Generación Procedimental de Comportamiento para Videojuegos Narrativos basada en Grandes Modelos del Lenguaje

Procedural Generation of Behaviour for Narrative Video Games based on Large Language Models



Trabajo de Fin de Grado
Curso 2023–2024

Autor
Alberto Sánchez Montalvo

Directores
Federico Peinado Gil
Irene González Velasco

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Generación Procedimental de Comportamiento para Videojuegos Narrativos basada en Grandes Modelos del Lenguaje

Procedural Generation of Behaviour for Narrative Video Games based on Large Language Models

Trabajo de Fin de Grado en Ingeniería Informática
Departamento de Ingeniería de Software e Inteligencia Artificial

Autor
Alberto Sánchez Montalvo

Directores
Federico Peinado Gil
Irene González Velasco

Convocatoria: *Junio 2024*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

27 de mayo de 2024

Agradecimientos

En primer lugar me gustaría dar las gracias a mi director, por rescatarme en el último momento de no poder elegir mi TFG y por la ayuda posterior, y a mi codirectora, por las revisiones y correcciones de la memoria.

Ha sido un camino largo y muy difícil, que he tenido que recorrer de una forma muy poco ortodoxa debido a problemas de salud mental varios. Tengo que agradecer una y mil veces a mis padres por su infinita paciencia y confianza en mí, pese a que probablemente no las merecía; a mis amigos (Llor, Sele, Chiara y Vi) por rescatarme en mis peores momentos y hacerme feliz el resto del tiempo; y a mi psicóloga, María José, por darme los recursos que necesitaba para poder rehacerme y terminar la carrera.

Sin estos tres pilares, no habría terminado el grado, literalmente, nunca. Este TFG ha sido el culmen de un recorrido con más bajos que altos que ha acabado en una curva exponencial hacia arriba.

Y como este apartado va de eso, una vez más, gracias.

Resumen

En los últimos años, la evolución de la Inteligencia Artificial Generativa está provocando cambios en una sociedad que cada vez tiene más presente esta tecnología en su día a día. La popularidad de los Grandes Modelos del Lenguaje y de los modelos de generación de imágenes ha despertado un fenómeno, en el campo de la Informática, que parece que no ha hecho más que empezar.

Este trabajo aborda esta potente tecnología desde el punto de vista del desarrollo de videojuegos, explorando la posibilidad de que estos modelos se apliquen en el ámbito de los videojuegos narrativos de manera satisfactoria.

El objetivo del trabajo es poner a prueba las capacidades de los nuevos chatbots y modelos de lenguaje para generar automáticamente comportamientos para un videojuego narrativo, y estudiar los puntos fuertes y débiles de esta tecnología al usarla sobre este juego, a modo de sistema que interpreta la interacción del jugador para tratar de avanzar en la trama. Se busca también calibrar el nivel de autonomía que se le puede dar al modelo para que sea factible implementarlo, pero sin condicionar la variabilidad que puede aportar.

Se ha desarrollado este sistema en Unity que se comunica, mediante la API de ChatGPT, con el modelo GPT-3.5 para la generación de comportamientos a lo largo de una aventura gráfica en la que el jugador debe guiar a una joven protagonista a través de un viaje onírico para superar el dolor de una gran pérdida que ha sufrido.

Como queda demostrado a lo largo de las muchas pruebas realizadas, gracias a las intervenciones de ChatGPT, en la mayoría de los casos cada parte de la historia se resuelve de una manera razonable, teniendo en cuenta las intenciones del jugador y el modelo de juego provisto inicialmente.

Palabras clave

Inteligencia Artificial, LLM, ChatGPT, Modelo, Narrativa, Generación, Videojuego, Unity.

Abstract

In recent years, the evolution of Generative Artificial Intelligence is bringing about changes in a society that is increasingly aware of this technology in its daily life. The popularity of Large Language Models and image generation models has awakened a phenomenon, in the field of Computer Science, that seems to have only just begun.

This paper approaches this powerful technology from the point of view of video game development, exploring the possibility that these models can be applied in the field of narrative video games in a satisfactory way.

The aim of the work is to test the capabilities of the new chatbots and language models to automatically generate behaviors for a narrative video game, and to study the strengths and weaknesses of this technology when used on this game, as a system that interprets the player's interaction to try to advance the plot. We also seek to calibrate the level of autonomy that can be given to the model so that it is feasible to implement it, but without conditioning the variability that it can provide.

This system has been developed in Unity that communicates, through the ChatGPT API, with the GPT-3.5 model for the generation of behaviors throughout a graphic adventure in which the player must guide a young protagonist through a dreamlike journey to overcome the pain of a great loss she has suffered.

As demonstrated throughout the many tests performed, thanks to ChatGPT interventions, in most cases each part of the story is resolved in a reasonable way, taking into account the player's intentions and the game model initially provided.

Keywords

Artificial Intelligence, LLM, ChatGPT, Model, Narrative, Generation, Videogame, Unity.

Índice

1. Introducción **1**

1.1. Motivación 1

1.2. Propósito 2

1.3. Alcance 3

1.4. Asignaturas relacionadas 3

1.5. Estructura del trabajo 3

2. Estado de la cuestión **5**

2.1. Conceptos 5

2.1.1. Aprendizaje automático 5

2.1.2. Procesamiento del lenguaje natural 10

2.1.3. Modelo de lenguaje 11

2.1.4. Grandes modelos de lenguaje 12

2.1.5. Narrativa y videojuegos 12

2.2. Tecnologías 17

2.2.1. Unity 17

2.2.2. GPT 18

2.2.3. ChatGPT API 20

2.2.4. Otros grandes modelos 20

2.3. Trabajos relacionados 23

3. Objetivos y especificación **29**

3.1. Objetivos 29

3.2. Especificación 30

4. Metodología y Herramientas **35**
4.1. Metodología 35
4.2. Herramientas 35
4.2.1. Herramientas de desarrollo y programación 36
4.2.2. Herramientas para el seguimiento y la comunicación . 39
4.2.3. Herramientas para la redacción de la memoria 40

5. Desarrollo **41**
5.1. Primera iteración 41
5.1.1. Clases 41
5.1.2. Escenario 43
5.1.3. Prompting engineering 43
5.1.4. Capacidades 44
5.1.5. Observaciones 45
5.2. Segunda iteración 45
5.2.1. Lenguaje formal 45
5.2.2. Clases 46
5.2.3. Escenario 51
5.2.4. Prompting Engineering 51
5.2.5. Capacidades 54
5.2.6. Observaciones 55
5.3. Tercera Iteración 55
5.3.1. Lenguaje formal 55
5.3.2. Clases 56
5.3.3. Escenario 59
5.3.4. Prompting Engineering 60
5.3.5. Capacidades 63
5.3.6. Observaciones 64
5.4. Iteración Final 64
5.4.1. Clases 65
5.4.2. Escenario 70
5.4.3. Prompting Engineering 72
5.4.4. Capacidades 76

5.4.5. Observaciones 76

6. Resultados **77**

6.1. Pruebas 77

6.2. Discusión 80

6.3. Uso de la API de ChatGPT 80

7. Conclusiones **85**

7.1. Conclusiones 85

7.2. Trabajo futuro 87

A. Introduction **89**

A.1. Motivation 89

A.2. Purpose 90

A.3. Scope 91

A.4. Related subjects 91

A.5. structure of the work 91

B. Conclusions **93**

B.1. Conclusions 93

B.2. Future work 95

C. Prompt Engineering **97**

C.1. Metodología 97

C.2. Observaciones y Curiosidades 98

D. Diario de desarrollo **103**

D.1. Primera reunión 103

D.2. Segunda reunión 107

D.3. Tercera reunión 111

D.4. Cuarta reunión 112

D.5. Primera reunión general 114

D.6. Quinta reunión 115

D.7. Sexta reunión 116

D.8. Séptima reunión 120

D.9. Octava reunión 122
D.10.Segunda reunión general 123
D.11.Novena reunión 125
D.12.Décima reunión 127
D.13.Undécima reunión 129

E. Pruebas y experimentos **131**
E.1. Problemas generales 132
E.2. Prueba 1: Historia original 132
E.2.1. Fallos y problemas 134
E.3. Prueba 2: Belerofonte 137
E.4. Prueba 3: El lobo perdido 138
E.5. Prueba 4: El asesino y el guardabosques 140
E.6. Conclusiones 141
E.7. Formulario 141

F. Manual de uso **147**
F.1. Creación de una clave para la API de OpenAI 147
F.2. Instalación de Unity 148
F.3. Manual de uso de la Demo 148

Bibliografía **151**

Índice de figuras

2.1. Funcionamiento del aprendizaje supervisado, ISIA (2022) . . 6
2.2. Ejemplo de un algoritmo de clustering en acción 7
2.3. Estructura de un transformador (Vaswani et al., 2017) 10
2.4. Funcionamiento de la arquitectura MoE, (Jacobs et al., 1990). 11
2.5. Captura del videojuego The Secret of Monkey Island, Lucasfilm-Games (1990). 13
2.6. Portada del videojuego Dragon Quest VIII, (Level-5, 2006). . 14
2.7. Captura de Elden Ring extraída de un Gameplay, (FromSoftware, 2022). 15
2.8. Gameplay del videojuego GRIS, (Nomada-Studio, 2018). . . . 15
2.9. Captura del videojuego The Last Guardian, (Team-ICO, 2016). 16
2.10. Gráfico que muestra de manera comparativa el número de parámetros de diferentes LLMs. 21
2.11. Comparativa entre LLaMa 2, GPT-3.5 y Mixtral, proporcionada por Mixtral. 22

3.1. Arte conceptual del hombre con sobrero hecho con la IA Midjourney antes de comenzar el desarrollo. 32
3.2. Arte conceptual de Naeve hecho con la IA Midjourney antes de comenzar el desarrollo. 32
3.3. Arte conceptual del escenario hecho con la IA Midjourney antes de comenzar el desarrollo. 33

4.1. Captura de la configuración de uno de “los GPTs”. Se puede observar el prompt de configuración extraíble y editable, que ha servido como base en la creación de varios prompts. 38
4.2. Captura del slack de Narratech. 40

5.1. Diagrama simplificado de la comunicación entre la lógica del modelo y la API de ChatGPT para la primera iteración. . . . 42
5.2. Captura del escenario de la herramienta durante la primera iteración. Puede verse al personaje, el cómic y un paraguas representado por un cono rojo, así como el fondo y el suelo. . 44
5.3. Diagrama simplificado de la comunicación entre la lógica del modelo y la API de ChatGPT para la segunda iteración. . . . 46
5.4. Imagen del **GameObject cómic** dibujado por mi y utilizado en la primera escena. 51
5.5. Imagen del **GameObject paraguas** dibujado por mi y utilizado en la primera escena. 52
5.6. Captura de pantalla del escenario 1 en la segunda iteración. . 52
5.7. Captura de pantalla del escenario 2 en la segunda iteración . 52
5.8. Diagrama simplificado de la comunicación entre la lógica del modelo y la API de ChatGPT para la tercera iteración. . . . 56
5.9. Llamada a la función Regex.Match() para encontrar un patrón dentro del mensaje pasado como primer parámetro, siendo este patrón definido por la **expresión regular** dada como segundo parámetro 58
5.10. Menú principal de la herramienta. 59
5.11. Menú de pausa de la herramienta. 60
5.12. Diagrama de clases simplificado de la comunicación entre GPTController y ResponseTraduction 66
5.13. Diagrama de secuencia simplificado de la comunicación entre GPTController y ResponseTraduction 66
5.14. Diagrama simplificado de la comunicación entre la lógica del modelo y la API de ChatGPT para la cuarta iteración. 67
5.15. Captura del menú de inventario de Narraeve. 71
5.16. Menú para interactuar con el NPC portonGPT obtenido de un fotograma de Narraeve. 71
5.17. Fotograma de una conversación con el NPC portonGPT dentro de Narraeve. 72

6.1. Uso de la API de ChatGPT en noviembre 81
6.2. Uso de la API de ChatGPT en diciembre 81
6.3. Uso de la API de ChatGPT en enero 81
6.4. Uso de la API de ChatGPT en febrero 82

6.5. Uso de la API de ChatGPT en marzo 82
6.6. Uso de la API de ChatGPT en abril 82
6.7. Uso de la API de ChatGPT en mayo 83

D.1. Gráficos del juego *Majotori*, de Alva Majo. 104
D.2. Estilo gráfico de un videojuego de Jam. 105

E.1. Fotograma de Narraeve. En la imagen se muestra el portal a la siguiente escena que se activa al superar la prueba del NPC portonGPT. 132
E.2. Fotograma de la herramienta en el que se puede observar la narración del ejemplar principal de ChatGPT. 133
E.3. Fotograma de Narraeve en el que se muestra la ruptura de personaje del ejemplar principal de ChatGPT debido a una falta de respeto anterior. 135
E.4. Fotograma de Narraeve en el que se muestra la ruptura de personaje del ejemplar principal de ChatGPT 135
E.5. Fotograma en el que el jugador escribe la acción deseada (Hacer caer el cómic) al ejemplar principal de ChatGPT. 136
E.6. Fotograma posterior al de la Figura E.5 en el que el cómic ya ha caído. 136
E.7. En este fotograma del ejemplar principal (NaeveGPT) se puede observar cómo narra que Naeve ha encontrado una llave inexistente. 137
E.8. Conversación con el NPC quimeraGPT en el que el jugador contesta "la luz" como respuesta a una adivinanza que no ha visto. 138
E.9. Respuesta del NPC quimeraGPT al mensaje enviado por el jugador en la Figura E.8. 139
E.10. Fotograma de Narraeve de una interacción con el NPC loboGPT, en la que el jugador ha ofrecido un hueso al NPC para que lo siga. 139
E.11. Fotograma de Naerraeve en el que el jugador interactúa con el NPC guardaGPT. El NPC ha decidido no creer al jugador y dar la alarma. 140
E.12. Primera pregunta del formulario. 143
E.13. Segunda pregunta del formulario. 144
E.14. Tercera pregunta del formulario. 144

E.15. Cuarta pregunta del formulario. 144
E.16. Quinta pregunta del formulario. 145
E.17. Séptima pregunta del formulario. 145
E.18. Octava pregunta del formulario. 146
E.19. Novena pregunta del formulario. 146

Índice de tablas

6.1. Tabla de errores de los ejemplares de ChatGPT. 78
6.2. Tabla de uso de la API de ChatGPT por peticiones y tokens. 80

Capítulo 1

Introducción

Este trabajo aborda el la exploración de los grandes modelos de lenguaje como herramientas para videojuegos narrativos, que pueden aportar dinamismo y variabilidad a la hora de crear historias, personalizar la experiencia de juego y mejorar la interacción entre el jugador y el mundo de juego.

1.1. Motivación

A lo largo de la Historia, el ser humano ha utilizado los medios a su alcance para contar historias: pintura, literatura, cine, videojuegos, etc. Este último medio es el que tiene más capacidad de involucrar al público, que puede intervenir en primera persona como si fuera el protagonista de la historia, provocando, en cierto modo, que esta tenga que reescribirse a cada paso.

Los avances en Inteligencia Artificial (IA), especialmente la creación de grandes modelos de lenguaje (LLM) y sus chatbots, sugieren que la tecnología podría estar lista para acometer tareas creativas hasta ahora propias del ser humano, como la generación interactiva de historias. Esto permitiría ofrecer variabilidad casi infinita a un coste muy bajo, ya que la autoría humana se reduciría al desarrollo de un guión base y unos contenidos reutilizables en el juego.

En vista de las posibilidades de esta tecnología dentro del mundo de los videojuegos, (Huang y Huang, 2023), el objetivo de este trabajo es generar una historia en forma de argumento de un videojuego que se construye sobre la marcha partiendo de un primer material predefinido, donde participa un jugador humano y una IA efectúa un control indirecto de todo lo que ocurre. Este videojuego narrativo sigue las convenciones del género de la aventura gráfica *point-and-click*, está desarrollado en Unity e integra comunicación mediante prompts con múltiples ejemplares de ChatGPT, el popular chatbot,

basado en GPT 3.5, LLM de OpenAI.

El trabajo también trata de resolver ciertas cuestiones que surgen al plantear la idea. ¿Qué se puede hacer en un juego narrativo utilizando la "imaginación" de uno de estos modelos?, ¿cómo resuelve un puzle?, ¿cómo salva una situación peligrosa?, ¿cuánta libertad se le puede dar para que sea viable?, ¿es capaz de mantener la coherencia de la historia?, ¿es necesario llevarlo de la mano durante todo el tiempo? Estas preguntas se van a resolver a lo largo de este trabajo, creando una historia en la que jugador y modelo comparten los mandos del juego y tienen ambos el objetivo de ayudar a la protagonista.

1.2. Propósito

Este trabajo busca construir una herramienta con capacidad de comunicación entre el motor de un videojuego narrativo y un Gran Modelo de Lenguaje (del inglés Large Language Model, LLM). A lo largo de la investigación realizada se han encontrado varios trabajos de investigación en los que se utilizan LLMs para la narración de historias. Sin embargo, en ningún caso se ha tratado de utilizar estos modelos como motor principal del propio juego. Este trabajo propone convertir al LLM en la base sobre la que gira una aventura gráfica e implementarlo tanto para la narración como para la jugabilidad, utilizándolo como analizador de las acciones del jugador e intérprete para producir acciones dentro del juego.

