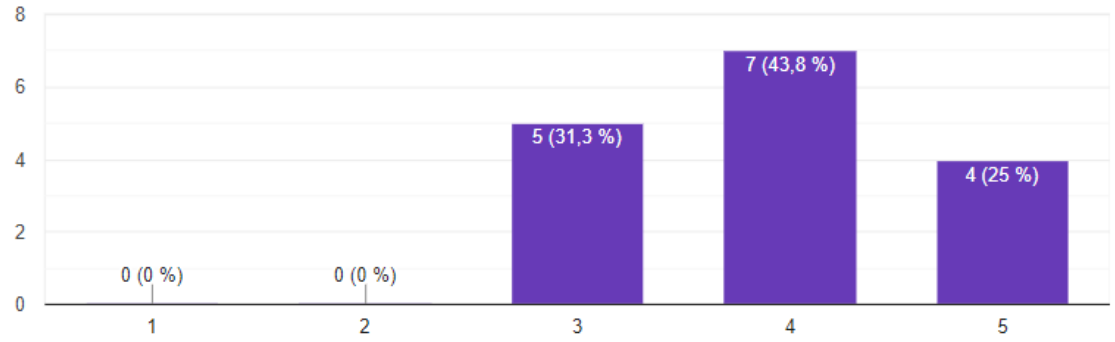

Figura 5.6: Gráfico que muestra los resultados obtenidos en la pregunta de conocimiento sobre Unity de la encuesta demográfica

Desarrollo de Videojuegos UCM

Programador de sonido en Videojuegos

Estudiante de grado en videojuegos y diseñador y artista en 2 juegos en producción

Estudio desarrollo de videojuegos

Software developer

Estudiante de desarrollo de videojuegos

Experiencia como artista técnico y actualmente programador en desarrollo de software para estructuras navales.

Figura 5.7: Primera parte del resultado de la pregunta referente a estudios o trabajo actual de la encuesta demográfica.

Programador GamePlay Unity

Programador C++

Estudio un master de arte y diseño visual para videojuegos.

Estudiante de Desarrollo de Videojuegos UCM

Grado en Desarrollo de Videojuegos

Master

Analista Ciberseguridad

Ingenieria software

Ingeniero de Software

Figura 5.8: Segunda parte del resultado de la pregunta referente a estudios o trabajo actual de la encuesta demográfica.

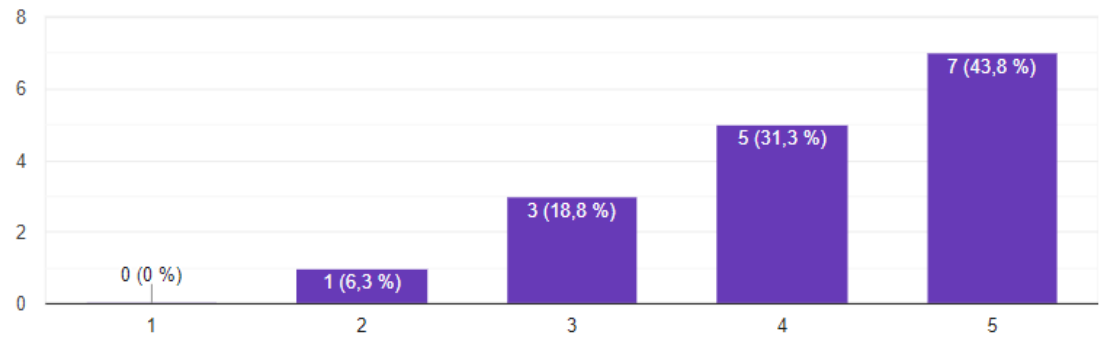

Figura 5.9: Gráfico que muestra los resultados de la pregunta referente al parecido que el usuario ha encontrado entre lo que tenía en mente y el resultado de la herramienta. Perteneciente a la encuesta de uso.

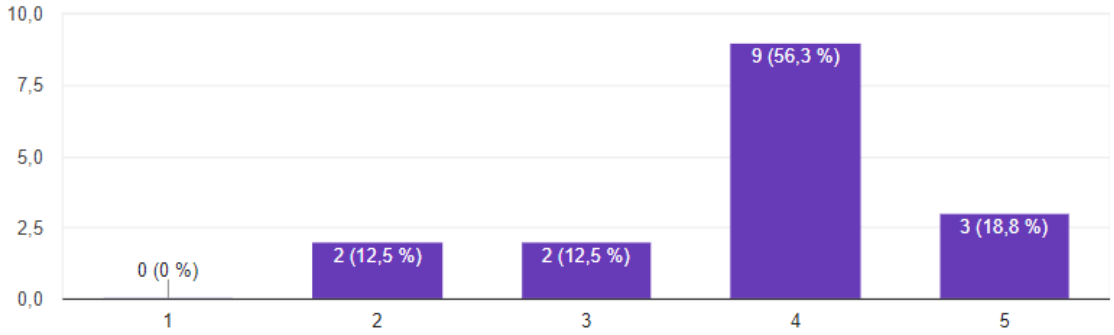

Figura 5.10: Gráfico que muestra los resultados de la pregunta referente a como de satisfecho ha estado el usuario con respecto al resultado de la herramienta. Perteneciente a la encuesta de uso.