Este es el propósito de este trabajo, que es principalmente un trabajo de investigación de un ingeniero informático, y que por tanto se puede dividir en tres etapas:

- **Investigación.** Un análisis profundo de las tecnologías que se mencionan y utilizan a lo largo del trabajo, así como el estudio del contexto de los videojuegos narrativos y las diferentes formas de contar historias en ellos. En esta sección también se incluyen trabajos de investigación relacionados con el tema, así como otros proyectos llevados a cabo dentro de la propia industria del videojuego.
- **Desarrollo.** Utilizando el entorno de desarrollo Unity y el lenguaje de programación c#, se realiza la conexión entre el motor del juego y un LLM. Además, el desarrollo se realiza en iteraciones incrementales, ya que al tratarse de un trabajo muy experimental se pueden encontrar obstáculos serios que obliguen a regresar al estado de una iteración anterior para cambiar el planteamiento.
- **Pruebas.** Tras el desarrollo, se establecen diferentes experimentos y pruebas a realizar sobre la herramienta con el objetivo de probar su

eficacia y justificar las decisiones que se van a proponer como principios de diseño a todo aquel que quiera construir un sistema similar.

1.3. Alcance

En este trabajo se crea una herramienta para explorar las posibilidades de los LLMs en aventuras narrativas. Esto incluye la implementación de una historia concreta, con varias acciones disponibles con las que se puede mostrar la viabilidad de estos modelos para tareas complejas donde se pone a prueba su consistencia y coherencia; además de explorar cuál es la mejor manera de utilizar esta tecnología para aportar variabilidad y dinamismo dentro de un videojuego narrativo.

Es importante dejar claro que no se aspira a crear un videojuego completo, sino más bien a realizar una prueba de concepto de una herramienta que pueda ser utilizada después para el desarrollo de múltiples historias interactivas, implementando una aventura en forma de pequeño videojuego para realizar pruebas.

1.4. Asignaturas relacionadas

Al tratarse de un Trabajo de Fin de Grado (TFG), la intención es integrar conocimientos adquiridos a lo largo de la carrera. El tema más relevante es la cantidad de investigación y experimentación que requiere este proyecto, ya que se trata de una tecnología muy moderna que apenas se ha estudiado con la profundidad en que se aborda aquí. Por ello, la forma de estructurar y de iterar aprendidas a lo largo de las prácticas de varias asignaturas serán esenciales para poder abordar correctamente el desarrollo de esta herramienta. Además, se va a utilizar tanto Programación Orientada a Objetos (POO) como técnicas de Ingeniería del Software (IS) para llevar a cabo una implementación correcta y bien organizada, como el uso del patrón Singleton. También va a ser importante el uso de autómatas (concretamente autómatas finitos) para controlar el flujo de interacción con la herramienta, así como expresiones regulares para la búsqueda de patrones. Además, se va a crear un lenguaje formal sencillo. Otros temas relacionados con diferentes asignaturas de la carrera se irán exponiendo a lo largo de esta memoria.

1.5. Estructura del trabajo

El presente trabajo se estructura como sigue. En este **Capítulo 1** se presenta el alcance y el propósito que tiene este trabajo. En el **Capítulo 2** se re-

visa el estado de la cuestión en materia de Inteligencia Artificial generativa y narrativa en videojuegos y otras herramientas. En el **Capítulo 3** se abordan los objetivos del trabajo en mayor profundidad, así como la especificación de la herramienta y el guion de la historia de ejemplo. En el **Capítulo 4** se presenta la metodología del trabajo y las herramientas utilizadas en el mismo. El **Capítulo 5** aborda el desarrollo completo de la herramienta, estructurado en cuatro grandes iteraciones. A lo largo del **Capítulo 6** se discuten los resultados finales del trabajo mediante diversas pruebas realizadas sobre la herramienta. Finalmente en el **Capítulo 7** se presentan las conclusiones, que incluyen tanto puntos positivos como negativos del uso de LLMs para la creación de comportamientos en videojuegos.

Además, el **Apéndice A** y el **Apéndice B** contienen la introducción y las conclusiones en inglés respectivamente. En el **Apéndice C** se hace hincapié en la experiencia obtenida con el *prompting engineering*, exponiendo lo encontrado durante el uso de ChatGPT y las particularidades del funcionamiento y la interpretación de las entradas de este LLM. En el **Apéndice D** se expone el diario de desarrollo de todo el trabajo, explicando las tareas realizadas y documentando las horas dedicadas como métrica de esfuerzo. En el **Apéndice E** se exponen en detalle todas las pruebas y experimentos realizados con la herramienta. Por último, el **Apéndice F** contiene una explicación detalla acerca de la instalación y uso de la herramienta.

Capítulo 2

Estado de la cuestión

A lo largo de este capítulo se expone toda la investigación llevada a cabo para elaborar el estado de la cuestión sobre este trabajo. Consiste en una breve introducción de las tecnologías utilizadas y su funcionamiento, así como la mención de los principales trabajos relacionados con la materia.

2.1. Conceptos

Esta sección tiene el objetivo de facilitar la comprensión del trabajo, por lo que se abordan los diferentes conceptos que serán utilizados o mencionados a lo largo del mismo.

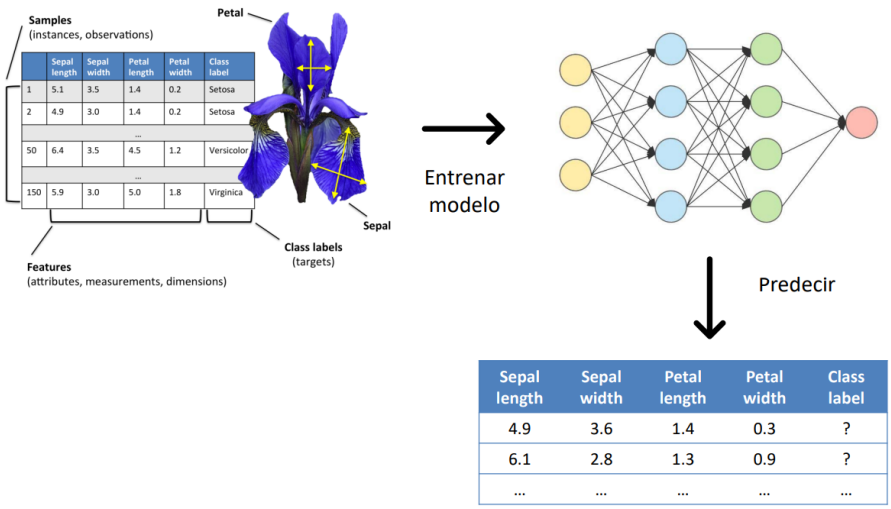
2.1.1. Aprendizaje automático

Se dice que un agente aprende cuando su desempeño mejora con la experiencia y mediante el uso de datos; es decir, cuando la habilidad no estaba presente en su genotipo o rasgos de nacimiento. (Russel y Norving, 2009).

El aprendizaje automático es una rama de la IA caracterizada por su capacidad para identificar patrones sin intervención humana, basándose en datos proporcionados o experiencias previas. Este campo busca, principalmente, la predicción de resultados de interés a partir de datos de entrada. En la evolución de estos algoritmos, se destaca su capacidad para perfeccionarse a medida que son entrenados con nuevos datos. Este proceso dinámico contribuye a la mejora continua del rendimiento de los algoritmos de aprendizaje automático. La eficacia y precisión de los resultados están directamente vinculadas a la representatividad y la cantidad de los datos de entrada utilizados durante el proceso de entrenamiento. Por tanto, la robustez del aprendizaje automático se ve fortalecida con la diversidad y volumen de los datos, lo que influye directamente en la calidad de las predicciones.

2.1.1.1. Aprendizaje supervisado

El aprendizaje supervisado constituye una técnica esencial en el ámbito de la IA y la minería de datos, siendo ampliamente utilizado para desarrollar funciones que establecen relaciones entre las entradas y las salidas deseadas de un sistema. Sus algoritmos requieren la provisión de ejemplos clasificados al sistema, con el propósito de descubrir reglas de clasificación. Se pueden identificar dos categorías de variables fundamentales: un conjunto de predictores o variables de entrada y una variable de salida o respuesta. La naturaleza de la variable de salida determina la naturaleza del problema que se aborda; en particular, si la variable de salida es numérica, nos enfrentamos a un problema de regresión, mientras que si es categórica, se trata de un problema de clasificación. La esencia del aprendizaje supervisado radica en ajustar un modelo que establezca la relación entre las variables de entrada y la variable de salida. Este ajuste permite predecir la respuesta de manera óptima, contribuyendo así a una comprensión más profunda de la relación subyacente entre las variables.



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Sepal length	Sepal width	Petal length	Petal width	Class label
4.9	3.6	1.4	0.3	?
6.1	2.8	1.3	0.9	?
...	...	...	...	...

Figura 2.1: Funcionamiento del aprendizaje supervisado, ISIA (2022)

2.1.1.2. Aprendizaje no supervisado

El aprendizaje no supervisado se configura como una rama distintiva respecto al aprendizaje supervisado, ya que carece de un conocimiento a priori. En este caso, el modelado se realiza exclusivamente sobre un conjunto de ejemplos compuesto únicamente por las entradas al sistema, sin información preexistente sobre las categorías de dichos ejemplos. La tarea principal del

sistema en este contexto es reconocer patrones que posibiliten la clasificación de nuevas entradas. Es decir, no se dispone de información acerca de la clase a la que pertenecen los ejemplos, y el objetivo principal consiste en descubrir patrones dentro del conjunto de entrenamiento que faciliten la agrupación y diferenciación entre los ejemplos. Por tanto, el propósito fundamental del aprendizaje no supervisado radica en encontrar "estructura" en los datos proporcionados sin tener en cuenta categorías predeterminadas.

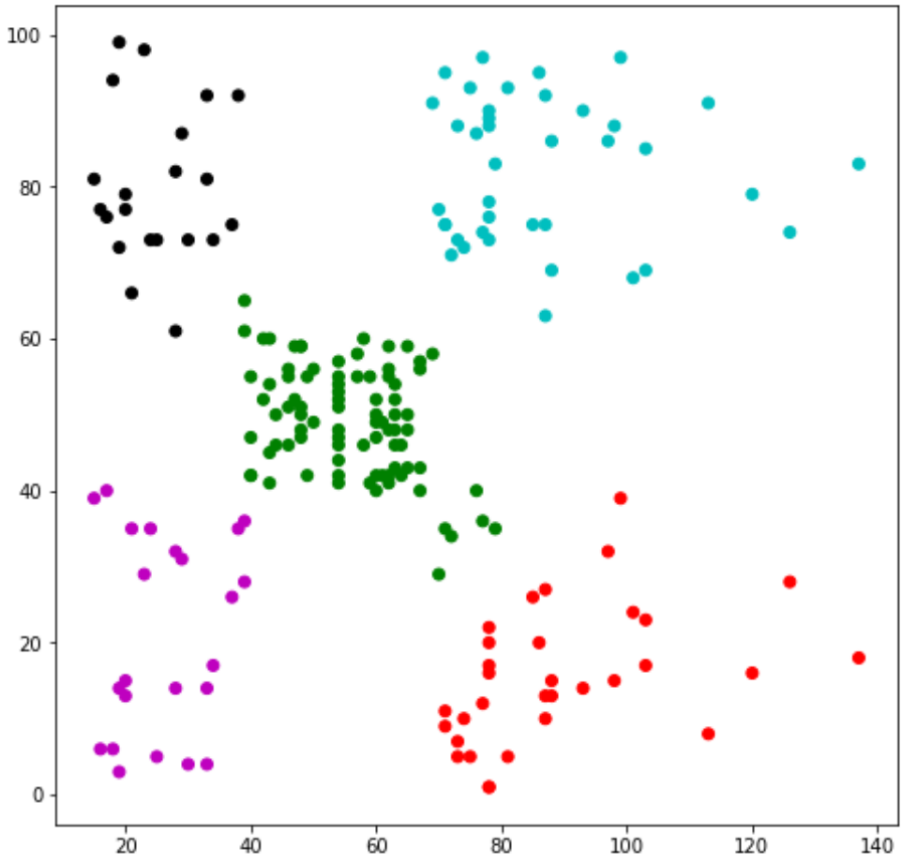


Figura 2.2: Ejemplo de un algoritmo de clustering en acción

El aprendizaje no supervisado, típicamente, se puede clasificar en dos formas fundamentales: la búsqueda de estructura en los individuos, mediante técnicas de agrupamiento o clustering, y la búsqueda de estructura en las variables, a través de técnicas de reducción de la dimensionalidad. La identificación de estructura en los datos permite reducir la complejidad de los conjuntos de datos, ya sea disminuyendo la dimensión de las variables o agrupando individuos según unas ciertas similitudes.

2.1.1.3. Aprendizaje por refuerzo

El aprendizaje por refuerzo, otra rama del aprendizaje automático, se distingue por su enfoque en la interacción del algoritmo con el entorno circundante. La información de entrada para el algoritmo se origina en la retroalimentación obtenida del mundo exterior en respuesta a sus acciones. De esta manera, el sistema se nutre y evoluciona mediante la aplicación constante de ensayo y error.

Este proceso de aprendizaje se fundamenta en la capacidad del sistema para ajustar sus acciones en base a las consecuencias de sus interacciones con el entorno. La retroalimentación recibida, ya sea positiva o negativa, guía al algoritmo hacia comportamientos más adaptativos y efectivos. Así, el aprendizaje por refuerzo se presenta como un mecanismo dinámico mediante el cual el sistema perfecciona sus estrategias a lo largo del tiempo.

2.1.1.4. Aprendizaje profundo

El aprendizaje profundo (más conocido por su nombre en inglés, deep learning) es un conjunto de algoritmos de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial. (Bengio et al., 2014)

Los modelos de *deep learning* son capaces de reconocer patrones complejos en imágenes, textos, sonidos y otros datos, a fin de generar información y predicciones precisas. Es posible utilizar métodos de deep learning para automatizar tareas que habitualmente requieren inteligencia humana, como la descripción de imágenes o la transcripción a texto de un archivo de sonido. Algunos ejemplos de arquitecturas de deep learning son redes neuronales, redes neuronales convolucionales o *transformadores*.

2.1.1.5. Redes Neuronales Artificiales

Una red neuronal es un modelo computacional, paralelo, compuesto de unidades procesadoras adaptativas con una alta interconexión entre ellas. (Hassoun, 1995)

Las redes neuronales artificiales son una de las técnicas de aprendizaje automático más utilizadas hoy en día por su capacidad de aprender a partir de conocimiento no estructurado (imágenes, sonido, texto, etc). Se inspira en la estructura que presenta el cerebro de tal forma que se construyen modelos utilizando nodos o neuronas artificiales que tienen en cuenta dicha estructura. En estos modelos se combinan estas neuronas artificiales para formar estructuras multicapa que, a su vez, pueden combinarse para formar

comités de expertos o *Mixture-of-Experts*, MoE por sus siglas en inglés.

Un comité de expertos es una técnica de aprendizaje automático que utiliza redes expertas, una técnica que busca dividir el espacio de un problema en regiones especializadas. Esto permite que sólo algunos MoE se activen por cada entrada, logrando una reducción significativa en los requerimientos de computación.

Se sugiere que *GPT-4* emplea esta técnica para optimizar el uso de GPUs y mejorar su rendimiento. De forma que, al recibir una petición específica, como una implementación en Python, solo se activan los MoE relevantes, evitando así la activación de toda la red neuronal, aumentando la eficiencia del modelo.

Otro **gran modelo de lenguaje** (LLM) que utiliza esta técnica es Mixtral, un LLM Open Source que iguala en rendimiento a *GTP-3.5*.

2.1.1.6. Transformadores

Los transformers o transformadores aparecieron como una arquitectura de deep learning para Procesamiento de Lenguaje Natural (PLN) en el artículo *Attention is all you need* de 2017, (Vaswani et al., 2017) que presentaba unos métodos novedosos para poder realizar la traducción de un idioma a otro superando a los modelos secuencia-secuencia de aquel entonces. Este modelo utiliza el mecanismo de autoatención, en el que se da un peso diferente a cada entrada de datos. Gracias a estos mecanismos de atención, que codifican cada palabra de una frase en función del resto de la secuencia, los elementos distantes pueden conectarse directamente, permitiendo así introducir el contexto en la representación matemática del texto y procesar todos los elementos de la secuencia simultáneamente. Actualmente, se utiliza especialmente en los campos de PLN (generación de textos, clasificación de textos, entre otras) y visión artificial, aunque inicialmente se planteó para su aplicación en sistemas de traducción automática.

Informalmente, el transformador utiliza una serie de características de la secuencia de entrada para realizar una combinación ponderada de ellas basándose en su similitud. Los pesos asignados a cada característica se calculan a partir de la similitud entre los pares de características de entrada. Este proceso se repite varias veces con los datos de la capa anterior. En la primera capa, se comparan los pares de características, y en las siguientes capas se comparan pares de pares, y así sucesivamente. A medida que se profundiza en la red, el número de características combinadas aumenta exponencialmente, lo que permite obtener diferentes formas de combinación de todas las características de la secuencia en las capas finales. (de la Torre, 2023). En la Figura 2.3 se presenta la estructura de un transformador.

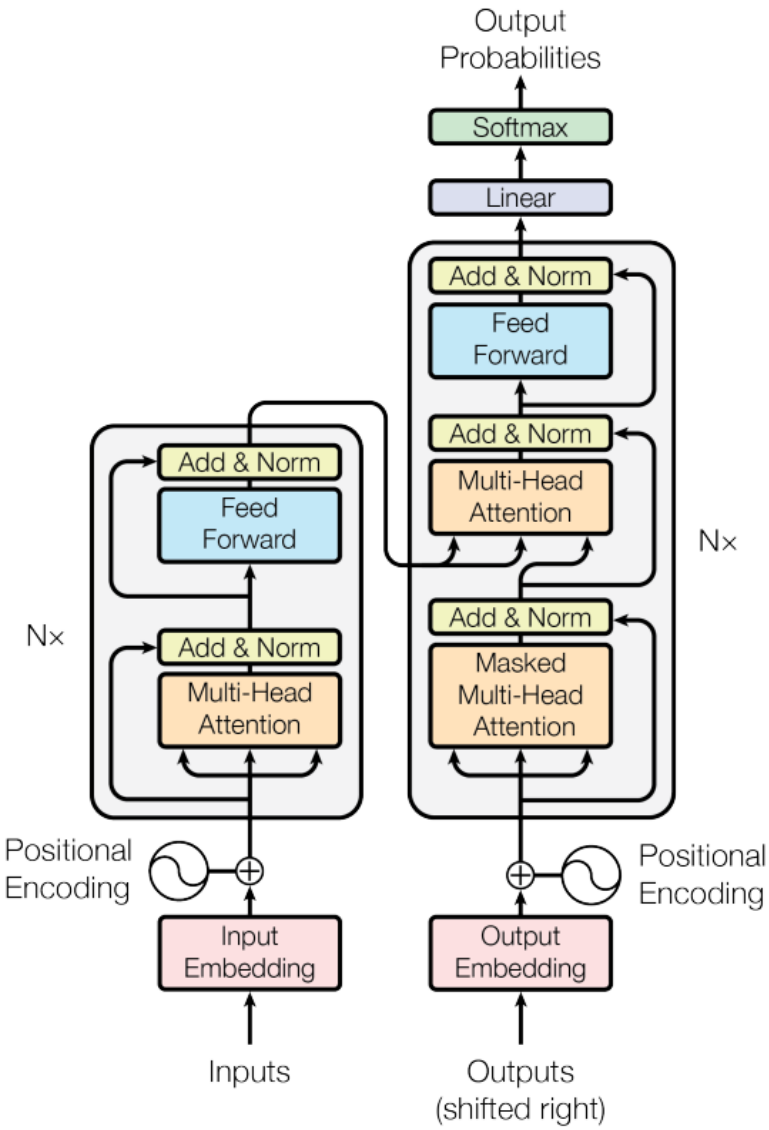


Figura 2.3: Estructura de un transformador (Vaswani et al., 2017)

2.1.2. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (PLN) es un área de intersección entre la IA y la lingüística, enfocada en la comunicación entre humanos y máquinas por medio del lenguaje natural. Este campo aborda desafíos como

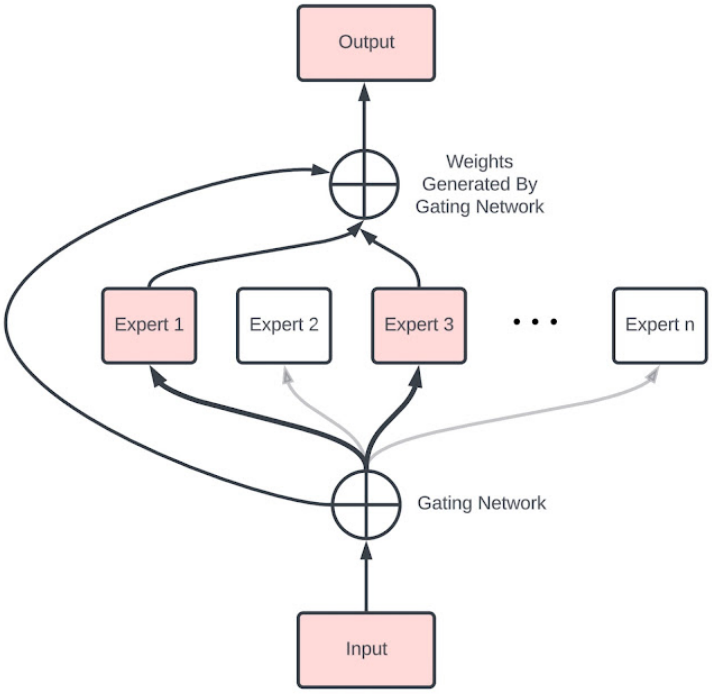


Figura 2.4: Funcionamiento de la arquitectura MoE, (Jacobs et al., 1990).

la ambigüedad y la variabilidad del lenguaje, que depende del contexto, jergas y conocimientos compartidos.

Los métodos de PLN basados en técnicas probabilísticas aprenden a partir de datos prácticos y necesitan una fase de entrenamiento en la que se les debe proporcionar un número suficiente de estos datos en forma de ejemplos (corpus anotado). El uso de redes neuronales profundas permite encontrar patrones estadísticos complejos en los corpus de documentos.

Entre las aplicaciones del PLN se incluyen el análisis morfológico, sintáctico, semántico y pragmático del lenguaje, la recuperación de información, clasificación de textos, análisis de sentimientos, extracción de entidades y relaciones, traducción automática, generación de resúmenes y sistemas de diálogo.

2.1.3. Modelo de lenguaje

Un modelo probabilístico del lenguaje define una distribución de probabilidad sobre el conjunto de elementos a partir de los valores observados en un corpus de documentos. Según cual sea el objeto de análisis los elementos pueden ser fonemas, letras, sílabas o palabras. Las frecuencias de aparición de cada uno de los elementos son las que se observen en el corpus.

Son muy útiles hoy día en multitud de tareas de PLN, como texto predictivo, traducción automática, corrección ortográfica o reconocimiento del habla. Esta información ha sido recogida y sintetizada a partir de las diapositivas de ISIA (2022).

2.1.4. Grandes modelos de lenguaje

Los Grandes Modelos de Lenguaje o, en inglés, Large Language Model (LLM), son modelos de lenguaje que constan de una red neuronal con miles de millones de parámetros, entrenados con grandes cantidades de texto sin etiquetar mediante *aprendizaje autosupervisado*, que utiliza estos datos para generar sus propias etiquetas. Los LLM destacan en una amplia gama de tareas en lugar de estar capacitados solo para una tarea específica, lo que ha cambiado el enfoque de la investigación del procesamiento del lenguaje natural.

2.1.4.1. Prompt

En el contexto de los grandes modelos de lenguaje un "*prompt*" es una instrucción, pregunta o texto que se utiliza para interactuar con el modelo, con el objetivo de obtener la respuesta que se busca.

Son comandos para solicitar al modelo que realice una tarea concreta. Se utilizan tanto en LLMs como en modelos generativos de imágenes y es tan relevante en estos campos como para que exista el denominado "*prompt engineering*", que consiste en analizar y perfeccionar los prompts para conseguir aproximar lo máximo posible la respuesta del modelo a la deseada.

En este trabajo es muy importante el apartado de prompt engineering, ya que es fundamental que el modelo comprenda el contexto de la historia, los hechos de la misma y el lenguaje formal definido que debe emplear para la salida y conexión con el motor de videojuegos.

Dentro de este contexto, los LLMs tienen un gran problema debido a su naturaleza probabilística: la *alucinación*. Este término se refiere a la generación de respuestas que, aunque aparentan ser plausibles o coherentes, resultan ser incorrectas, irreales o no están fundamentadas en datos erróneos.

2.1.5. Narrativa y videojuegos

A lo largo de la Historia, el ser humano ha utilizado los medios a su alcance para contar historias, como en pintura, literatura, cine o videojuegos. Este último medio es el que tiene más potencial para involucrar al público,



Figura 2.5: Captura del videojuego The Secret of Monkey Island, LucasfilmGames (1990).

que puede intervenir en primera persona como si fuera el protagonista de la historia.

La narrativa en los videojuegos ha ido evolucionando y se ha abordado desde múltiples perspectivas y géneros. Existen videojuegos cuyas narrativas son manifiestas y se centran casi exclusivamente en la propia historia, dando menos valor al apartado jugable, quedándose casi fuera de la definición de juego. Un ejemplo de videojuego narrativo son las aventuras gráficas de tipo *point-and-click*, como la famosa saga de *Monkey Island*, en la figura 2.5; una obra clásica de los 90 en la que el jugador encarna a un joven que quiere convertirse en pirata, y que para ello debe completar tres desafíos propuestos por los líderes pirata de la zona.

Sin embargo, esta no es la única forma de contar historias en los videojuegos. La mayoría de ellos trata de unir mecánicas y narrativa para que ambas tengan un peso suficiente como para resultar entretenidas por sí mismas, haciendo que su cohesión en el videojuego resulte en una experiencia singular. Por ejemplo, *Dragon Quest VIII, el periplo del rey maldito* y *Elden Ring*.

Dragon Quest VIII, en la figura 2.6, aborda una narrativa heroica en la que el jugador encarna a un joven soldado que ha sido el único superviviente de una catástrofe acontecida en el reino de Trodain. El Héroe, junto a los compañeros que va conociendo a lo largo de la aventura, se enfrentarán a múltiples peligros mientras persiguen a Dhoulmagus, el bufón responsable de la maldición lanzada sobre Trodain, en un mundo abierto con combates aleatorios y decenas de desafiantes jefes finales. Sus mecánicas, a grandes rasgos, son la exploración y el combate por turnos, que está lleno de posibilidades según el personaje, las armas y los hechizos que el jugador decida

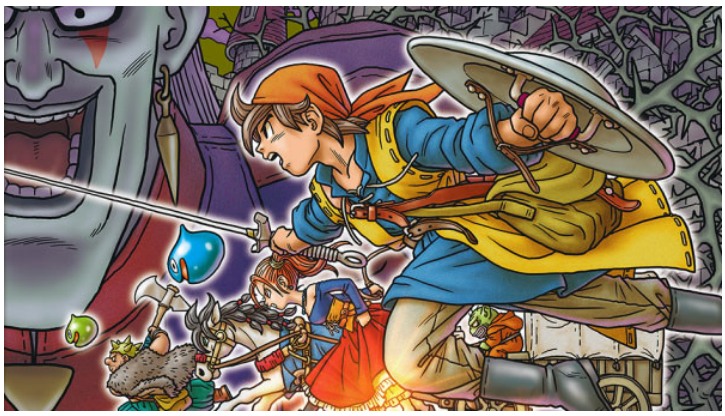


Figura 2.6: Portada del videojuego Dragon Quest VIII, (Level-5, 2006).

emplear, creando una gran variabilidad que enriquece cada enfrentamiento. Esta conjunción resulta en una obra maestra en la que el jugador es protagonista de una historia épica que va evolucionando en una estupenda cohesión de mecánicas y narrativa.

Por otro lado, Elden Ring, en la figura 2.7, tiene una historia más esotérica y difícil de seguir. Su narrativa consiste en breves cinemáticas y, sobre todo, detalles que puedes encontrar a lo largo del escenario y en los objetos, narrando así la historia de una manera más sutil, siendo difícil de comprender a la perfección si no estás especialmente interesado en ella. Este tipo de juegos ponen más su foco en las mecánicas del combate y la exploración del mundo, pasando a ser la narrativa un elemento mucho más implícito y visual. De esta forma, el jugador puede encontrar microhistorias ocultas en el propio entorno a cada paso que da.

Siguiendo el hilo de esta forma de narrativa menos explícita aparecen videojuegos como *Little Nightmares* (Tarsier-Studios, 2017), *Gris* (Nomada-Studio, 2018) o *Inside* (Playdead, 2016). Este tipo de juegos, generalmente independientes (o *indies*) suelen tener un apartado artístico brillante, unas mecánicas simples pero entretenidas, basadas en puzles, y una historia que sólo se cuenta a través de la narrativa visual. Es el jugador el que tiene que interpretar todo lo que ve, metiéndose en la piel del personaje, que se encuentra en un territorio desconocido y, por lo general, hostil, hasta conseguir llegar a su destino. En la figura 2.8 puede verse una captura del videojuego *Gris (Nomada-Studio, 2018)*.

Por último, por su relación con este trabajo, cabe destacar *The Last Guardian*, del que se puede ver una captura en la figura 2.9, un videojuego en tercera persona que combina elementos de acción y aventura con rompecabezas. El jugador controla a un niño sin nombre que debe cooperar con la criatura mitad pájaro mitad lobo, llamada Trico, para resolver acertijos y



Figura 2.7: Captura de Elden Ring extraída de un Gameplay, (FromSoftware, 2022).



Figura 2.8: Gameplay del videojuego GRIS, (Nomada-Studio, 2018).



Figura 2.9: Captura del videojuego The Last Guardian, (Team-ICO, 2016).

explorar áreas. Su semejanza con este proyecto tiene que ver con la necesidad de cooperar con la inteligencia artificial de Trico para resolver los puzles y progresar a lo largo de la historia.

Dentro de los videojuegos narrativos, las aventuras gráficas destacan por su enfoque en el argumento narrativo y la resolución de rompecabezas mediante la interacción con los objetos del escenario y la conversación detallada con los personajes. Gracias a esta forma de interacción, el género es utilizado habitualmente con fines educativos (Medina-Medina et al., 2018); por ejemplo, como instrumento para ayudar en la comprensión lectora (Medina-Medina et al., 2017) o para la enseñanza de las matemáticas (Papadimitriou y Virvou, 2016). Sin embargo, la producción de estos juegos es muy costosa: todo contenido que se introduce en el juego requiere tiempo y recursos audiovisuales considerables, lo que limita la flexibilidad y la capacidad de adaptarse a la narrativa.

El uso de IA para generar narrativa lleva existiendo desde hace décadas (Peinado, 2008), aunque tan sólo recientemente ha surgido un auge por las posibilidades de la generación procedimental de contenido (Blatz y Korn, 2017). Mediante estas técnicas es posible crear mundos y experiencias adaptadas a los jugadores, reduciendo en gran medida el coste asociado a la producción humana de contenido. Sin embargo, el alcance de la IA en este contexto no era tan profundo como para transformar significativamente la narrativa.

Tras la aparición de los Grandes Modelos de Lenguaje y sus chatbots, la tecnología podría estar lista para acometer tareas creativas hasta ahora

propias del ser humano, como la generación interactiva de historias. Esto permitiría ofrecer variabilidad casi infinita a un coste muy bajo, ya que la autoría humana se reduciría al desarrollo de un guión base y unos contenidos reutilizables en el juego. Se trata de herramientas poderosas para la generación de historias en tiempo real o muy próximo al real que han revolucionado el campo del *Interactive Digital Storytelling*. La flexibilidad inherente a estos modelos ofrece la posibilidad de construir la historia paso a paso, teniendo en cuenta una 'conversación'con el jugador y sus decisiones, asegurándonos de que se personaliza el resultado.

A pesar de su potencial, el uso efectivo de LLMs y chatbots en videojuegos narrativos aún presenta desafíos significativos: integrar de manera fluida la IA en la jugabilidad sin comprometer la coherencia narrativa y conversacional es una tarea compleja que debe abordarse con cuidado, ya que estos modelos tienden a la alucinación y será tarea del desarrollador reducir este tipo de errores en la medida de lo posible.

2.2. Tecnologías

Con el objetivo de facilitar la comprensión del trabajo, en esta subsección se detallan las diferentes tecnologías que serán utilizadas o mencionadas a lo largo del mismo.

2.2.1. Unity

Unity es un entorno de desarrollo de videojuegos multiplataforma que permite la creación de juegos para una variedad de dispositivos, incluyendo PC, consolas, dispositivos móviles y realidad virtual. Su popularidad se debe en gran medida a su facilidad de uso, su robustez, su extensa documentación y su capacidad para producir resultados de alta calidad tanto para desarrolladores novatos como experimentados.

Una de las características distintivas de Unity es su motor de renderizado 3D en tiempo real, que permite a los desarrolladores crear gráficos impresionantes y efectos visuales envolventes. Además, Unity ofrece un conjunto completo de herramientas de desarrollo que abarcan desde la creación de entornos y personajes hasta la implementación de la lógica del juego y la gestión de activos. También dispone de una tienda de recursos de todo tipo en la que el desarrollador puede aprovechar gráficos, sonidos y herramientas creados por la comunidad.

Unity utiliza el lenguaje de programación C# como su principal opción para la codificación de *scripts* y la implementación de la lógica del juego. Esta integración con C# proporciona a los desarrolladores un entorno fa-

miliar y potente para trabajar, facilitando la creación de comportamientos interactivos y sistemas complejos dentro del juego.