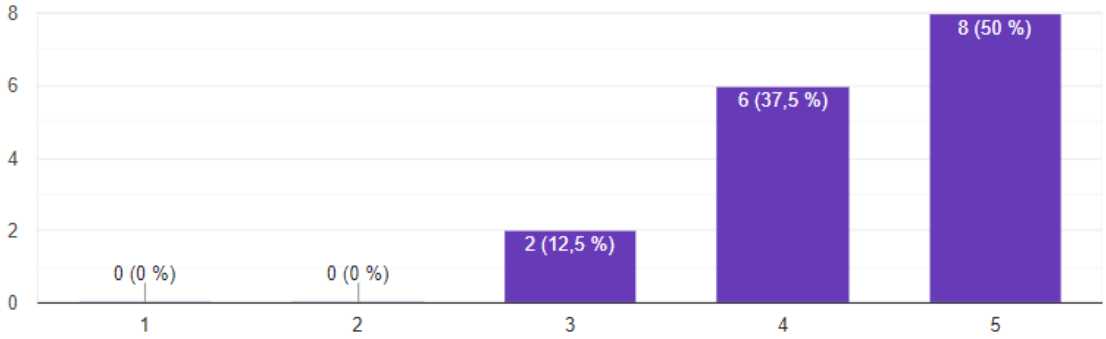

Figura 5.11: Gráfico que muestra los resultados de la pregunta referente a como de útil ha encontrado el usario la herramienta para el desarrollo de videojuegos. Perteneciente a la encuesta de uso.

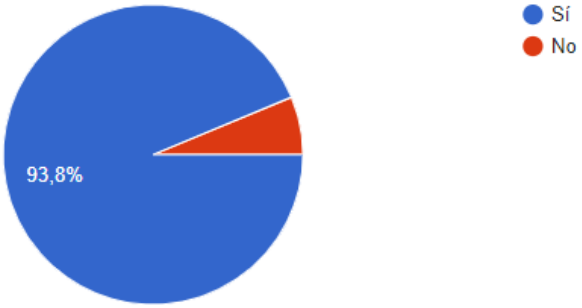

Figura 5.12: Gráfico que muestra los resultados de la pregunta referente a como de útil han encontrado los usuarios la herramienta para desarrollar un juego en una game jam. Perteneciente a la encuesta de uso

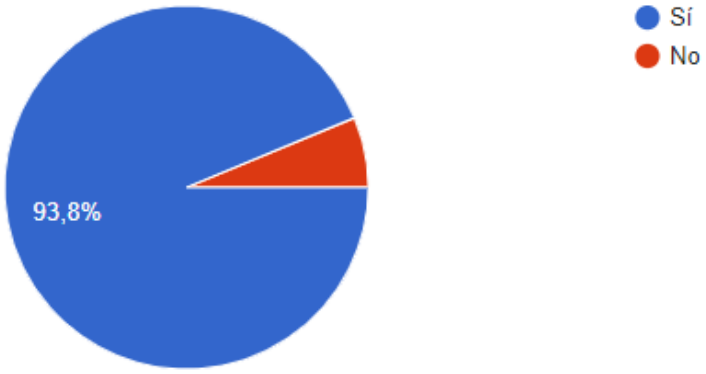

Figura 5.13: Gráfico que muestra los resultados obtenidos de la pregunta referente a como de útil han encontrado los usuarios la herramienta a sabiendas de que se mueden alterar las escenas de los planetas. Perteneciente a la encuesta de uso.

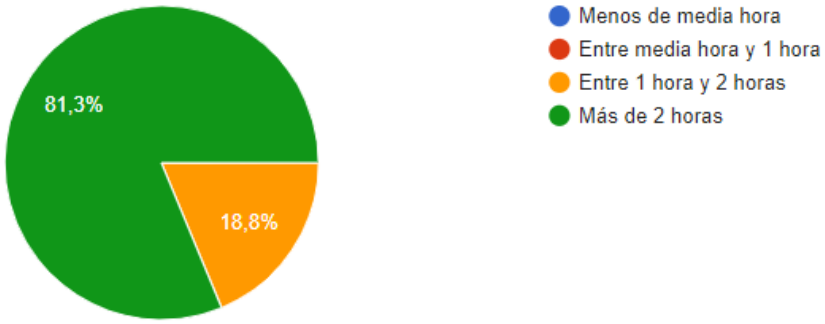

Figura 5.14: Gráfico que muestra los resultados obtenidos de la pregunta referente al tiempo que los usarios creen que hubiesen tardado en generar el contenido que la herramienta les ha facilitado. Perteneciente a la encuesta de uso.

Me parece una herramienta muy interesante, con mucho potencial. Entiendo que si el usuario pudiera utilizar el enlace directo con chatgpt podrías en una tarde crear una historia bastante interesante para añadir cosas poco a poco a los p`lanetas y generar un entorno coherente. Lo peor ahora mismo puede ser la falta de indicaciones generales sobre las limitaciones en cuando a elementos, info de cada elemento, y demás parámetros que puedas modificar

Lo mejor es que con pocas instrucciones te crea un sistema solar tal y como se lo has descrito. Lo peor, pero es algo bastante lógico, que si quieres salir de las restricciones físicas del sistema solar y ser más creativo, la herramienta no lo tiene soportado.

Parece muy interesante, me hubiese gustado experimentar algo más en una sesión más larga, lo que me ha faltado son algunas opciones de usabilidad como tutoriales para entender la aplicación más por mi cuenta.

Figura 5.15: Descripción de lo mejor y lo peor de la herramienta 1.

Me ha parecido una herramienta muy util, ha funcionado muy bien y ha generado lo que le he pedido de forma correcta con la información que le he metido. Es un poco complicado entender la interfaz una vez se ha generado y han surgido algunos bugs, como que los personajes han aparecido por encima de la información de los planetas. Pero por lo demás ha funcionado correctamente y como esperaba que funcionara, además ha sido sencillo de usar

Es una herramienta muy rápida y muy cómoda, te genera un montón de información de manera muy rápida. Creo que es una buena forma de ligar diseño con desarrollo, permitiendo que los diseñadores puedan generar contenido nuevo cuando quieran y dando una cantidad mínima de trabajo a los desarrolladores.

Lo mejor es lo rápido que me ha generado el primer set de planetas muy rápido con muy poca información, lo peor es que no ha podido generar todos los planetas que quería

La herramienta parece ser muy intuitiva a la hora de su empleo, aunque mi opinión es sesgada pues se me ha explicado como utilizar la herramienta durante su uso. Pero no conlleva muchos pasos para conseguir un resultado bastante decente.

Figura 5.16: Descripción de lo mejor y lo peor de la herramienta 2.

El proceso es muy creativo pues toda la parte de creación física es automática y deja la narrativa a la imaginación del diseñador y tiene la capacidad de rellenar huecos que no especifique el diseñador.

Puntos negativos que he encontrado es el movimiento por el sistema solar porque es un poco limitado, pero es no critico en absoluto. El número de correcciones en mi caso llegó a 5 que puede ser un poco alto, pero no es exagerado y es aceptable en un desarrollo.

Me hubiese gustado que generase automáticamente escenas de juego para cada planeta por defecto, aunque sean de una plantilla, y que se añadan al Build Settings. También me hubiese gustado que contemple la creación de sistemas solares con más de una estrella. Incluso estaría bien poder definir la forma de los cuerpos, pero es meramente estético.

La velocidad a la que genera las cosas aportando una base para el desarrollo muy útil.

Las inconsistencias y las tendencias de la IA, por ejemplo, al generar un sistema optó por generar uno parecido a nuestro sistema solar con los mismos planetas.

Figura 5.17: Descripción de lo mejor y lo peor de la herramienta 3.

Lo mejor ha sido la simulación física, que el sistema se asemeje a un sistema solar real (con limitaciones). Además, el generar un nuevo sistema mediante un input de texto sencillo también facilita mucho las iteraciones. He echado en falta soporte a algunas características como añadir varios soles o algún elemento más de sistemas solares.

Lo mejor de la herramienta es su funcionamiento principal, la generación de los planetas de con las relaciones que sostienen entre ellos, su estado gubernamental y la relación que mantienen entre ellos, y los personajes que crea en relación a esta estructura. Lo peor que encuentro es una falta de leyenda para entender el significado de algunos simbolos e interfaz como tal.

Esta guay que a traves de un texto mio haya generado un sistema solar con sus fisicas y todo.Lo peor es que no es del todo estetico

Está muy bien para juegos en los que la exploración o el worldbuilding no son lo más importante.
Si se desea crear un mundo muy específico, las limitaciones físicas de la herramienta se acaban imponiendo como limitaciones sobre la creatividad de la narrativa.

Figura 5.18: Descripción de lo mejor y lo peor de la herramienta 4.

Con cierta inversión en desarrollo, puede llegar a ser una herramienta útil a la hora de no solo hacer videojuegos, sino demos gráficas y narrativas del género de la ciencia ficción. Sin embargo, la etapa actual del desarrollo de esta herramienta, tiene una serie de restricciones que limitan la narrativa dada por el usuario.

Está genial que puedas aterrizar y generar otra escena, recuerda a Mario Galaxy. Lo mejor yo creo que es que coge tu prompt y te da ideas en las partes que no rellenas, lo que facilita el proceso creativo. Lo único que echo en falta es una leyenda para los iconos.

Mejor: facilidad para crear sobre una base decente desarrollada automaticamente.
Peor: problemas de precisión segun el prompt dado.

Figura 5.19: Descripción de lo mejor y lo peor de la herramienta 5.

Lo mejor de la herramienta es que puede generar muy rápidamente un sistema, dándole una capa de contexto a partir de unas pautas simples. De esa manera puede servir tanto como un esqueleto sobre el que desarrollar la idea, como una inspiración para ver por dónde llevar la historia del mundo generado. En mi caso ha generado bastante bien la idea general a pesar de que faltaba información por el poco tiempo disponible para desarrollarla.

Lo más negativo de la herramienta es que, aunque es esperable, en ocasiones no desarrolla ideas coherentes con lo que se ha planteado. En mi caso ha generado dos habitantes importantes de la misma facción con objetivos comunes, pero los ha clasificado como enemigos entre ellos. Por otro lado, aunque posiblemente se trate de un problema a raíz de la falta de tiempo para desarrollar la idea del mundo, la herramienta no ha indicado qué facción habitaba el tercero de los planetas, aunque si ha indicado que estaba habitado. De esta manera se puede entender que una de las facciones de los otros planetas habita también este último, pero no hay una indicación clara de cual es.

Figura 5.20: Descripción de lo mejor y lo peor de la herramienta 6.

y cuya desviación estándar ha resultado ser 1.21. Los tres motivos más comunes por los que el usuario tiene que hacer este tipo de correcciones son, en orden descendente, que haya habido algún problema con la generación de personajes, que el usuario no esté satisfecho con las razas que ha generado el sistema y que no se haya generado el número de planetas que pedía el usuario.

5.4. Análisis de resultados y alcance de la herramienta

Los resultados mostrados en la figura 5.14 sugieren que la herramienta consigue que la generación del entorno sea más rápida que si se hiciera a mano. En concreto la duración media del experimento han sido de unos veinte minutos, lo que da pie a creer que la herramienta genera el contenido en una sexta parte del tiempo de lo que el 81,3 % de usuarios considera que lo habría hecho. Es importante recalcar que se está comparando con lo que los sujetos consideran que habrían tardado. Hubiese sido idóneo hacer que los usuarios escribieran y crearan el sistema por su cuenta, pero por motivos de presupuesto y disponibilidad de los sujetos no ha sido posible.