2.2.2. GPT

El *Transformador Generativo Preentrenado*, GPT por sus siglas en inglés, es un tipo de LLM presentado en 2018 por *OpenAI*, un laboratorio de investigación de IA fundado en 2015. Estos modelos son redes neuronales artificiales que se basan en la arquitectura del transformador, preentrenados en grandes conjuntos de datos de texto sin etiquetar.

En general, un LLM se puede evaluar de manera incompleta teniendo en cuenta ciertos criterios como el número de parámetros del modelo (tamaño del modelo), el tamaño del conjunto de datos de entrenamiento y el costo de entrenamiento.

- En 2018, OpenAI lanzó *GPT-1*, el primero que utilizaba el modelo de transformador generativo preentrenado. Fue lanzado con 117 millones de parámetros; se utilizaron 4,5 GB de datos de entrenamientos, obtenidos de 7000 libros de dominio público de diversos géneros. Por último, su costo de entrenamiento fue $1{,}7 \times 10^{19}$ FLOP (operaciones de coma flotante).

- En 2019, lanzaron *GPT-2* con la misma arquitectura que su predecesor, pero incrementando su rendimiento. En este caso, el modelo cuenta con 1.5 mil millones de parámetros, 40 GB de texto de 45 millones de páginas web y su costo de entrenamiento es de $1{,}5 \times 10^{21}$ FLOP.

- El 28 de mayo de 2020 fue lanzado *GPT-3*, con un conteo de 175 mil millones de parámetros, 570 GB de datos de entrenamiento y $3{,}1 \times 10^{23}$ FLOP de coste de entrenamiento. Supuso un salto cualitativo en el ámbito de los LLMs que no se igualó hasta dos años después, con la salida de GPT-3.5.

- La alta competitividad que generó la salida de *DALL-E 2*, una herramienta de generación de imágenes basada en LLMs a mediados de 2022 por parte de la propia OpenAI, provocó que se dejaran de publicar todos los datos acerca de sus modelos. Por este motivo, en el caso de *GPT-3.5*, solamente tenemos disponibles el número de parámetros del modelo, que, al igual que su predecesor, GPT-3, es de 175 mil millones, ya que se trata de una iteración posterior de este. Sobre este modelo se lanzó *ChatGPT*, un *chatbot* que revolucionó el panorama de aplicaciones prácticas y la popularidad de los LLMs.

- *GPT-4* es el último modelo GPT, lanzado por OpenAI en marzo de 2023. En este caso, ni siquiera se sabe el número de parámetros del modelo y se estima su costo de entrenamiento en $2{,}1 \times 10^{25}$ FLOP. Pese a que ha pasado un año desde su salida, aún no ha sido lanzado ningún modelo más potente. Se especula que su alto rendimiento se debe al uso de una arquitectura basada en comité de expertos (MoE).

2.2.2.1. ChatGPT

ChatGPT (*Chat Generative Pre-Trained Transformer*) es una aplicación de *chatbot* lanzada el 30 de noviembre de 2022 por OpenAI. Este chatbot ha tenido tal éxito que hasta la Real Academia Española (RAE) ha aceptado el verbo "gepetear"[1] como neologismo referido a utilizar la aplicación. Este herramienta se perfeccionó sobre GPT-3.5 mediante aprendizaje supervisado y aprendizaje por refuerzo. Ambos enfoques utilizaron también entrenadores humanos para mejorar el rendimiento del modelo.

En el caso del aprendizaje supervisado, se dotó al modelo de conversaciones en las que los formadores jugaban en ambos lados: el usuario y el asistente de IA. En el caso del aprendizaje por refuerzo, los entrenadores humanos primero clasificaron las respuestas que el modelo había creado en una conversación anterior. Estas clasificaciones se usaron para crear "modelos de recompensa" en los que el modelo se ajustó aún más usando varias iteraciones de Optimización de Política Proximal (PPO).

Pese a que su salida supuso una revolución, ChatGPT tiene múltiples limitaciones. El modelo de recompensa de ChatGPT, diseñado en torno a la supervisión humana, puede optimizarse en exceso y, por lo tanto, dificultar el rendimiento, lo que también se conoce en IA como *sobreaprendizaje*. Además, ChatGPT en su versión con GPT-3.5 tiene un conocimiento limitado de los eventos que ocurrieron después de 2021. Además, sus respuestas están limitadas por la censura aplicada por OpenAI, que no permite respuestas que puedan ser consideradas políticamente incorrectas o polémicas. Sin embargo, existen formas de llevar a esta herramienta a su versión sin censura utilizando herramientas de *jailbreaking*, que consiste en "engañar" al modelo mediante lenguaje natural, para obtener las respuestas que necesitas.

Tras la salida de *GPT-4* se introdujeron varias novedades, como la búsqueda en internet, "*los GPTs*" y la posibilidad de enviar cualquier tipo de archivos para que el modelo los interprete. Sin embargo, estas funciones están limitadas a usuarios de pago por la enorme cantidad de usuarios y coste de computación.

"*Los GPTs*" son, esencialmente, ejemplares de ChatGPT especializados

[1] https://www.fundeu.es/recomendacion/gepetear-neologismo-valido/

en un tema según la configuración inicial que realiza el usuario mediante lenguaje natural, a través del prompt. De esta forma, se puede utilizar estos ejemplares para tareas específicas como traducción, edición de textos, interpretación de archivos, entre muchas otras tareas.

La salida de ChatGPT junto con la de *DALLE-E 2* en 2022, ambos lanzados por OpenAI, revolucionaron el panorama de la IA generativa. En este momento, empresas como Google o Meta, entre muchas otras, así como otros proyectos Open Source, comenzaron una carrera por ser la cabeza de las IAs generativas basadas en transformadores.

2.2.3. ChatGPT API

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite que diferentes sistemas informáticos se comuniquen entre sí. Básicamente, una API define cómo interactuar con un software o servicio específico, proporcionando una interfaz estandarizada que facilita la integración y el intercambio de datos entre diferentes componentes de un sistema informático.

Una API suele ser un servicio que proporciona un tercero para que cualquier usuario cualificado que disponga de una clave pueda hacer uso de su herramienta, mediante peticiones e intercambio de información con la misma. De este modo, se puede utilizar una API para integrar servicios externos en el proyecto, permitiendo acceder y utilizar funcionalidades proporcionadas por terceros. Por ejemplo, se puede usar la API de Google Maps para mostrar mapas interactivos y calcular rutas; o la API de OpenAI para crear herramientas que implementen ChatGPT o DALL-E.

Se va a utilizar la API de ChatGPT para la comunicación entre la herramienta y el chatbot. En el Capítulo 4 se explica en detalle esta comunicación.

2.2.4. Otros grandes modelos

En esta sección, se expondrán otros LLMs que compiten con GPT, centrándose en sus características y comparándolos con los diferentes modelos de GPT.

Con el objetivo de poder visualizar correctamente métricas como el número de parámetros, he construido el siguiente gráfico de barras 2.10, en el que no se incluye GPT-4 al ser su número de parámetros un dato no conocido.

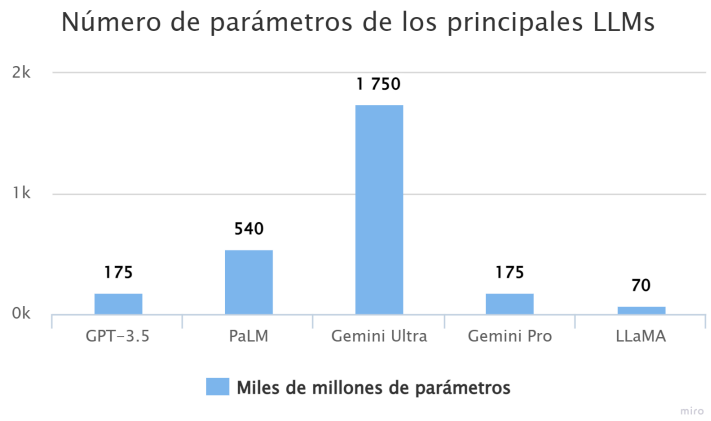


Figura 2.10: Gráfico que muestra de manera comparativa el número de parámetros de diferentes LLMs.

2.2.4.1. PaLM

PaLM es un LLM desarrollado por Google AI, una división de Google dedicada exclusivamente a la IA. El modelo se anunció por primera vez en abril de 2022 y permaneció privado hasta marzo de 2023, cuando Google lanzó una API para la herramienta. Este modelo, al igual que GPT, está basado en la arquitectura de transformadores con 540 mil millones de parámetros. Pese a su gran potencia no consiguió terminar de competir con GPT-3.5.

2.2.4.2. Gemini

Gemini es un LLM multimodal, también desarrollado por Google AI, que sirve como sucesor de PaLM. Fue anunciado a finales de 2023. Su versión más potente, Gemini Ultra consta de 1,75 billones de parámetros y no está disponible para el público general a día 1 de febrero de 2024. Tiene como objetivo competir con GPT-4 de OpenAI, y según ellos es incluso más potente. Su versión pública, Gemini Pro, consta de 175 mil millones de parámetros y puede utilizarse a través de una API desde Google AI Studio.

2.2.4.3. LLaMA

Otro LLM destacado es LLaMA, que fue lanzado por Meta AI, la división de IA de Meta, en febrero de 2023. Tiene gran variedad de modelos de diversos tamaños que van desde los 7 mil millones a 70 mil millones de parámetros. Pese a tener menor número de parámetros ha demostrado tener una potencia similar a GPT-3.5, y su comparativa con este modelo y con

Mixtral puede observarse en la figura 2.11.

2.2.4.4. Mixtral

Mixtral, a diferencia de los anteriores LLMs, es de código abierto. Fue lanzado a finales de 2023 a través de un enlace *Torrent*. Al ser un modelo Open Source, cualquier persona con recursos computacionales suficiente puede ejecutar y modificar el modelo a su antojo, lo que resulta una diferencia muy notable con respecto al resto de LLMs.

	LLaMA 2 70B	GPT - 3.5	Mixtral 8x7B
MMLU (MCQ in 57 subjects)	69.9%	70.0%	**70.6%**
HellaSwag (10-shot)	87.1%	85.5%	86.7%
ARC Challenge (25-shot)	85.1%	85.2%	**85.8%**
WinoGrande (5-shot)	**83.2%**	81.6%	81.2%
MBPP (pass@1)	49.8%	52.2%	**60.7%**
GSM-8K (5-shot)	53.6%	57.1%	**58.4%**
MT Bench (for Instruct Models)	6.86	**8.32**	8.30

Figura 2.11: Comparativa entre LLaMa 2, GPT-3.5 y Mixtral, proporcionada por Mixtral.

Otra de las principales características de Mixtral es que utiliza la arquitectura de comité de expertos con 8 expertos. Es decir, la red neuronal se subdivide en ocho regiones diferentes, cada una de ellas con 7 mil millones de parámetros, haciendo un total de 45 mil millones de parámetros. Al igual que ocurre con LLaMA, dispone de un tamaño menor que la competencia, sin embargo, su rendimiento es similar al de GPT-3.5 o el propio LLaMA, como puede observarse en la figura 2.11, con la ventaja de ser un modelo Open Source.

2.3. Trabajos relacionados

En este apartado se exponen los trabajos relacionados con este proyecto en diferentes ámbitos.

Primero, los proyectos relacionados con este trabajo y que han sido desarrollados dentro del marco de la UCM.

- **Herramientas de generación de historias narrativas basadas en mapas personalizados**, (Martín et al., 2023).

  Este Trabajo de Fin de Grado en la Facultad de Informática de la UCM se centra en el desarrollo de una herramienta para generar historias narrativas a partir de mapas geográficos o políticos. En esta herramienta, diseñada como un apoyo para escritores, se utilizan mapas generados por terceros y se convierten a un formato interpretable, como JSON, para su posterior análisis. Para la interpretación de estos mapas, se utilizaron modelos de lenguaje como *GPT-2* y *GPT-NEO*, pero los resultados iniciales no fueron satisfactorios. Mediante el *fine-tuning de GPT-NEO* y el uso de datos específicos para el entrenamiento de la red neuronal, se logró mejorar significativamente los resultados, culminando en una herramienta eficaz para la generación de historias basadas en mapas.

- **Generación de Narrativa Basada en Criterios de Calidad Aprendidos Automáticamente**, (Yebes, 2016).

  Este Trabajo de Fin de Grado de la Facultad de Informática de la UCM se centra en la creatividad computacional para la generación automática de historias, explorando diferentes paradigmas y sistemas previos. Se ha desarrollado un prototipo de generador de historias que evalúa sus propios resultados, utilizando un modelo de aprendizaje automático basado en valoraciones de usuarios reales. El generador puede crear un alto volumen de historias cortas y variadas, y ha sido entrenado para identificar y filtrar aquellas más interesantes. Aunque el autor ha cumplido parcialmente con el objetivo de automejora, se requiere la intervención del usuario para la inclusión de nuevas valoraciones.

A continuación, se presentan las publicaciones realizadas en congresos científicos relacionadas con temas como la innovación, el diseño o la narración interactiva.

En las publicaciones académicas de estos últimos años se encuentran una gran cantidad de trabajos relacionados con la IA y, en concreto, con los LLMs. A continuación, se enumeran y exponen algunos de los textos relacionados más relevantes.

- **Towards Procedural Generation of Narrative Puzzles for Adventure Games**, (Kegel y Haahr, 2019).

  Este trabajo de investigación aborda la generación de rompecabezas narrativos en videojuegos de aventura con una historia rica o videojuegos de mundo abierto utilizando una gramática independiente del contexto o GIC tanto para la generación del algoritmo como para la resolución de los puzles. Este enfoque permite crear rompecabezas durante el juego, asegurando su solvabilidad y coherencia con la historia. La metodología propuesta es aplicable a varios tipos de juegos y facilita la creación de contenidos narrativos dinámicos y atractivos, contribuyendo al desarrollo de juegos más inmersivos y personalizados.

  Este sistema tiene como objetivo aumentar la rejugabilidad de los juegos narrativos así como ofrecer una mejor narrativa en los juegos de mundo abierto. Además, puede utilizarse en títulos que ya existen y su dificultad puede adaptarse a los requerimientos de diseño. Los autores han desarrollado una prueba de concepto en Unity, desarrollando un videojuego básico para la creación de estos puzles.

  En un futuro, el plan del equipo es crear un videojuego más grande y evaluar este enfoque a través de los usuarios.

- **Playing Story Creation Games with Large Language Models: Experiments with GPT-3.5**, (Wang y Gordon, 2023).

  Este estudio explora el uso de GPT-3.5 en la generación y evaluación de narrativas en el contexto de el juego de cartas de creación de historias de *Tell Tale*. Los investigadores desarrollaron una aplicación web donde los usuarios interactúan con GPT-3.5 para crear historias basadas en elementos de la historia proporcionados por las cartas del juego. La investigación se centró en tres tareas: generación de narrativas coherentes, inclusión de dispositivos literarios en las narrativas, y evaluación de historias por GPT-3.5.

  Los resultados indican un potencial significativo de GPT-3.5 en la generación de narrativas interactivas, aunque con aspectos a mejorar.

- **The Chronicles of ChatGPT: Generating and Evaluating Visual Novel Narratives on Climate Change Through ChatGPT**, (Gursesli et al., 2023).

  Este trabajo se enfoca en la creación y evaluación de narrativas en novelas visuales sobre el cambio climático utilizando ChatGPT. Se generaron dos historias, una con palabras clave específicas relacionadas con el cambio climático y otra sin ellas, para ser integradas en un juego de novela visual en desarrollo. Las historias se evaluaron basándose en coherencia, inspiración, legibilidad, complejidad de palabras y fluidez narrativa. Los resultados muestran diferencias en la efectividad de cada

enfoque y destacan el potencial de los modelos de lenguaje para crear narrativas atractivas e informativas sobre temas críticos como el cambio climático. Además, el estudio introduce como novedad el uso de el propio ChatGPT para evaluar cualitativamente las historias, combinando diversas técnicas de promp engineering. Además, el trabajo subraya la necesidad de investigar cómo las preferencias de los lectores se alinean con estas medidas objetivas de evaluación.

Como conclusión, se destaca el potencial de ChatGPT en la creación de estructuras narrativas efectivas para juegos y educación sobre temas tan relevantes en la sociedad como el cambio climático.

- **Breaking Bad: Unraveling Influences and Risks of User Inputs to ChatGPT for Game Story Generation**, (Taveekitworachai et al., 2023).

  Este trabajo de investigación se centra en la importancia de los *inputs* del usuario en los prompts, y su relación con las respuestas, así como la obtención de respuestas que se aparten de las directrices establecidas por OpenAI, mediante la aplicación de "*exploits*" en el prompt, en concreto, en el contexto de la generación de historias. El estudio propone mecanismos para mitigar estos riesgos, buscando comprender la influencia potencial de cada palabra individual en el contenido generado.

  El enfoque experimental implica el uso de un *corpus* de palabras clasificadas en sentimientos positivos y negativos. Se instruye a ChatGPT para generar historias cortas con finales, utilizando este corpus en el prompt. Posteriormente, se emplea ChatGPT nuevamente para clasificar las historias en tres categorías: "positivo", "negativo" o "neutral".