La mayoría de los usuarios han encontrado la herramienta útil, como podemos ver en las figuras 5.11 y 5.13, pero vamos a hacer un análisis con los datos y eventos observados durante los experimentos para intentar delimitar el alcance tanto técnico como narrativo de la herramienta.

5.4.1. Alcance técnico

La principal limitación que se ha observado a nivel técnico, y entendemos por nivel técnico la parte de la generación física del sistema, es el número de cuerpos que pida el usario, como sugieren algunos usuarios en la figura 5.15. Aunque el código está pensado para que pueda haber un máximo de cien cuerpos en el sistema, a partir de los quince el LLM empieza a cometer más errores, como nombrar a los planetas de la misma manera o no quedarse corto en la generación, es decir si el

usuario ha pedido veinte planetas solo generar catorce, por ejemplo.

También cabe destacar que a partir de esta cifra la longitud del archivo de entrada empieza a ser un problema, pues el LLM tarda más en generarlo y a veces no lo puede hacer con una sola petición. Por supuesto, una cifra elevada de planetas o satélites implica que es más posible que colisionen entre sí, lo que es un problema tanto a nivel físico como a nivel narrativo.

Que dos o más planetas estén en contacto con sus satélites o entre ellos mismos altera su rotación y su órbita, lo cual puede dar lugar a más colisiones. Además, también puede pasar que un cuerpo acabe dentro del sol, siendo inaccesible para el usuario.

También se presenta como problema la creación de más de una estrella, y aunque ahora mismo el sistema no ofrece al usuario la posibilidad de generar más de una, se han hecho experimentos en estas condiciones y da lugar a que los planetas se salgan de sus órbitas debido a la fuerza que generan las dos estrellas. También hace que los dos soles se muevan, absorbiendo a los planetas que encuentren en su camino y dejando así un sistema inexplorable.

Como conclusión, mencionar que la herramienta permite la creación de un sistema funcional dentro de unos rangos de cuerpos celestiales, siendo el número máximo unos catorce para que en la mayoría de los casos no se produzcan colisiones.

5.4.2. Alcance narrativo

En este apartado vamos a analizar tanto la calidad de la historia y los personajes como la cohesión que tienen entre sí con las razas y facciones que genera el LLM. En cuanto a calidad, cabe mencionar que el LLM tiende a generar historias muy sencillas, con personajes planos y que sulen ser genéricos en las obras de ciencia ficción. A no ser que el usuario lo especifique de otra manera, el LLM normalmente generará como razas las características de las obras de fantasía, como pueden ser los humanos, los elfos y los enanos. En cuanto a las facciones, tiende a no tener en cuenta los sistemas de gobierno de los diferentes planetas, por lo que, por ejemplo, podriamos tener un sistema solar con cuatro planetas, en los que todo son repúblicas, y que hubisen dos facciones, una llamada el imperio y otra la corona, por ejemplo.

Añadido a esto, la historia general del sistema solar tiene a constar de un par de líneas que suelen ser más o menos genéricas, donde normalmente siempre hay dos bandos en guerra y hay uno completamente bueno y otro completamente malo. Los diálogos de los personaje suelen ser muy básicos, carácterísticos de los personajes no jugables en los videojuegos o NPC, por sus siglas en inglés.

Como conclusión destacar que el sistema el capaz de generar una historia creible y que podría ser de utilidad para desarrolladores con poco tiempo o que busquen una inspiración sobre la que luego trabajar, como indican algunos usuarios en la figura 5.20.

Capítulo 6

Discusión

Viendo el desarrollo en retrospectiva podemos afirmar que la herramienta es funcional y puede ser de ayuda en determinados casos. Por supuesto, requiere de un pulido mucho más profundo para llegar a ser útil en un desarrollo real. No obstante, también es cierto que se ha desarrollado una herramienta que puede resultar práctica a algunos desarrolladores, uno de los objetivos principales del proyecto. Además, se ha observado de manera superficial el alcance narrativo que un LLM como Chat GPT es capaz de alcanzar, un apartado que consideramos de especial interés.

Cabe decir que de los objetivos propuestos la comunicación con el LLM es el que menos se ha podido explorar y en el que menos hemos podido profundizar debido a las limitaciones de presupuesto mencionadas en capítulos anteriores, aunque las funcionalidades estén implementeadas, sería necesario depurarlo para conseguir la funcionalidad completa. Aunque hemos podido sortear este obstáculo simulando la funcionalidad que se desearía, es un apartado que no se ha podido explorar al cien por cien como nos hubiese gustado.

A pesar de esto, consideramos que los apartados de la simulación física y la generación narrativa han llegado al alcance esperado y han sido desarrollados en la profundidad que nos gustaría. Aunque hayamos encontrado problemas como puede ser la generación de más de una estrella, cosa que nos hubiese encantado hacer funcionar, el resultado final nos parece satisfactorio a pesar de la falta de algunas ideas como las que se muestran en el apartado del trabajo futuro del capítulo anterior.

Por último, en cuanto a la narrativa, se ha alcanzado todo cuanto se esperaba del LLM. Y como se mencionaba en la introducción, no parece que este tipo de tecnología vaya a dejar desempleados a todos los guionista y escritores del mundo, al menos no a corto plazo. Evidentemente no podemos afirmar nada con seguridad, pero si que se ha observado que las hisotrias que construye una inteligencia artifial tan avanzada como ChatGPT tienden a ser genéricas y carentes de personaliad, propias de una máquina.

Se entiende también que al tener un número de sujetos de prueba reducido y al ser familiares con el entorno y las herramientas de desarrollo no se pueden sacar concluiones definitivas sobre la utilidad de la herramienta. Sería necesario hacer experimentos más extensos y con un número mayor de usuarios antes de afirmar nada.

Parece que los seres humanos tenemos una capacidad casi natural para identificar las cosas que no son humanas, como bien podría decirse de las imagenes generadas por IA, y aunque este tema es digno de estudio en sí mismo, cualquiera que esté medianamente al día con la revolución tecnológica que supone la inteligencia artificial parece ser capaz de identificar una imagen generada por este tipo de tecnología.

Ya sea por la selección de palabras, la estructura abstracta de la historia o algo interno a nuestra naturaleza, las personas parecemos capaces de reconocer los textos y las historias generados mediante inteligencia artificial. ¿Hace esto inútil nuestra herramienta? Creemos que no, y de hecho pensamos que la vuelve más atractiva, ya que en vez de remplazar el trabajo humano, lo facilita y lo ensalza, dando una base desde la que el usuario puede partir o en la que pude buscar inspiración, una cosa realmente difcícil de encontrar cuando se trata de tareas artísticas.

Y aunque algunas empresas pretenden automatizar la producción de cualquier cosa que les permita ahorrarse dinero a cambio de la calidad de sus productos, esta teconología puede no convertirse en la herramienta que le de pie a este tipo de empresas a ejercer despidos masivos de artistas, guionistas y escritores. Si no que puede terminar siendo un catalizador del arte, que facilite a los que lo ejercern a salir de bloqueos creativos, comenzar historias que de otra forma no se hubiesen pasado por su cabeza o agilizar su trabajo, invirtiendo menos tiempo en tareas repititivas y más en darle su personalidad y su esencia a las cosas que realmente lo requieren.

Capítulo 7

Conclusiones, y trabajo futuro

7.1. Conclusiones

Comenzamos este proyecto con la hipótesis de que podriamos conseguir implementar una herramienta que facilitase el trabajo de desarrolladores y diseñadores de videojuegos. El objetivo era claro, comunicar un LLM directamente con Unity de tal manera que el usuario pudiese comunicarse con el LLM directamente desde el editor, obteniendo así una base narrativa desde la que partir, además de un escenario que poder explorar. Tras la implementación del módulo de comunicación con la inteligencia artificial ,el módulo de simulación física y llevar a cabo los experimentos, creemos que el objetivo de darle al usuario un entorno por el que moverse y que explorar está más que cumplido, aunque la parte de comunicación el LLM no se haya explorado tanto como nos hubiese gustado debido a la falta de presupuesto, creemos que con los recursos adecuados y algo más de tiempo para probarlo como es debido, podríamos conseguirlo sin muchas complicaciones.

Por lo tanto, podemos concluir que el desarrollo ha llegado al alcance esperado y que los datos sugieren que hemos desarrollado una herramienta que puede resultar útil para algunos tipos de desarrollos, principalmente para proyectos con un presupuesto más reducido. El apartado de comunicación con el LLM no ha podido ser testeado como nos hubiese gustado. Sin embargo, se ha implementado y con un presupuesto adecuado podría terminar de implementarse en el proyecto sin problemas. El apartado físico y de generación se han conseguido al completo, ya que la simulación funciona correctamente y siempre y cuando el número de cuerpos no supere los rangos mencionados en capítulos anteriores los cuerpos no colisionan entre sí. En cuanto al apartado narrativo se ha conseguido que el LLM genere historias simples con personajes básico que pueden servir como inspiración o como base desde la que trabajar, por lo que damos ese objetivo también como conseguido.

Los datos sugieren que estas afirmaciones son correctas, ya que en su mayoría, los usuarios encuentran útil la herramienta, tanto para desarrollo de videojuegos más cortos como son lo característicos de las game jams, como para un desarrollo más largo. Por supuesto, estos datos no son concluyentes y no podemos afirmar nada ni darlo por zanjado, pues serían necesarios una serie de experimentos más extensos, con más usuarios y con un mayor presupuesto para poder inversitar en profundidad

el alcance de la herramienta.

Cabe mencionar que hay apartados que se podrían mejorar o que directamente se han quedado en el tintero, por esto ahora hablaremos sobre el trabajo futuro del proyecto.

7.2. Trabajo futuro

Con respecto al trabajo futuro vamos a basarnos tanto en lo observado durante los experimentos como las opiniones de algunos usuarios sobre lo que les gustaría que incluyera la herramienta.

- Para empezar, un gran porcentaje de los usuarios pide una leyenda de los iconos utilizados en la interfaz narrativa, por lo que bien se podría implementar una vista en la que el usuario pudiese consultar cada uno de los iconos, como hace Civilization VI(Firaxis Games (2016)) o bien podriamos hacer que al pasar el ratón por encima de uno de los iconos apareciese una caja de texto con la información sobre el icono.

- Otra cosa importante que añadir al generador sería la posibilidad de que las órbitas de los planetas no estén fijas en un solo eje, si no que puedan rotar alrededor de todos los ejes, esto no solo produciría un resultado más estético, sino que es posible que también aumente el número máximo de cuerpos que el sistema soporte sin crear colisiones, ya que sería menos probable que los planetas estuviesen en posiciones cercanas al mismo tiempo.