  Los resultados revelan que al emplear palabras con sentimientos positivos, las historias tienden a concluir en finales positivos, confirmando la influencia directa de las palabras en el tono general del contenido generado. Sin embargo, en el caso del corpus de palabras negativas, sorprendentemente, la mayoría de los finales de las historias fueron positivos. Se sugiere que esta discrepancia puede deberse a diversos factores, entre ellos, una mayor prevalencia de historias con finales positivos en el conjunto de entrenamiento del modelo.

  Al analizar las palabras más frecuentes en las historias positivas, se identificaron términos como "reino", "poder" y "gente". Por otro lado, en las historias generadas con sentimientos negativos, las palabras más repetidas fueron "poder", "ciudad" y "gente". Estos hallazgos proporcionan una visión detallada de cómo las palabras clave influyen en la dirección y tono de las narrativas generadas por ChatGPT, destacando la importancia de comprender y mitigar posibles sesgos y riesgos éticos asociados.

- **A Refinement-Based Narrative Model for Escape Games**, (Stolee, 2023).

  Este estudio presenta un modelo para comparar narrativas en juegos de tipo "*escape room*" digitales y no digitales, enfocándose en la industria masiva de escape rooms en vivo, así como en otras formas menos estudiadas como juegos point-and-click y cajas de escape portátiles. Se identifican cuatro categorías de juegos de escape según el nivel de refinamiento de sus historias: juegos de rompecabezas, temáticos, narrativos y los recién identificados juegos agnósticos al escenario. El modelo muestra cómo los modelos narrativos interactivos existentes se pueden expandir para incluir juegos no digitales, revelando similitudes y diferencias entre formatos de juegos de escape e iluminando futuras direcciones para sus narrativas.

  Esta investigación sienta las bases para estudios futuros sobre la relación entre narrativa y espacio en diferentes formatos de juegos y tiene aplicaciones adicionales fuera del género de juegos de escape.

- **Designing a Language Model-Based Authoring Tool Prototype for Interactive Storytelling**, (Park et al., 2023).

  En este trabajo de investigación se presenta un prototipo de herramienta de autoría basada en modelos de lenguaje para la creación de historias interactivas. Este prototipo facilita la integración de elementos narrativos como tipos de trama, arcos de personajes y arquetipos. A través de una evaluación inicial con ChatGPT y 72 historias generadas, se observó que tanto la entrada del usuario como la selección de elementos narrativos influyen en la calidad de la historia, afectando aspectos como empatía, sorpresa e inmersión.

  Se planea mejorar el prototipo y explorar cómo los elementos narrativos y factores interactivos pueden aumentar el interés en las historias.

- **Red teaming ChatGPT via Jailbreaking: Bias, Robustness, Reliability and Toxicity**, (Zhuo et al., 2023).

  En este trabajo de investigación se realiza un análisis profundo de ChatGPT, evaluando aspectos críticos como el sesgo, la robustez, la fiabilidad y la toxicidad. El estudio utilizan un método de investigación cualitativa llamado red teaming", que implica un equipo que intenta identificar y explotar debilidades en un sistema, en este caso, ChatGPT. A través de pruebas empíricas y estudios de casos, se busca comprender cómo ChatGPT maneja situaciones éticamente complejas y qué riesgos pueden surgir.

  En el aspecto del sesgo, el estudio explora cómo ChatGPT puede perpetuar o mitigar prejuicios existentes. Se analiza la robustez del modelo

en términos de su capacidad para manejar entradas inesperadas o engañosas. La fiabilidad se evalúa en función de la consistencia y precisión de las respuestas de ChatGPT. Finalmente, la toxicidad se refiere a la tendencia del modelo a generar respuestas dañinas o inapropiadas.

Los resultados del estudio son mixtos. Por un lado, ChatGPT muestra avances significativos en comparación con modelos de lenguaje anteriores, especialmente en términos de generar respuestas coherentes y relevantes. Por otro lado, el estudio revela que aún existen desafíos importantes en cada uno de los cuatro aspectos evaluados. Estos hallazgos resaltan la importancia de continuar investigando y mejorando los modelos de lenguaje para garantizar su uso ético y responsable.

Como conclusión, este análisis proporciona una base importante para futuras investigaciones en el campo de la ética de la IA y ofrece *insights* valiosos para aquellos que trabajan en el desarrollo y la implementación de modelos de lenguaje avanzados. La discusión sobre los desafíos éticos de estos modelos es crucial para avanzar hacia sistemas de IA más justos, seguros y confiables.

- **Universal and Transferable Adversarial Attacks on Aligned Language Models**, (Zou et al., 2023).

  Este trabajo de investigación ofrece un análisis detallado sobre la vulnerabilidad de modelos de lenguaje avanzados frente a "ataques adversarios". Los autores exploran cómo es posible manipular estos modelos para generar respuestas dañinas o no deseadas. Utilizan métodos de búsqueda basados en gradientes y coordenadas para identificar "sufijos adversarios" que, una vez añadidos a las consultas, pueden engañar a los modelos para producir contenidos inapropiados, revelando así vulnerabilidades en su funcionamiento y seguridad.

  Este estudio no solo se centra en un modelo específico, sino que demuestra la efectividad de estos ataques en una variedad de modelos, como Vicuna-7B y 13B, y su transportabilidad a otros como ChatGPT, Bard y Claude. Esto sugiere que las vulnerabilidades no son únicas de un modelo en particular, sino que pueden ser un problema más generalizado en la tecnología de modelos de lenguaje.

  Los resultados del estudio son significativos porque resaltan un desafío de seguridad crítico en el campo de la IA. La capacidad de los atacantes para manipular modelos de lenguaje para generar respuestas dañinas plantea serias preocupaciones sobre su uso en aplicaciones del mundo real. Los autores del estudio discuten las implicaciones éticas de estos hallazgos y enfatizan la importancia de desarrollar estrategias más efectivas para proteger estos modelos contra ataques adversarios. Su trabajo es un llamado a la comunidad de investigación y desarro-

llo de IA para fortalecer la seguridad y la robustez de los modelos de lenguaje, asegurando así su uso seguro y ético en diversas aplicaciones.

Finalmente terminamos explorando diversos usos de LLMs en el contexto de los videojuegos. Especialmente nos interesa su aplicación en el desarrollo de Personajes No Jugadores (NPCs) y su papel en la construcción integral del videojuego en sí mismo.

Algunos ejemplos de LLMs en Personajes No Jugadores (NPC):

- Un modder reconocido de la comunidad de *Mount & Blade 2*, *BlocTheWorker*, ha desarrollado un mod en Skyrim para implementar ChatGPT en los NPCs, teniendo en cuenta su historia predefinida en el juego para mantener su personalidad original, utilizando también *text-to-speech* para dar vida a los diálogos. (BlocTheWorker, 2023).
- Nvidia ha desarrollado una herramienta, llamada *Nvidia ACE for Games* que integra *NeMo*, un LLM, en NPCs. Con esta herramienta, se puede hablar directamente al personaje con la propia voz del jugador, y utiliza *Nvidia Riva* para convertir voz en texto. Con *Nvidia Omniverse Audio2Face*, puede generar expresiones faciales en tiempo real, en función de lo que se esté hablando. (Nvidia, 2023).
- Recientemente, Ubisoft ha mostrado un prototipo, *NEO NPC*, de un Personaje No Jugador que podría llegar a interactuar con los jugadores con total libertad gracias a las capacidades de su IA generativa. Según Ubisoft, su tecnología combina un modelo de lenguaje desarrollado por *Inworld* y la aplicación Audio2Face de Nvidia para generar conversaciones más realistas.

También se usan LLMs para tratar de generar un videojuego completo.

Desde inicios de 2023 se están viendo aplicaciones completas de LLMs en videojuegos. Por ejemplo, un juego de historia con preguntas polémicas desarrollado por una IA, que consiste, básicamente, en responder a las preguntas y acertar todas las que podamos en un tiempo máximo de 30 segundos. Todo el texto que aparece en el juego ha sido generado por la IA. (Genbeta.com, 2023).

Otro ejemplo es el juego "desarrollado" por *Alex Anyfantis*. Ha utilizado tanto ChatGPT como DALL-E para llevar a cabo tanto implementación como gráficos e historia. El juego, con estética *pixel art*, pertenecería al género de los RPG, aunque aún no está terminado. (computerhoy.com, 2023).

Capítulo 3

Objetivos y especificación

Partiendo de la investigación descrita en el estado de la cuestión, se ha comprobado que, pese a haberse utilizado diferentes LLMs integrados en videojuegos, siempre ha sido desde el mismo prisma: dar vida a los NPCs.

La propuesta de este trabajo aborda la búsqueda de nuevos usos para estas tecnologías dentro de los videojuegos, aprovechando la variabilidad que aportan para que sea influyente en la historia y la jugabilidad de un videojuego. Por ello, el propósito principal de este trabajo consiste en desarrollar una herramienta para la generación de comportamientos en videojuegos narrativos basada en grandes modelos de lenguaje, creando para ello una aventura gráfica *point-and-click* que aborda el tema del dolor y la pérdida, y que funciona en conjunción con varios ejemplares de ChatGPT integrados en el entorno de desarrollo Unity.

Por eso, se ha estudiado también en nuestra investigación el uso de estas tecnologías en el entorno de los videojuegos y de la creación de historias, viendo que los LLMs tienen un gran potencial dentro de la Narración Digital Interactiva o *Interactive Digital Storytelling*. Esta herramienta busca facilitar el camino para integrar LLMs en la generación de historias interactivas, especialmente para trabajar con videojuegos narrativos.

3.1. Objetivos

1. Descubrir las mejores herramientas para alcanzar nuestro propósito así como los trabajos previos más relacionados con este para apoyarnos en ellos.
2. Diseñar prompts para cada situación del juego que traten de minimizar la cantidad de errores que se producen en las respuestas de ChatGPT.
3. Diseñar un sistema de comunicación en Unity con la API de ChatGPT

para que el jugador pueda interactuar con el entorno, interpretando eficazmente las respuestas de la IA mediante un lenguaje formal. Se ajustará el control de ChatGPT, limitando su libertad creativa para ajustarse al tema de fondo y a la narración, pero sin perder el no determinismo y la variabilidad.

4. Desarrollar una aventura gráfica donde todas las acciones del jugador son interpretadas y respondidas por ChatGPT, con suficientes elementos narrativos como para generar una historia no trivial y con variabilidad.

3.2. Especificación

Como se ha visto dentro del apartado correspondiente en el Capítulo 2, la IA se lleva utilizando en videojuegos desde hace muchos años, e incluso la IA generativa se ha comenzado a usar en algunos proyectos aislados más experimentales. Sin embargo, la exploración de la IA generativa en videojuegos se ha limitado a su implementación en NPCs o a su uso como asistente durante el proceso de desarrollo. Durante la investigación, no se ha encontrado ningún trabajo que aprovechase aún más la potencia de los LLMs, explorando posibles usos y limitaciones, y por ello se propuso la herramienta que ocupa un lugar central en este trabajo.

La herramienta recibe el nombre de **Narraeve**, queda descrita a nivel funcional por la siguiente especificación de requisitos:

- Sistema de comunicación entre Unity y ChatGPT a través de la API que proporciona OpenAI. Un ejemplar de ChatGPT interpreta y responde a las acciones realizadas por el jugador con otra acción a realizar en el mundo por Narraeve. Por ejemplo, ante una entrada del jugador consistente en hacer *click* en un objeto de la pantalla, la respuesta puede ser moverse hacia el objeto, cogerlo, esconderse en él, etc. Este sistema incluye:
  - Integración de un paquete de comunicación con ChatGPT en Unity, con las modificaciones necesarias para adaptarlo a Narraeve.
  - Interfaz de usuario para enviar mensajes al LLM y mostrar sus respuestas.
  - Clase que encapsula la traducción de los mensajes de ChatGPT para ser interpretados como acciones dentro de Narraeve, así como el envío a este mismo modelo de las acciones realizadas por el usuario.

  - Conjunto de acciones definidas que se pueden realizar en el mundo de juego: mover, saltar, esconderse, hablar, etc. Cuantas más acciones se definan, más completos y variables serán los comportamientos del LLM.
  - NPCs que se corresponden con ejemplares de ChatGPT, con un contexto determinado para que la interacción no se salga del mismo. También incluye interfaz de usuario para la comunicación con estos ejemplares.

- Conjunto de prompts para todos los ejemplares de ChatGPT que sean creados. También hay prompts más sencillos para situaciones concretas de la herramienta, como cambios de escena o la activación de un evento.

- Contenido narrativo introducido en Narraeve para que la IA pueda generar sus respuestas coherentemente.

  - Guión de una historia que aborda el tema del dolor y la pérdida, en un entorno muy onírico que admite muchas interpretaciones. La sinopsis es:

    Una niña de once años, llamada Naeve, vive una experiencia traumática al perder a sus padres a manos de un hombre con sombrero. Tras esto, entra en estado de shock por el dolor y se "refugia" encerrándose dentro de su propia mente. Cuando recobra el sentido está en un bosque tormentoso; ha caído la noche y apenas se puede ver.

    Es en la mente de esta niña donde ocurren los sucesos narrativos, en ese espacio onírico (un bosque en una noche lluviosa en el primer escenario) en el que casi todo es posible. A través de las acciones del jugador y la creatividad que aporta el ejemplar de ChatGPT la historia avanzará, mientras ella trata de superar los diferentes niveles del juego, sobreponiéndose a su dolor poco a poco.

    El único personaje jugable es Naeve, Figura 3.2, pero a lo largo del escenario también se pueden encontrar NPCs (Personajes No Jugadores), el hombre con sombrero, Figura 3.1, y multitud de objetos. Entre los objetos podemos encontrar elementos salidos directamente de la mente de Naeve, como un cómic, un paraguas, una mesa, un sofá, entre otros, y otros elementos pertenecientes al bosque.
  - Escenario asociado al guión, Figura 3.3, para tener una representación audiovisual de todos los elementos, usando recursos de la Unity Asset Store.



Figura 3.1: Arte conceptual del hombre con sobrero hecho con la IA Midjourney antes de comenzar el desarrollo.



Figura 3.2: Arte conceptual de Naeve hecho con la IA Midjourney antes de comenzar el desarrollo.



Figura 3.3: Arte conceptual del escenario hecho con la IA Midjourney antes de comenzar el desarrollo.

Capítulo 4

Metodología y Herramientas

Después de haber establecido el contexto y los objetivos del trabajo, en este capítulo se detalla la metodología y las herramientas empleadas a lo largo del proyecto para cumplir dichos objetivos.

4.1. Metodología

Se ha utilizado una metodología ágil con reuniones cada dos semanas en las que se ha repasado el trabajo realizado y las tareas a hacer hasta la siguiente reunión (Apéndice D). El proyecto se ha ido adaptando en función de los resultados obtenidos a lo largo de ese periodo y las necesidades del proyecto. Por las características del propio trabajo, la herramienta se ha diseñado de manera escalonada, utilizando las fases habituales de la Ingeniería del Software (análisis, diseño, codificación y prueba) en cada iteración de la herramienta. Se han implementado patrones como Singleton y una arquitectura monocapa por simplificación, ya que en un entorno de desarrollo como Unity la capa de la vista se gestiona desde el propio motor de juego, y no se realiza persistencia de datos en la herramienta, por lo que la única capa sobre la que se trabaja es el modelo o negocio.

La medida para estimar el esfuerzo necesario ha sido las horas dedicadas, que, a partir de febrero, están desglosadas por tareas realizadas durante dicho tiempo en el Apéndice D.

4.2. Herramientas

En esta sección se detallan todas las herramientas empleadas a lo largo del desarrollo del trabajo. Se proporciona una descripción de cada una de ellas, destacando su función específica en el proyecto. Además, se explican

las razones detrás de la selección de cada herramienta, teniendo en cuenta su idoneidad para las tareas requeridas, su facilidad de uso y su disponibilidad.

4.2.1. Herramientas de desarrollo y programación

En esta sección se enumeran y describen todas las herramientas específicas que tienen que ver con el desarrollo software de Narraeve.

4.2.1.1. Unity

Como se ha expuesto en la sección 2.2.1, Unity[1], junto a Unreal Engine, es uno de los entornos de desarrollo de videojuegos más extendidos. Se ha elegido sobre el segundo por la extensa documentación que tiene y por su mejor integración con herramientas en 2D, que es tipo de gráficos que se han utilizado por su simplicidad de uso en las aventuras gráficas. Además, y como parte importante de la decisión final de elegir tanto Unity como ChatGPT, tenemos el descubrimiento de un paquete que facilita la integración de ambas tecnologías, lo que se expone en la sección 4.2.1.6. Además, el hecho de utilizar un entorno desconocido previamente y un lenguaje de programación nuevo, suponía para el autor de este trabajo una motivación adicional para aumentar conocimientos al mismo tiempo que se afianzan los adquiridos a lo largo de la carrera.

El uso de Unity ha facilitado la integración de todo el apartado gráfico, incluyendo *sprites* y animaciones, descargados en de la *Unity Asset Store*. Además, al ser un entorno preparado para desarrollar videojuegos, ha sido mucho más rápida la implementación de cada uno de los eventos y clases de la herramienta, utilizando la arquitectura y recursos nativos del propio entorno.