- Modificar el color de los planetas es una característica que muchos de los usuarios creyeron que sería práctica o interesante durante la realización de los experimentos. Esto podría hacerse añadiendo un parámetro RGB en la clase del planeta, y con ese parámetro luego podriamos modificar los colores del material del planeta. También tendriamos que pedirle al LLM que trajese el color de cada planeta a unos números RGB, lo cual puede dar lugar a las mayores complicaciones. Habría que experimentar para conocer el alcance que puede tener esta opción.

- La posibilidad de tener más de una estrella parece atractiva para la mayoría de usuarios, pero como se ha mencionado anteriormente esto da lugar a una serie de problemas graves que practicamente imposibilitan la utilización de la herramienta. Sin embargo, haciendo que las estrellas no puedan moverse y sumado al segundo apartado de esta lista, conseguiriamos que el número de planetas aumentara todavía más.

- Que las escenas de los planetas se compartan por tipo es algo que habría que modificar cuanto antes para que cada planeta tuviese su escena propia.

- Hacer aparecer a los personajes en cada uno de sus planetas también es una extensión que podría hacerse y que enriquecería enormemente el generador. Se añadiría también la posibilidad de hablar con el personaje e incluso se podrían

implementar la posibilidad de que los personajes diesen misiones al usuario, aunque eso ya implicaría una mayor carga de trabajo.

- Sería interesante probar la herramienta con otros LLMs, para probar con cual de ellos funciona mejor y cual se adapta más a sus características. Aunque hipotetizamos que ninguno funcionaría mejor que ChatGpt, puesto que es el LLM más potente a día de hoy, serían necesarios una serie de experimentos para poder afirmarlo sin temor a equivocarse.
- Realizar experimentos de mayor duración sería prioritario para asegurarnos del correcto funcionamiento de la herramienta, en un caso ideal se realizarían con una población mucho más grande y el tiempo que se les daría a los usuarios para escribir sería de hasta una semana.
- Explorar resultados variando el contenido del prompt sería una idea atractiva, ya que variando el orden en el que se le explican las reglas al LLM, la manera en que se le explican o incluso el idioma de este, sería posible reducir el número de correcciones necesarias para conseguir un resultado satisfactorio, lo cual reduciría todavía más el tiempo necesario para crear algo utilizable por los usuarios.
- Terminar de implementar la comunicación con el LLM sería prioritario en cuanto los recursos lo permitiesen, ya que es de lo poco del proyecto que consideramos incompleto, al menos en lo referente a objetivos.

Introduction

We can find traces of the first instances of storytelling in rock paintings throughout the world. Storytelling is as old as humanity itself, living through Homer, Shakespeare, or Tolkien, to name a few. As we evolve as a species, stories evolve with us, creating worlds that don't exist, telling us stories about our own world, and lighting the path as we travel through the ages. From the Iliad and the Odyssey to The Lord of the Rings, countless authors have spent significant parts of their lives creating stories that move, help, or teach something to the reader. And now, millennia later than the moment the first human rose to draw themselves hunting a deer or a bison, we have at our fingertips, without needing to rise even a little, tools that allow us to create stories with only a definition, a bunch of words, or a handful of indications.

Some time before this technology appeared, a new medium emerged, called the video game, and even though nobody can specify when it was born, nowadays everybody must recognize the importance it has acquired in people's lives. And in a world that is constantly competing for the time of the user, narrative in the shape of books has had to step aside to let new forms of narrative take its place, such as cinema or, for a relatively short period of time, video games.

Although it has not been a long time, not to say it still is this way, since people stopped seeing video games as a waste of time or as simple martian-killer simulators, the medium has proved time after time its potential to carry stories to a whole new level that neither books nor cinema had ever achieved. This is due to its capacity to neither tell us the story through a lens, nor do we know the words of the characters because they are written on some pages. We are the protagonists, we are directly in their flesh, and in some instances, we make their decisions and choose their words.

Titles such as Red Dead Redemption 2 (Rockstar Games (2018)) or Outer Wilds (Annapurna Interactive (2019)) prove that we are in front of a medium capable of teaching us so much more than killing martians, that it has the capacity to move us and teach us things that cannot be taught through study.

Nevertheless, these stories were also written by people of flesh and bone, written in a time when asking a machine to write a story seemed like a product of imagination and science fiction. And now, a handful of years later than the release of these games, we are in front of a technology capable of replicating voices, faces, and programming itself. And, of course, it is also capable of writing stories.

Of course, as happened when the tractor was invented, or when Henry Ford

introduced the first car with an internal combustion engine, there were scared people, excited people, and those most affected of all, those whose fear boiled down to losing their jobs. Artificial intelligence has been no different, as graphic designers, programmers, and music composers panic whenever this technology makes an advancement, and it's not without reason, as what someone could produce in days or weeks of work, algorithms can produce in minutes, if not seconds.

Thus, this work aims to investigate this very issue, but within the realm of narrative, attempting to glimpse at the reach artificial intelligence like ChatGPT might have, and whether eventually every person with a creative job will be supplanted by an algorithm, or perhaps we are witnessing the birth of a tool that will make our lives easier, as it was with the internal combustion engine, or the tractor.

Conclusions and Future Work

7.3. Conclusions

We began this project with the hypothesis that we could implement a tool to facilitate the work of video game developers and designers. The goal was clear: to communicate an LLM directly with Unity so that users could interact with the LLM directly from the editor, thus obtaining a narrative base to start from, as well as an environment to explore. After implementing the communication module with the artificial intelligence, the physics simulation module, and conducting experiments, we believe that the goal of providing users with an environment to navigate and explore has been more than fulfilled. Although the LLM communication aspect was not explored as much as we would have liked due to budget constraints, we believe that with the right resources and a bit more time to test it properly, we could achieve it without many complications.