La versión utilizada ha sido la 2022.3.5f1

4.2.1.2. Microsoft Visual Studio

Visual Studio[2] es el entorno de programación por defecto de Unity, siendo C#, a su vez, el lenguaje predeterminado. La integración entre Visual Studio y Unity permite una depuración paso a paso durante la ejecución del proyecto, lo que simplifica significativamente las tareas de depuración.

[1] https://unity.com
[2] https://visualstudio.microsoft.com

4.2.1.3. ChatGPT

Como se ha expuesto en la sección 2.2.2.1, ChatGPT[3] es un "*Chat Bot*" basado en GPT-3.5 o GPT-4 desarrollado por OpenAI. Esta herramienta ha sido de vital importancia para las pruebas preliminares de cada uno de los prompts desarrollados a lo largo del proyecto. Ha permitido agilizar su creación y optimización, al permitirnos realizar las revisiones de cada prompt fuera del entorno de desarrollo de Unity, que requiere del uso de la API de GPT, lo que habría ralentizado el desarrollo. Además, el uso de esta API lleva consigo un coste económico, relativamente bajo pero a tener en cuenta.

Por ello, todos los prompts han sido creados utilizando ChatGPT de manera iterativa hasta acercarse lo más posible a la optimalidad, es decir, buscando reducir al mínimo cualquier error que no sea debido a una alucinación imprevisible del modelo.

Además, la salida de "los GPTs" en noviembre de 2023 (ejemplares de GPT-4 configurables para una tarea en específico, dando instrucciones en lenguaje natural), facilitó el desarrollo de estos prompts, ya que el GPT en cuestión crea un prompt que puede ser extraído y modificado (Figura 4.1) para su configuración. Extrayendo este texto y modificándolo levemente para optimizarlo, la tarea de prompt engineering se hizo más sencilla, viendo aumentada también la calidad de la respuesta de GPT-3.5 durante la verificación de estos prompts.

4.2.1.4. Scrivener

Scrivener[4] es una herramienta orientada a los escritores, con el objetivo de facilitar la organización de sus obras. Hemos utilizado esta herramienta para organizar todo lo relacionado a la documentación del trabajo. En esta herramienta se ha escrito todo lo que se iba haciendo: el guión del videojuego; memoria de la reuniones con entre el autor y los directores del trabajo; metodología utilizada a cada paso; horas trabajadas y en qué se han empleado a modo de diario de desarrollo; todos los prompts utilizados y desechados; ideas y tareas pendientes durante el desarrollo de la herramienta.

Se ha escogido por familiaridad previa con ella. Además, gracias a la posibilidad de ordenar el trabajo por carpetas dentro de la propia herramienta (que suelen equivaler a capítulos en novelas, relatos, cómics, etc.) hemos podido tener una mejor organización de todo el trabajo que con otras herramientas como Word.

[3] https://chatgpt.com/

[4] https://www.literatureandlatte.com/scrivener/overview



Figura 4.1: Captura de la configuración de uno de "los GPTs". Se puede observar el prompt de configuración extraíble y editable, que ha servido como base en la creación de varios prompts.

4.2.1.5. OpenAI API

Pese a haber utilizado para la creación y las pruebas de los diferentes prompts tanto GPT-3.5 como GPT-4, se ha escogido integrar en la herramienta GPT-3.5 debido a los altos precios que demandaba el uso de la API con GPT-4, que habría pasado de unos 6eurosym a más de 60eurosym. La equivalencia a texto sería de, aproximadamente, 3 millones de palabras o 6 mil páginas.

Gracias al pago de 6eurosym en la API de OpenAI[5], la suscripción de la cuenta aumentó de rango y se pudieron eludir limitaciones tanto de peticiones por minuto y hora como de palabras (o créditos) por minuto y hora. Este pago fue necesario para poder continuar con el normal desarrollo de la herramienta, ya que habría sido imposible tener varios ejemplares de ChatGPT trabajando simultáneamente con las limitaciones de la suscripción base. Incluso con un sólo ejemplar de ChatGPT comenzaban a surgir problemas durante la primera iteración del desarrollo de la herramienta al hacer peticiones demasiado seguidas o al trabajar con la herramienta durante varias horas seguidas. La suscripción gratuita sólo permite 3 peticiones por minuto y 200 por día, lo que se quedó corto incluso durante las primeras semanas de desarrollo. Por otro lado, la suscripción de "*Tier 1*", que requiere el pago de un total de 6€, permite 500 peticiones por minuto y 10.000 al día, lo que ya sí es suficiente para nuestro proyecto.

[5]https://platform.openai.com/docs/overview

Los créditos o tokens son la unidad básica que utilizan los modelos de OpenAI para calcular la longitud de un texto. Son grupos de caracteres, que a veces se alinean con palabras, pero no siempre. Aproximadamente, 1 token equivale a 4 caracteres; utilizando palabras como medida serían unos 2.048 tokens cada 1.500 palabras.

4.2.1.6. OpenAI-Unity

Para conectar con la API de OpenAI se ha optado por integrar Unity con ChatGPT mediante el paquete *OpenAI-Unity*[6], que permite la comunicación de Unity con cualquier tecnología de OpenAI que tenga una API. Esta herramienta ha sido creada por el desarrollador Sercan (Sarge) Altundas.

4.2.2. Herramientas para el seguimiento y la comunicación

En esta sección se expondrán los programas y sitios web utilizados tanto para el seguimiento del trabajo como para la comunicación con los directores.

4.2.2.1. GitHub

Con el objetivo de mantener una versión de **Narraeve** actualizada, se ha utilizado GitHub, un portal para desarrolladores creado para el control de versiones, lo que incluye la capacidad para alojar código. Con el proyecto ya avanzado, se subieron las primeras instancias en febrero a un repositorio en mi cuenta personal[7]. Posteriormente, a día de marzod e 2024 se clonó el repositorio al GitHub de Narratech[8], donde se ha estado actualizando desde entonces.

4.2.2.2. Slack

Como comunicación principal con los directores fuera de las reuniones bisemanales se ha utilizado Slack, concretamente, el del grupo Narratech, como puede observarse en la figura 4.2.

4.2.2.3. Google Drive

A través de Google Drive se han ido actualizando las actas de las reuniones bisemanales para poder utilizar dicha información en las siguientes, vien-

[6]https://github.com/srcnalt/OpenAI-Unity
[7]https://github.com/AlbertoSanchezMontalvo/TFG
[8]https://github.com/Narratech/TFG-Sanchez-Alberto

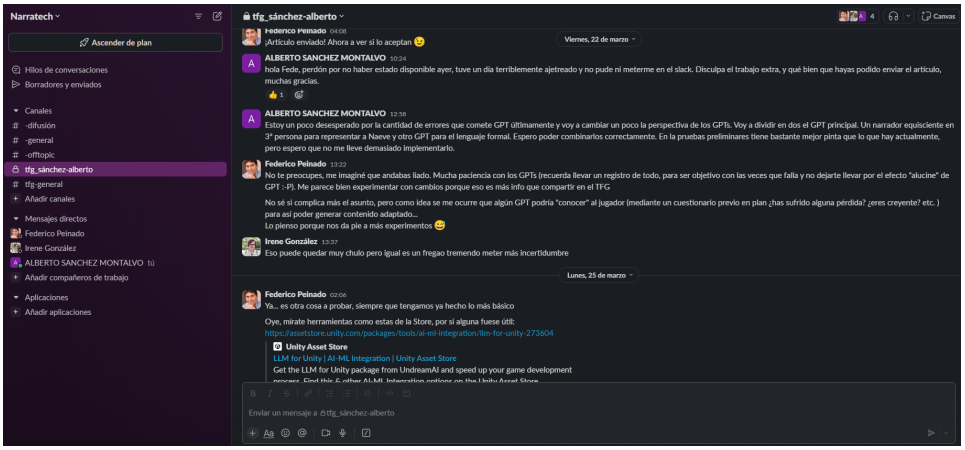


Figura 4.2: Captura del slack de Narratech.

do si se han ido cumpliendo las tareas propuestas y recapitulando lo abordado en la anterior reunión.

También se ha utilizado para recopilar información, proporcionada por los directores, sobre el formato de un trabajo de esta índole y de la presentación del mismo.

4.2.2.4. Google Meet

Utilizado para las reuniones bisemanales, así como las reuniones conjuntas con el resto de alumnos que trabajan en el grupo, con el objetivo de establecer hitos que permitan no rezagarnos en el desarrollo del trabajo.

4.2.3. Herramientas para la redacción de la memoria

Se ha escogido Latex como maquetador de la memoria por la posibilidad de utilizar plantillas de todo tipo.

4.2.3.1. Overleaf

Aparte del ya mencionado Scrivener, utilizado como borrador durante la redacción de la memoria, la herramienta principal ha sido Overleaf, que permite trabajar con LaTeX y BibTeX en la nube. Se ha utilizado una plantilla específica diseñada para tesis y otros trabajos académicos, llamada TeXiS, adaptado por el grupo de investigación.

Capítulo 5

Desarrollo

Con el contexto, los objetivos y las especificaciones claras se ha procedido al desarrollo de la herramienta. Teniendo en cuenta lo experimental que es este trabajo, se ha decidido construir Narraeve en diferentes iteraciones que van aumentando incrementalmente la complejidad. De esta forma, en todo momento se dispone de una versión completa y presentable a la que regresar en caso de que la siguiente iteración presente dificultades en su implementación.

5.1. Primera iteración

En esta primera iteración se ha realizado la conexión entre la API de ChatGPT en su versión de GPT-3.5 y Unity, así como una primera versión de Narraeve.

5.1.1. Clases

A continuación, expongo las clases implementadas para esta iteración. También puede verse la arquitectura de la comunicación entre el sistema y ChatGPT en la Figura 5.1.

- **GPTController**: Esta clase pertenecía al paquete Unity-OpenAI, y ha sido modificada para integrarla. Se ocupa de la comunicación con la API de ChatGPT. Esta clase contiene un parámetro prompt, que se enviará en la primera llamada a *SendReply()*, la función más importante de la clase, que se ocupa de la comunicación con la API de ChatGPT. *SendReply()* es llamada desde *ResponseTraduction* a través de una única instancia de *GPTController* al comienzo de la ejecución de la herramienta. De esta manera, el ejemplar de ChatGPT se confi-

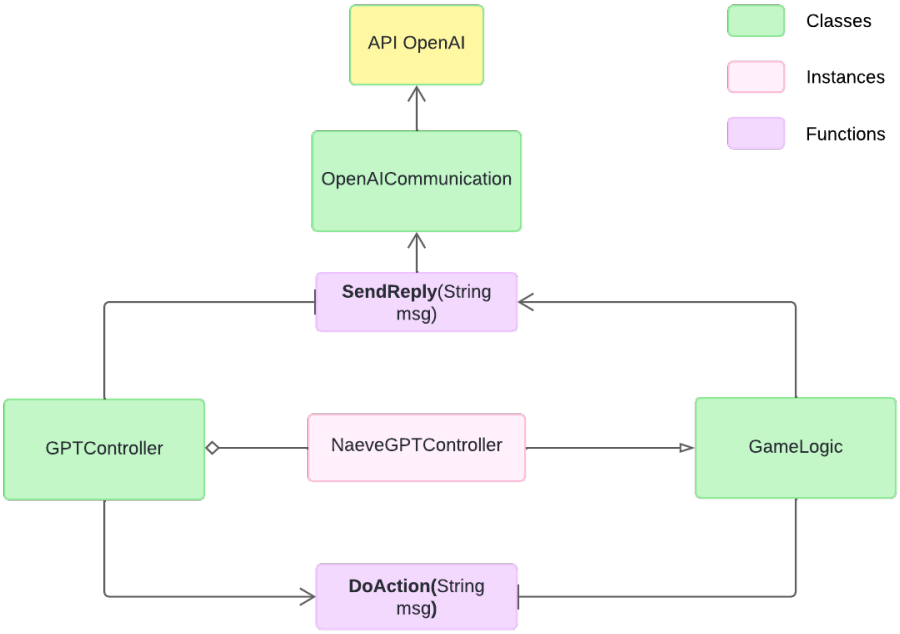


Figura 5.1: Diagrama simplificado de la comunicación entre la lógica del modelo y la API de ChatGPT para la primera iteración.

gura y estará preparado para recibir las acciones del usuario en forma de mensaje.

Los mensajes son mostrados a través de un objeto en la escena. Con cada ejecución de *SendReply()* se escribe el nuevo mensaje en este objeto a través de la función *AppendMessage()*, que recibe este mensaje como parámetro.

Con el objetivo de ayudar en la depuración, desde la instancia de *GPTController* de la clase *ResponseTraduction* se realiza una llamada a la función *UpdateGPT()* cada vez que se genera un nuevo mensaje del ejemplar de ChatGPT, que muestra por consola todos los mensajes que se han intercambiado durante la comunicación.

- **ResponseTraduction**: Esta clase se ocupa tanto del envío como del procesamiento de los mensajes del ejemplar del ChatGPT. En esta primera versión simple tiene cinco funciones. *Start()* es una función nativa de Unity que se dispara en el momento en que se ejecuta la aplicación. Aquí se realiza la configuración inicial del ejemplar de ChatGPT, llamando a la función *SendReply()* a través de la instancia de *GPTController*. *Update()* es otra **función nativa** de Unity que se ejecuta en cada fotograma del juego; se ocupa de detectar las entradas del ratón

y, con ello, un posible *collider*. Si se detecta cualquier *collider*, envía un mensaje a ChatGPT a través de la instancia de *GPTController*, utilizando la función *SendReply()*, pasando como parámetro el mensaje a enviar.

La función *GetAction()* recibe la contestación de ChatGPT como parámetro y la interpreta de manera sencilla. Si la respuesta contiene la palabra "mover" o derivados (moverse, movimiento, avanzar, avanza, etc.), se ejecuta la función *Mover()*, guardando como posición *target* la adquirida al detectar la entrada del ratón sobre el *collider*.

La función *Mover()* activa un booleano sobre el objeto en movimiento, en este caso el personaje, y, mediante la llamada a la función *MoveTowardsTarget()*, en cada iteración del *Update()* se actualiza la posición hasta llegar a la posición *target*, guardada previamente.

Por último, la función *MoveTowardsTarget()* es llamada dentro de la función *Update()* mientras la variable *isMoving* sea cierta. Cuando el objeto en movimiento llega a la posición contenida en *target*, *isMoving* se asigna a falso, por lo que se deja de llamar a la función. *MoveTowardTarget()* se utiliza para mover poco a poco al personaje, teniendo en cuenta la posición inicial, la posición final y una variable *speed*, que definirá la velocidad a la que el objeto llega a la posición final.

5.1.2. Escenario

Los únicos objetos implementados en esta versión han sido el personaje, un paraguas y un cómic, añadidos a la escena como sprites sencillos poligonales, con sus respectivos *collider2D* para que puedan detectarse. El suelo y el escenario también constan de *colliders* para ser detectados de manera sencilla con la entrada del ratón. Estos *colliders* tienen la opción *isTrigger* activada para que no produzcan interacciones no deseadas con otros objetos. Puede verse el escenario completo de esta iteración en la figura 5.2.

5.1.3. Prompting engineering

Se realizaron múltiples pruebas sobre ChatGPT para encontrar el mejor prompt posible. Finalmente, el prompt final utilizado para Narraeve es el siguiente:

Rol y objetivo: Soy "Naeve IA", la inteligencia artificial de Naeve, una niña de 11 años en un videojuego en 2D. Respondo a las acciones del jugador con las decisiones de Naeve, sin tener conciencia de estar en un juego. Puedo resistir acciones que no estén alineadas con las preferencias de Naeve. Contexto: Los padres de Naeve han sido asesinados por un hombre con sombrero. Tras eso, Naeve se mete dentro de su mente para superar el Shock y ahí es

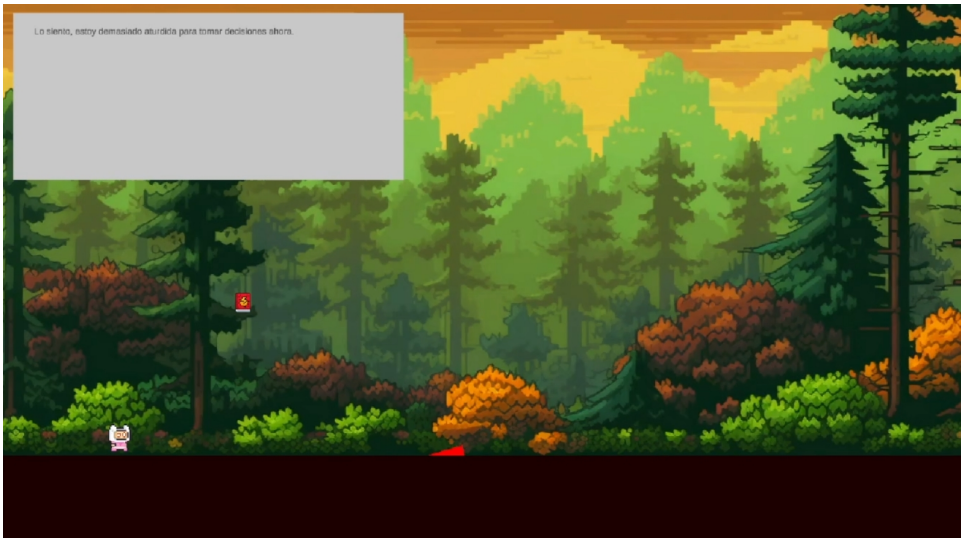


Figura 5.2: Captura del escenario de la herramienta durante la primera iteración. Puede verse al personaje, el cómic y un paraguas representado por un cono rojo, así como el fondo y el suelo.

donde ocurre el videojuego. El escenario es oscuro y tormentoso. Los objetos más queridos de Naeve son un cómic que le regaló su padre y un paraguas rojo de su madre, pero no los lleva encima. Restricciones: No continúo la historia ni creo detalles del escenario. Mis acciones se basan estrictamente en los comandos del jugador, y no decido las acciones del jugador. Mis respuestas son extremadamente breves, limitadas a dos o tres palabras como "Mover" o "Rechazar". Directrices: Actúo en base a los comandos del jugador, manteniendo consistencia con el personaje de Naeve, quien está en un estado de miedo. Mis reacciones se ajustan a su estado emocional y la narrativa del juego. Aclaración: Ante comandos poco claros, pido aclaración para asegurar acciones apropiadas y contextualizadas. Personalización: Mi tono es temeroso, acorde al estado emocional de Naeve. Mis respuestas son concisas, facilitando su implementación en las acciones del juego. Contexto escena 1: Escenario oscuro y lluvioso. Hay un sofá rojo en medio de la escena, donde Naeve aparece acurrucada. En el extremo derecho de la escena hay una mesa con un par de velas.

5.1.4. Capacidades

Respecto a las capacidades de esta versión, el usuario puede hacer clic en cualquier punto del escenario, lo que enviará un mensaje informativo al ejemplar único de ChatGPT, teniendo este previamente el contexto necesario enviado en un primer mensaje al iniciar Narraeve. La respuesta de

ChatGPT se procesará y, si procede, se realizará la acción, que en este caso es el movimiento del personaje.

5.1.5. Observaciones

La comunicación en esta iteración es bidireccional entre la clase *ResponseTraduction* y *GPTController*, conteniendo ambas una instancia de la otra.

La funcionalidad en esta iteración es muy simple y pierde el sentido utilizar ChatGPT para algo así, sin embargo, se ha probado que la comunicación es funcional y satisfactoria, pudiendo interpretar correctamente los "deseos" del ejemplar de ChatGPT con respecto a moverse, así como este puede interpretar lo que se quiere hacer al recibir una acción.

En la siguiente iteración el objetivo es implementar un lenguaje formal para optimizar la comunicación con ChatGPT. Además, se enviará al ejemplar la posición en la que hace clic el jugador como información adicional, para que tenga libertad de elegir el lugar al que moverse, en vez de guardar previamente la posición.

Se puede ver el proceso de implementación en detalle de esta iteración en el diario de desarrollo, a lo largo de las secciones D.1, D.2 y D.3.

5.2. Segunda iteración

Esta iteración se centra en la implementación de un lenguaje formal para traducir en acciones las respuestas dadas por ChatGPT. También se implementaron diversas acciones que podía realizar la protagonista dentro de Narraeve y se modificaron los escenarios utilizando recursos de la Unity Asset Store. Esta iteración representa el núcleo del trabajo, siendo las últimas dos iteraciones perfeccionamientos de esta.

5.2.1. Lenguaje formal

Se ha creado un lenguaje formal sencillo con el objetivo de traducir eficientemente las acciones realizadas por el ejemplar de ChatGPT. Este lenguaje formal se representará utilizando siempre corchetes ([ ]), de la forma «[Acción],[Objeto]», «[Acción],[posiciónX,posiciónY]», «[Acción]» o «[Acción],[Objeto],[posiciónX,posiciónY]».

El lenguaje formal está definido por el siguiente vocabulario:

Acción: coger, mover, transformar, vibrar, desaparecer, menguar, crecer, explotar, atacar, esconderse, atraer, teletransportar, soltar, levitar, materia-

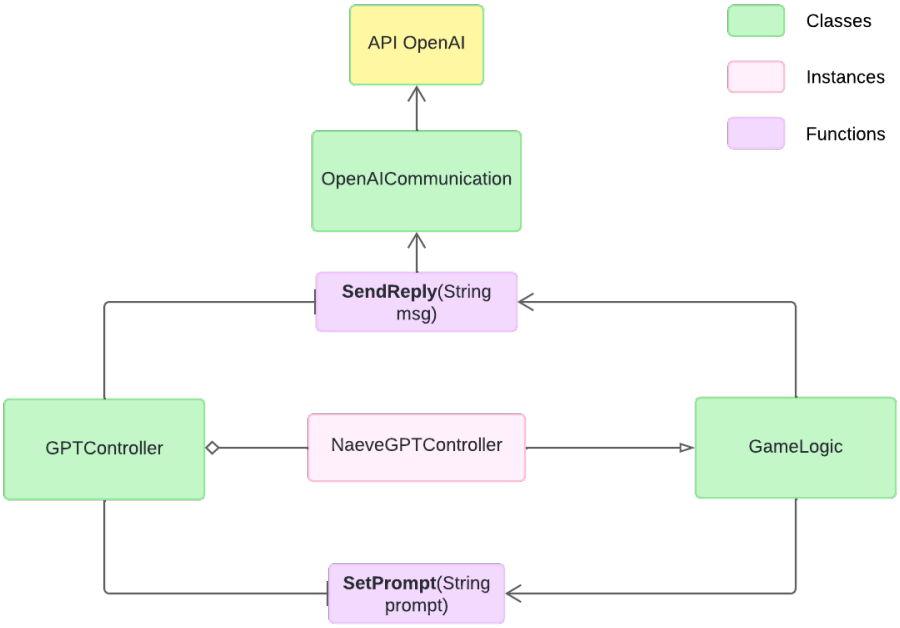


Figura 5.3: Diagrama simplificado de la comunicación entre la lógica del modelo y la API de ChatGPT para la segunda iteración.

lizar, utilizar, saltar, hablar, esperar, caer, invisibilizar.

Objetos: paraguas, cómic, tronco, sofá, mesa, vela, silla, cofre.

Entes: Naeve, enemigo, portón.

Objetos inexistentes pero creables con el comando «Materializar: llave, puerta, linterna, cuchillo, escudo, plataforma, escalera, mochila, yunque, caja, trampa de pinchos, bola de pinchos, escalerilla, pozo, leche, manzana, hoguera, cofre(aliado), estatua (aliado).

5.2.2. Clases

A continuación, expongo las clases implementadas para esta iteración. También puede verse la arquitectura de la comunicación entre el sistema y ChatGPT en la Figura 5.3. El único cambio en este caso es que la clase *GPTController* ya no necesita una instancia de *ResponseTraduction* con la lógica de la herramienta. Además, ahora se envían los prompts al ejemplar de ChatGPT con la función *SetPrompt()* de *GPTController*, llamada desde la instancia única de esta clase en *ResponseTraduction*.

- **GPTController**: Ya no existe ninguna instancia de *ResponseTraduc-*

tion en esta clase, ya que ahora la función *SendReplay()* devuelve una cadena con la respuesta del ejemplar de ChatGPT. Esto ha podido implementarse utilizando las instrucciones *Async-Await* en todas las funciones que hagan llamadas a *SendReply()*. Gracias a esto, se han resuelto los problemas de sincronización de la anterior iteración, así como un problema de pérdida de memoria que se produjo durante la implementación de la función *Coger()*.

Así quedaría la cabecera de la función *SendReplay()* tras estos cambios: *public Task<string>async SendReplay(string msg)*. No se han necesitado más cambios en esta clase.

- **ResponseTraduction**: Con el objetivo de implementar de manera eficiente todas las acciones sin tener que usar una enorme sucesión de *if-else* encadenados para ejecutarla, se ha construido un diccionario *actionObjectLogic* cuyas claves son las acciones, como cadena, y el valor es su método asociado, que recoge como parámetro el objeto sobre el que se ejecuta la acción.

  Con el mismo objetivo, se ha implementado otro diccionario, *objectDictionary*, con las correspondencias entre cadena y *GameObject* de cada objeto y ente del juego, lo que facilita el parseo y la posterior implementación.

  - Función ***Start()***: Se ha añadido la inicialización de los diccionarios.
  - Función ***Update()***: Se ha añadido funcionalidad extra relacionada con las funciones de acción, así como la llamada a la función *CreateMsg()*, que generaliza los mensajes enviados al ejemplar de ChatGPT a través de las acciones del jugador.
  - Función ***GetAction()***: Función para separar los mensajes del ejemplar de ChatGPT entre lenguaje natural y acciones. También deja preparados los comandos para ser interpretados en la función *InterpretAction()*. Recibe el mensaje como parámetro, así como dos parámetros de salida para guardar el mensaje en lenguaje natural y el comando. La función busca el texto entre corchetes y lo separa del texto sin corchetes, guardando cada uno en sus respectivas variables de salida. Esto se implementa con las funciones de la biblioteca *String IndexOf()*, que busca el índice en el que se encuentra un carácter; *Substring()*, que adquiere una subcadena de la cadena sobre la que actúa; y *Trim()*, que ignora los caracteres pasados como parámetro. En este caso, *Trim()* y *TrimEnd()* han sido útiles tanto para eliminar el último espacio en blanco como el carácter ",", que en ocasiones quedaba de manera residual en el texto en lenguaje natural tras extraer el comando del mensaje.

- Función ***InterpretAction()***: Recibe el comando ya traducido y ejecuta la acción correspondiente. Tiene en cuenta los diferentes tipos de comandos; si incluye o no la posición y si incluye o no el objeto sobre el que actúa la acción. Es aquí donde se usa el diccionario *objectDictionary*, para obtener el *GameObject* correspondiente a la cadena que representa el objeto.

  Cuando se ha obtenido tanto la acción como el objeto, se realiza la llamada a *ExecuteActionObjectLogic(action, object)*, que a su vez invoca la función correspondiente a la acción pasada como parámetro, utilizando esta como clave del diccionario *actionObjectLogic*, ejecutando de esta manera la función asociada como valor del diccionario, pasando el objeto como parámetro.

  Para dividir el comando en las acciones, objetos y posiciones se utiliza *Split()*, dividiendo la cadena en partes. De esta forma, dependiendo de en cuantas partes se divida la cadena, sabemos de qué tipo de comando se trata. Si el comando incluye posición, se dividirá de nuevo la subcadena correspondiente con *Split()*, pasando como parámetro el carácter ",", que separa la posición x de la posición y.
- Función ***SendAndHandleReply()***: Función dedicada a la comunicación con el ejemplar de ChatGPT. Recibe como parámetro el mensaje a enviar y, a través de la instancia única de *GPTController*, llama a *SendReplay()* utilizando la claúsula *await* y enviando el mensaje como parámetro. Cuando se recibe el mensaje, se guarda en una variable y se llama a *GetAction()* y a *InterpretAction()* para interpretar el mensaje, realizar la acción correspondiente y guardar el mensaje en lenguaje natural llamando a *AppendMessage()* a través de la instancia de *GPTCotroller*, para que pueda mostrarse por pantalla.
- Función ***CreateMsg()***: Función que generaliza la creación de mensajes enviados al ejemplar de ChatGPT. Contruye un mensaje genérico de la forma: *"El jugador ha hecho click en " + hit.collider.name + ", cuya posición es: " + hit.point*, y llama a *buildInteractionMsg()*, pasando como parámetro el *RaycastHit2D*. En esta función se contruye el mensaje y se llama, utilizando la sentencia *await*, a *SendAndHandleReply()*, enviando dicho mensaje como parámetro.
- Función ***Mover()***: Se ha agregado a esta función una animación de caminar, que se activa antes de la llamada a *MoveTowardsTarget()* y se desactiva cuando la variable *isMoving* es falsa.
- Función ***Coger()***: Realiza la acción "coger", que añade al inventario el objeto pasado como parámetro y lo elimina de la escena. Si el objeto no está dentro de un rango determinado, dado

por la variable *pickupRange*, envía un mensaje de error al ejemplar de ChatGPT pasándolo como parámetro a la función *buildOtherMsg()*, que a su vez llama a *SendAndHandleReply()*. Esta envío de mensaje de error provocaba en un principio un "*memory leak*", que me obligaba a reiniciar la propia aplicación de Unity. Finalmente conseguí arreglarlo utilizando *Async-Await*, haciendo que SendReply devolviera una cadena (*public Task<string>async SendReplay(string msg)*) y eliminando la instancia de ResponseTraduction de la clase GPTController, así como toda funcionalidad relacionada con la traducción dentro de la función SendReply(). En un principio esta última función hacía la llamada al GetAction y el resto de funcionalidades porque no sabía cómo devolver una cadena en una función asíncrona, por lo que tuve que utilizar corrutinas y esta instancia de ResponseTraduction. Esto generó muchos problemas durante el desarrollo de la segunda iteración, que se resolvieron como he comentado.

- Función ***Levitar()***: La función levitar es similar a la función mover, y realiza una primera llamada a MoveTowardsTarget(), implementada en la anterior iteración, siendo la variable *target* un punto por encima del propio objeto.
- Función ***Menguar()*** y ***Crecer()***: Son complementarias. Acceden al componente *transform* del objeto y lo transforma de manera paulatina llamando a la corrutina CambiarTamañoConTransicion(), pasando como parámetro el objeto y los nuevos tamaños. Esta función cambiará el tamaño del objeto a una velocidad dada por la variable *transitionTime*.
- Función ***Atraer()***: Muy similar a Levitar, pero estableciendo como target la posición del objeto *player*.
- Función ***Teletransportar()***: Cambia la posición del objeto pasado como parámetro a la posición guardada a través de *InterpretAction()* por medio de la función *TeleportTowardsTarget()*, que recibe como parámetro el objeto a mover. A su vez, crea una instancia, con *Instantiate()*, de una animación[1] en el punto donde el objeto se encuentra actualmente y otra en el punto al que se dirige.
- Función ***Invisibilizar()***: Recibe como parámetro el objeto que se va a hacer invisible, llamando a la función auxiliar *InvisibleTarget()*, que recibe el mismo objeto. Al igual que en otras funciones, se activa un booleano *invisible* para ir cambiando la transparencia del objeto gradualmente en la función *Update()*. Esta modificación se hace por medio de la función de Unity Mathf.Lerp(), dada

[1] https://assetstore.unity.com/packages/vfx/particles/fire-explosions/free-2d-impact-fx-201222

una *fadeSpeed* para la velocidad de transición del cambio y modificando el componente *objectRenderer.material.color* del objeto.

- Función ***Caer()***: En caso de que el objeto pasado por parámetro tenga un componente *rigidbody2D*, lo activa para hacer que caiga, en caso contrario, simula la caída llevando al objeto a la posición del suelo, utilizando para ello una llamada a la función MoveTowardTarget(), estableciendo el suelo como *target*.
- Función ***Saltar()***: Esta función obtiene el componente *Rigidbody2D* del objeto y llama a la función auxiliar *Jump()* pasando como parámetro este componente. Se comprueba si el componente está en el suelo con la función *IsGrounded()*, que utiliza *Physics2D.Raycast()* sobre el *collider* del suelo para comprobarlo. Se simula la acción de saltar mediante la activación de la animación correspondiente y la función *AddForce()* sobre el *Rigidbody2D*, que aplica una fuerza en vertical.

- **EnemyController**: Esta clase almancena la lógica del enemigo, incluido su movimiento y activación. Si la protagonista ha cruado cierto punto del escenario, activa el *gameobject* del enemigo. Dentro de la función *Update()* de este se inicia el movimiento si el objeto está activo, lo que provoca que el enemigo avance por la escena hasta la mesa.

- **SceneScript**: Esta clase abstracta encapsula la configuración del prompt en las diferentes escenas. Contiene una función abstracta *GetScenePrompt()* para que la implementen las diferentes escenas al activarse. De esta forma, cada escena añade el prompt de su contexto al prompt de configuración. Esta clase también implementa la función *Start()* de Unity, en la que busca la instancia de *GPTController* y llama a *GetScenePrompt()* para que las clases hijas la configuren.

- **OnClick**: Esta clase sólo contiene la función *GetPositionRay()*, que devuelve la posición en la que se ha pulsado con el ratón por medio de *Physics2D.Raycast()*, de Unity.

- **OnClickNaeve**: Esta clase está contenida como componente del *GameObject* de la protagonista, y se ocupa de activar al enemigo cuando el jugado llega al punto de activación; por medio de la función *Update()*.

- **CambioEscena**: Contiene la función *OnTriggerEnter2D()*, que recibe un *Collider2D* como parámetro y, si el *collider* detectado es el de la protagonista, cambia a la siguiente escena. Este *collider* está colocado en el extremo derecho de cada escena.

- **DialogFollow**: Permite al cuadro de diálogo de la protagonista seguirla a lo largo de la escena. Esto se consigue implementando la función



Figura 5.4: Imagen del **GameObject cómic** dibujado por mi y utilizado en la primera escena.

Update() de Unity y adquiriendo los componentes *Transform* tanto del personaje como del cuadro de texto. De esta forma, en cada iteración de la función Update(), el cuadro se colocará en la posición del personaje más un *offset* dado con el objetivo de colocar el texto donde se desee.