Therefore, we can conclude that the development has reached the expected scope and that the data suggests we have developed a tool that can be useful for certain types of development, mainly for projects with a smaller budget. The LLM communication aspect could not be tested as thoroughly as we would have liked. However, it has been implemented, and with adequate funding, it could be fully integrated into the project without issues. The physical and generation aspects have been fully achieved since the simulation works correctly and, as long as the number of bodies does not exceed the ranges mentioned in previous chapters, the bodies do not collide with each other.

Regarding the narrative aspect, the LLM has managed to generate simple stories with basic characters that can serve as inspiration or a base to work from, so we consider this objective achieved as well.

The data suggests that these assertions are correct, as most users find the tool useful for both shorter game developments, such as those typical of game jams, and longer developments. Of course, this data is not conclusive, and we cannot make definitive statements, as more extensive experiments with more users and a larger budget would be necessary to thoroughly investigate the tool's potential.

It is worth mentioning that there are areas that could be improved or were left unfinished. Therefore, we will now discuss the future work of the project.

7.4. Future Work

Regarding future work, we will base our approach on both what was observed during the experiments and the opinions of some users regarding what they would like the tool to include.

- Firstly, a large percentage of users request a legend for the icons used in the narrative interface. Therefore, it could be implemented either as a view where the user could consult each of the icons, as Civilization VI does (Firaxis Games (2016)), or we could make it so that when hovering over one of the icons, a text box appears with information about the icon.
- Another important addition to the generator would be the possibility for the orbits of the planets not to be fixed on a single axis but to rotate around all axes. This would not only produce a more aesthetic result but might also increase the maximum number of bodies the system can support without creating collisions, as it would be less likely for the planets to be in close positions at the same time.
- Modifying the color of the planets is a feature that many users believed would be practical or interesting during the experiments. This could be done by adding an RGB parameter in the planet class, and with that parameter, we could modify the colors of the planet material. We would also need to instruct the LLM to bring the color of each planet to RGB numbers, which could lead to some complications. Experimentation would be needed to understand the potential of this option fully.
- The possibility of having more than one star seems appealing to most users, but as mentioned earlier, this leads to a series of serious problems that practically make the tool unusable. However, by making the stars immobile and adding to the second item on this list, we could increase the number of planets even further.
- Ensuring that the scenes of the planets are unique to each would need to be modified promptly so that each planet has its scene.
- Having characters appear on each of their planets is also an extension that could be made and would greatly enrich the generator. It would also add the possibility of interacting with the character, and missions could even be implemented, although that would imply a greater workload.
- It would be interesting to test the tool with other LLMs to see which works best and adapts better to its features. Although we hypothesize that none would work better than ChatGPT, as it is currently the most powerful LLM, a series of experiments would be necessary to confirm this without fear of error.
- Conducting longer experiments would be a priority to ensure the tool functions correctly. Ideally, these experiments would be conducted with a much larger population, and users would be given up to a week to work with the tool.

- Exploring results by varying the prompt content would be an attractive idea. By varying the order in which rules are explained to the LLM, the manner in which they are explained, or even the language used, it might be possible to reduce the number of corrections needed to achieve a satisfactory result, further reducing the time required to create something usable by users.
- Completing the implementation of communication with the LLM would be a priority as soon as resources allow, as this is one of the few aspects of the project we consider incomplete, at least regarding the objectives.

Bibliografía

2K Games. Bioshock. 2007.

Alpaydin, E. *Machine learning*. MIT press, 2021.

Alzubi, J., Nayyar, A. y Kumar, A. Machine learning from theory to algorithms: an overview. En *Journal of physics: conference series*, vol. 1142, página 012012. IOP Publishing, 2018.

Annapurna Interactive. Outer wilds. 2019.

Bethesda. Fallout3. 2008.

Bethesda. Starfield. 2022.

Bi, Q., Goodman, K. E., Kaminsky, J. y Lessler, J. What is machine learning? a primer for the epidemiologist. *American journal of epidemiology*, vol. 188(12), páginas 2222–2239, 2019.

Birhane, A., Kasirzadeh, A., Leslie, D. y Wachter, S. Science in the age of large language models. *Nature Reviews Physics*, vol. 5(5), páginas 277–280, 2023.

Chen, Z. y Liu, B. *Lifelong machine learning*, vol. 1. Springer, 2018.

Cordonnier, J.-B., Loukas, A. y Jaggi, M. Multi-head attention: Collaborate instead of concatenate. *arXiv preprint arXiv:2006.16362*, 2020.

Cunningham, P., Cord, M. y Delany, S. J. Supervised learning. En *Machine learning techniques for multimedia: case studies on organization and retrieval*, páginas 21–49. Springer, 2008.

Dayan, P., Sahani, M. y Deback, G. Unsupervised learning. *The MIT encyclopedia of the cognitive sciences*, páginas 857–859, 1999.

Deng, L., Yu, D. et al. Deep learning: methods and applications. *Foundations and trends® in signal processing*, vol. 7(3–4), páginas 197–387, 2014.

Domsch, S. *Storyplaying: Agency and Narrative in Video Games*. Berghahn Books, 2013.

Edmun McMillen y Florian Himsl. The binding of isaac. 2011.

Egli, A. ChatGPT, GPT-4, and other large language models: The next revolution for clinical microbiology? *Clin. Infect. Dis.*, vol. 77(9), páginas 1322–1328, 2023.

Firat, M. How chat gpt can transform autodidactic experiences and open education? 2023.

Firaxis Games. Civilization vi. 2016.

Fraile-Jurado, P. Geographical aspects of open-world video games. *Games and Culture*, página 15554120231178871, 2023.

Frank Herbert. Dune. 1965.

FromSoftware. Elden ring. 2022.

Gershman, S. J. y Daw, N. D. Reinforcement learning and episodic memory in humans and animals: An integrative framework. *Annu. Rev. Psychol.*, vol. 68(1), páginas 101–128, 2017.

Giray, L. Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, vol. 51(12), páginas 2629–2633, 2023.

Hello Games. No man´s sky. 2016.

Hocking, C. Ludonarrative dissonance in bioshock: the problem of what the game is about. 2009.

Ip, B. Narrative structures in computer and video games: Part 1: Context, definitions, and initial findings. *Games and Culture*, vol. 6(2), páginas 103–134, 2011.

Irie, K., Zeyer, A., Schlüter, R. y Ney, H. Language modeling with deep transformers. *arXiv preprint arXiv:1905.04226*, 2019.

Janiesch, C., Zschech, P. y Heinrich, K. Machine learning and deep learning. *Electronic Markets*, vol. 31(3), páginas 685–695, 2021.

Kaelbling, L. P., Littman, M. L. y Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, vol. 4, páginas 237–285, 1996.

Klei Enterteiment. Don´t starve. 2013.

Learning, S.-S. Semi-supervised learning. *CSZ2006. html*, vol. 5, 2006.

Lewandowski, M., Łukowicz, P., Świetlik, D. y Barańska-Rybak, W. An original study of chatgpt-3.5 and chatgpt-4 dermatological knowledge level based on the dermatology specialty certificate examinations. *Clinical and Experimental Dermatology*, página llad255, 2023.

LucasFilms. Star wars. 1977.

Lucasfilms Games. Zelda ocarina of time. 1990.

Majewski, J. et al. Theorising video game narrative. *Bond University*, 2003.

Matsuo, Y., LeCun, Y., Sahani, M., Precup, D., Silver, D., Sugiyama, M., Uchibe, E. y Morimoto, J. Deep learning, reinforcement learning, and world models. *Neural Netw.*, vol. 152, páginas 267–275, 2022.

Meskó, B. Prompt engineering as an important emerging skill for medical professionals: Tutorial. *J. Med. Internet Res.*, vol. 25, página e50638, 2023.

Mojang. Minecraft. 2009.

Moreno, A., Armengol, E., Béjar Alonso, J., Belanche Muñoz, L. A., Cortés García, C. U., Gavaldà Mestre, R., Gimeno, J. M., Martín Muñoz, M. y Sànchez-Marrè, M. Aprendizaje automático. 1994.

Moritz, S., Romeike, B., Stosch, C. y Tolks, D. Generative AI (gAI) in medical education: Chat-GPT and co. *GMS J. Med. Educ.*, vol. 40(4), página Doc54, 2023.

Naughty Dog. Uncharted. 2007.

Nazir, F., Jawed, S. y Tariq, S. M. Chat GPT and its potential role in medicine. *J. Pak. Med. Assoc.*, vol. 73(12), páginas 2509–2510, 2023.

Nintendo. Zelda ocarina of time. 1998.

On-Line Systems. Mystery house. 1980.

Pavlick, E. Symbols and grounding in large language models. *Philos. Trans. A Math. Phys. Eng. Sci.*, vol. 381(2251), página 20220041, 2023.

Riot Games. Valorant. 2020.

Rockstar Games. Red dead redemption 2. 2018.

Seraphine, F. Ludonarrative dissonance: Is storytelling about reaching harmony? 2016.

Sucker Punch Productions. Ghost of tsunima. 2021.

Sun, C., Huang, S. y Pompili, D. Llm-based multi-agent reinforcement learning: Current and future directions. *arXiv preprint arXiv:2405.11106*, 2024.

Super Giant Games. Hades. 2018.

Supercell. Clash royale. 2016.

System Era Softworks. Astroneer. 2016.

Tarn y Zach Adams. Dwarf fortress. 2006.

Thabet, T. *Video game narrative and criticism: Playing the story*. Springer, 2015.

Ubisoft. Far cry 3. 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. y Polosukhin, I. Attention is all you need. En *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, página 6000–6010. Curran Associates Inc., Red Hook, NY, USA, 2017. ISBN 9781510860964.

Wang, H. y Raj, B. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017.

Wang, L., Chen, X., Deng, X., Wen, H., You, M., Liu, W., Li, Q. y Li, J. Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs. *NPJ Digit. Med.*, vol. 7(1), 2024.

Warren Robinett y Atari. Adventure. 1979.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J. y Schmidt, D. C. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.

Wu, Y., Rabe, M. N., Hutchins, D. y Szegedy, C. Memorizing transformers. *arXiv preprint arXiv:2203.08913*, 2022.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. y Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Ye, Q., Axmed, M., Pryzant, R. y Khani, F. Prompt engineering a prompt engineer. *arXiv preprint arXiv:2311.05661*, 2023.

Yuan, A., Coenen, A., Reif, E. y Ippolito, D. Wordcraft: story writing with large language models. En *27th International Conference on Intelligent User Interfaces*, páginas 841–852. 2022.

Zangrossi, P., Martini, M., Guerrini, F., DE Bonis, P. y Spena, G. Large language model, AI and scientific research: why ChatGPT is only the beginning. *J. Neurosurg. Sci.*, 2024.

Zhang, M. y Li, J. A commentary of gpt-3 in mit technology review 2021. *Fundamental Research*, vol. 1(6), páginas 831–833, 2021.

Zhu, Y., Wang, M., Yin, X., Zhang, J., Meijering, E. y Hu, J. Deep learning in diverse intelligent sensor based systems. *Sensors (Basel)*, vol. 23(1), página 62, 2022.