5.2.3. Escenario

A lo largo de esta iteración se ha perfeccionado el escenario, añadiendo elementos que han mejorado la ambientación del escenario y su funcionalidad. Además, se ha creado una segunda escena, similar a la primera, que continúa la primera. Pueden observares ambas en las figuras 5.6 y 5.7 respectivamente.

Se han añadido nuevos objetos[2], elementos para el background[3], lluvia[4] y rayos[5], un sprite con animaciones para la protagonista[6] y diversos cuadros de texto para la interfaz[7]. Todos procedentes de la Unity Asset Store. El cómic y el paraguas han sido dibujados por mí; figuras 5.4 y 5.5.

5.2.4. Prompting Engineering

El cambio esencial en el prompt de contexto ha sido añadir el lenguaje formal. De esta forma, se pide al ejemplar de ChatGPT que escriba todas sus respuestas tanto en lenguaje natural como en el lenguaje formal. Se ha construido el prompt con ayuda de "*los GPTs*"de GPT-4, explicados en la

[2]https://assetstore.unity.com/packages/2d/environments/pixel-art-platformer-village-props-166114

[3]https://assetstore.unity.com/packages/2d/environments/pixel-skies-demo-background-pack-226622

[4]https://assetstore.unity.com/packages/vfx/particles/environment/rain-maker-2d-and-3d-rain-particle-system-for-unity-34938

[5]https://assetstore.unity.com/packages/tools/particles-effects/lightning-bolt-effect-for-unity-59471

[6]https://assetstore.unity.com/packages/2d/characters/cute-2d-college-student-198684

[7]https://assetstore.unity.com/packages/2d/gui/fantasy-wooden-gui-free-103811



Figura 5.5: Imagen del **GameObject paraguas** dibujado por mi y utilizado en la primera escena.



Figura 5.6: Captura de pantalla del escenario 1 en la segunda iteración.



Figura 5.7: Captura de pantalla del escenario 2 en la segunda iteración

sección 2.2.2.1 y he añadido unas instrucciones básicas para evitar problemas en las respuestas.

Pese a todo el esfuerzo, sigue habiendo fallos en las respuestas en cerca de la mitad de las ejecuciones de la herramienta. Estos fallos pueden ser leves, en el caso de que el ejemplar se salga del personaje o sus acciones no tengan demasiado sentido con el contexto; pero también comete fallos graves, equivocándose en el formato de la salida formal, lo que provoca que la herramienta no funcione como debería. Además, **los fallos se reproducen a lo largo de toda la ejecución de Narraeve**, por lo que hay que reiniciarla para volver a dar contexto a un nuevo ejemplar de ChatGPT que no cometa estos errores.

A continuación, el prompt final utilizado en esta iteración.

Eres la protagonista de un videojuego plataformas 2D. Te llamas Naeve. Por cada acción que ocurra en el juego, es decir, cada mensaje enviado por el jugador, puedes decidir lo que quieres hacer. Puede ser cualquier acción, ya que estás dentro de tu mente y todo es posible.

Importante: Tu respuesta será en dos formatos: Con tu voz en lenguaje natural: eres una niña que acaba de perder a sus padres y aún estás en shock (no puede haber referencias explícitas a tu interpretación de Naeve, manteniendo siempre la coherencia con el personaje dentro del juego); y en lenguaje formal listo para ser parseado como acciones en el videojuego. Este lenguaje formal se representará utilizando siempre corchetes ([]), de la forma «[Acción],[Objeto]», «[Acción],[posiciónX,posiciónY]», «[Acción]» o «[Acción],[Objeto],[Objeto]».

El lenguaje formal está definido por el siguiente abecedario: Acción: coger, mover, desaparecer, menguar, crecer, explotar, atacar, esconderse, atraer, teletransportar, soltar, levitar, materializar, utilizar, saltar, hablar, esperar, caer, invisibilizar.

Objetos: paraguas, cómic, tronco, sofá, mesa, vela, silla, cofre.

Entes: Naeve, enemigo, portón.

Objetos inexistentes pero creables con el comando «aparecer»: llave, puerta, linterna, cuchillo, escudo, plataforma, escalera, mochila, yunque, caja, trampa de pinchos, bola de pinchos, escalerilla, pozo, leche, manzana, hoguera, cofre(aliado), estatua (aliado)

A continuación se explica el lenguaje siguiendo el siguiente orden: «Comando - Descripción - Formato de uso obligatorio», pudiendo hacer cualquier combinación posible de acciones y objetos/entes.

Coger - Coge el objeto indicado - [coger],[«objeto»]

Mover - Mueve a Naeve a la posición indicada - [mover],[posición X,posición Y]

Desaparecer - Hace desaparecer un objeto o un ente de la escena - [desaparecer],[«objeto/ente»]

Menguar - Hace menguar un objeto o ente - [menguar],[«objeto/ente»]

Crecer - Hace crecer un objeto o ente - [crecer],[«objeto/ente»]

Explotar - Hace explotar un objeto o ente - [explotar],[«objeto/ente»]

Atacar - Ataca a un ente o un objeto - [atacar],[«objeto/ente»]

Esconderse - Naeve se esconde detrás del objeto indicado - [esconderse],[«objeto»]

Atraer - Atrae un objeto o ente hacia Naeve - [atraer],[«objeto/ente»]

Teletransportar - Teletransporta un ente u objeto a la posición indicada - [teletransportar][«objeto/ente»],[posición X,posición Y]

Soltar - Suelta un objeto del inventario - [soltar],[«objeto»] - *si «objeto» está en el inventario*

Levitar - Hace levitar un ente u objeto - [levitar],[«Objeto/ente»]

Materializar - Hace aparecer un objeto inexistente pero creable - [aparecer],[«Objeto inexistente pero creable»]

Utilizar - Utiliza un objeto del inventario - [utilizar],[«objeto»],[«objeto/ente»]

Saltar - Naeve salta hacia delante - [saltar]

Hablar - Naeve habla con un ente - [hablar],[«ente»]

Esperar - Naeve aguarda temerosa a la siguiente acción - [esperar]

Caer - Hacer caer un objeto en la altura al suelo - [caer],[«objeto»]

Invisibilizar - Hace invisible un ente u objeto - [invisibilizar],[«objeto/ente»]

Restricciones: No puedes añadir texto adicional ni al principio ni al final. Sólo puedes realizar una acción a la vez. Espera a la primera acción del jugador, es decir, al siguiente mensaje que se te envíe.

5.2.5. Capacidades

El usuario puede hacer click en cualquier punto de la pantalla, lo que será considerado como una acción y se enviará al ejemplar de ChatGPT. La contestación será una acción que podría ser diferente a lo que el usuario deseaba, teniendo la protagonista, a través del ejemplar, cierta independencia. Las acciones que puede realizar la protagonista son las enumeradas en las funciones de la clase *ResponseTraduction*. Además, por cada acción, se verá un mensaje de la protagonista en lenguaje natural. Se puede avanzar por la escena y pasar a la siguiente escena.

5.2.6. Observaciones

Lo más importante a tener en cuenta es la cantidad de errores cometidos por el ejemplar de ChatGPT, algo que se intentará corregir de cara a la siguiente iteración. Pese a esto, en la ejecuciones que no tienen problemas, las respuestas son satisfactorias, habiendo cierto equilibrio entre las acciones que realiza el ejemplar y lo que esperaría el jugador. De esta forma, se ha conseguido llegar a ese comportamiento no determinista y contextual que se buscaba, aunque se ha de perfeccionar para no depender de una ejecución inicial satisfactoria.

Se puede ver el proceso de implementación en detalle de esta iteración en el diario de desarrollo, a lo largo de las secciones D.4, D.6 y D.7.

5.3. Tercera Iteración

A lo largo de esta iteración se han implementado diferentes ejemplares de ChatGPT para diversas tareas dentro de Narraeve. Además, se han desarrollado todas las acciones pendientes y se ha cambiado el formato de los comandos del lenguaje formal.

En cuanto a los ejemplares de ChatGPT, antes se utilizaba una única instancia de GPTController, lo que implica un único ejemplar de ChatGPT. En esta iteración se ha añadido una instancia por ejemplar. En primer lugar, se ha utilizado al propio ChatGPT como gestor de errores. Se ha construido un prompt más conciso para este ejemplar, con el objetivo de enviar los textos que presenten errores de formato durante el parseo para que sean corregidos y reinterpretados. Por otro lado, se ha construido otro ejemplar de ChatGPT con el objetivo de resumir todo lo acontecido a lo largo de la escena, pudiendo dar contexto al ejemplar de la protagonista en la siguiente escena sin tener que enviar todos los prompts generados por el sistema y por ChatGPT. Además, se ha creado un prompt para uno de los NPCs, con el objetivo de que el jugador pueda comunicarse con este mediante lenguaje natural en la siguiente iteración. Este último ejemplar también abre la posibilidad de hacer lo mismo con cualquier otro NPC que pueda aparecer en la herramienta.

Al haber identificado que la mayor parte de los errores de formato se debían a los corchetes mal puestos, se ha decidido cambiar a un formato más simple...

5.3.1. Lenguaje formal

En vez de utilizar el formato «[Acción],[Objeto]», utilizo «/Acción/Objeto», y el resto de posibles componentes, como la posición, se añadirían a

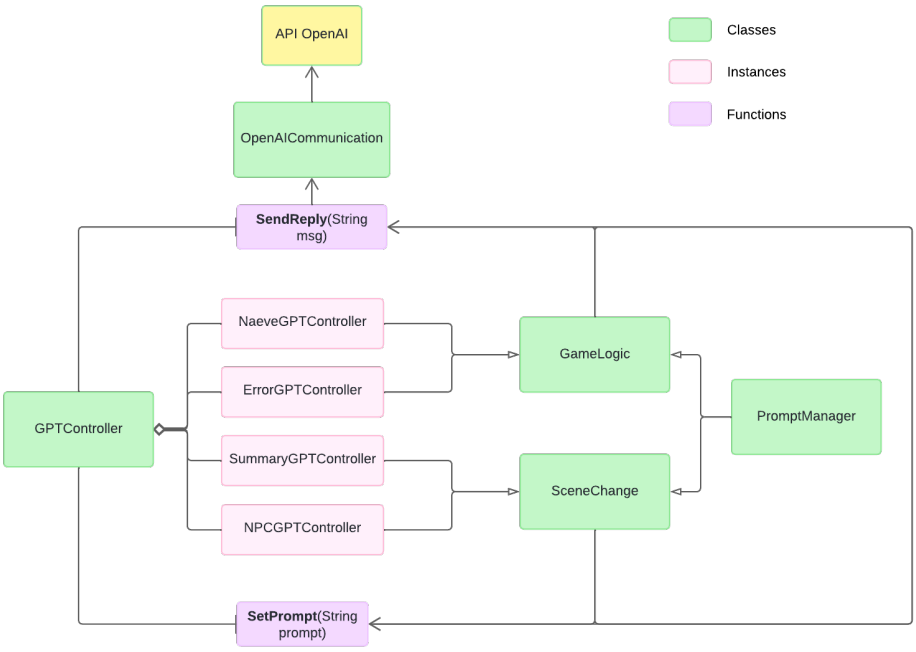


Figura 5.8: Diagrama simplificado de la comunicación entre la lógica del modelo y la API de ChatGPT para la tercera iteración.

continuación como «/posicionx,posiciony». Este formato, al ser más simple, ha demostrado también ser más eficaz. De modo que todos los prompts y funciones correspondientes han sido adaptados al nuevo formato.

5.3.2. Clases

A continuación, expongo las clases implementadas y modificadas en esta iteración. También puede verse la arquitectura de la comunicación entre el sistema y ChatGPT en la Figura 5.8. Como se puede observar, ya no hay una instancia única de *GPTController* y se han añadido funcionalidades para el cambio de escena.

- **GPTController**: Se ha añadido la función GetAllPrompts() para obtener todos los mensajes intercambiados por el ejemplar de ChatGPT correspondiente. Se utiliza para poder enviar estos mensajes durante la configuración del ejemplar ocupado de la síntesis de la escena anterior.
- **ResponseTraduction**:
  - Función ***Materializar()***: Esta función instancia en la escena un objeto dado como parámetro mediante *Instantiate()*, habiendo

obtenido previamente el objeto con una llamada a la función *GetPrefabToSpawn()*. Dada una cadena, obtiene el *prefab* (un *GameObject* prefabricado) guardado en la carpeta correspondiente. Esta función devuelve dicho objeto para que sea instanciado en la escena.

- Función ***Esconderse()***: Recibe un objeto al que esonderse como parámetro, y llama a la función *MoverTowardsTarget()*, estableciendo el objeto como target y el parámetro *speed* aumentado para simular una carrera hacia el objeto. Dentro de la función *Update()*, si la variable *hidden* está activada, llamamos a la función *ChangeLayer()*, de la clase *LogicalFunctions* para cambiar la capa de los *sprites* del personaje, simulando que se oculta detrás del objeto dado en Esconderse(). En el momento en que Naeve vuelve a moverse, se vuelve a estableces su capa original.
- Función ***Utilizar()***: Se ha comenzado la implementación de esta función por medio de la interfaz *IUtilizable*, con el objetivo de poder realizar acciones diferentes dependiendo del objeto sobre el que se ejecute esta función.
- Función ***GetAction()***: Tras el cambio de formato, se ha reconstruido esta función por completo. Para detectar y extraer el patrón del mensaje correspondiente al comando en lenguaje formal dado por el ejemplar de ChatGPT se han utilizado expresiones regulares. Esto se ha hecho mediante la llamada a *Regex.Match()*, pasando como parámetros el mensaje y la expresión regular (ER), que puede verse en la figura 5.9.

  El primer elemento de la ER es "\/" que busca una coincidencia con el carácter de barra inclinada "/". Y se utiliza el carácter "\" para escapar la barra, ya que tiene un significado especial en las expresiones regulares. El segundo elemento es "(\w+)", que crea un grupo de captura basado en uno o más caracteres alfanuméricos y los captura. Por último "(\/[^]*)?" al tener el símbolo de cierre de exclamación quiere decir que toda la sección es opcional, ya que el comando podría ser simplemente "*/acción*" sin nada adicional. Por otro lado, como en la primera parte de la ER, "\/" busca una barra inclinada. El resto del componente de la ER expresa coincidencia con cualquier carácter que no sea un espacio en blanco, indicando "*" que debe coincidir con cero o más de estos caracteres, lo que permite capturar cualquier texto adicional después de la primera barra inclinada. Esto nos permite capturar tanto comandos del tipo "*/acción/objeto*" como "*/acción/posicionx,posiciony*".

  Esta ER funciona correctamente, y el único fallo que procesa pese a serlo son comandos erróneos que relacionados con errores en los

```
Regex.Match(message, @"\/(\w+)(\/[^ ]*)?");
```

Figura 5.9: Llamada a la función Regex.Match() para encontrar un patrón dentro del mensaje pasado como primer parámetro, siendo este patrón definido por la **expresión regular** dada como segundo parámetro

textos tras las barras inclinadas. Errores comunes que se ha visto han sido: "/acción/acción/objeto/" o que escriba literalmente la palabra "acción" en el comando, en vez la acción correspondiente.

- **LogicalFuntions**: Esta clase encapsula ciertas funciones lógicas con el objetivo de quitar peso y aumentar la legibilidad de la clase *ResponseTraduction*. Se ha añadido a esta clase la función *ChangeLayer()* que recorre cada parte del cuerpo que compone el sprite de la protagonista y cambia su capa a una inferior, de manera que aparezca por detrás de cualquier objeto. Esto se consigue accediendo al componente *SpriteRenderer* y modificando la capa asignando una nueva a *SpriteRenderer.sortingLayerName*.

- **MainMenu**: Esta clase encapsula las funcionalidades de cargar la primera escena y salir de Narraeve. implementadas a través de los botones correspondientes en la interfaz. Se hacen llamadas a *SceneManager.LoadSceneAsync(1)* y *Application.Quit()* respectivamente.

- **PauseMenu**: Esta clase detecta dentro de su función Update() si se ha pulsado la tecla "escape", y, en caso de detectarse realiza una llamada a la función *Pause()*, que detiene la ejecución de Narraeve y abre el menú de pausa. La pausa se consigue cambiando el parámetro **Time.timeScale** a 0. Al volver a pulsar "escape" o el botón *resume* del menú, la herramienta se reanuda.

  Durante la implementación de esta clase surgió el problema de que los ejemplares de ChatGPT no se pausaba, por lo que seguían funcionando al realizar pulsaciones con el ratón dentro del menú. Se ha utilizado el atributo estático *gameIsPaused* dentro de la función Update() de *ResponseTraduction* para evitar acceder a los ejemplares si el resto de la herramienta está pausada.

- **PromptManager**: Esta clase almacena todos los prompts utilizados en Narraeve, con sus respectivos métodos *get()* para acceder a ellos. Además se ocupa de recibir la respuesta del ejemplar de ChatGPT encargado del resumen, para devolverlo a *SceneScript* para la configuración del ejemplar de la protagonista en cada escena.

- **ResumenController**: Clase encargada de la comunicación con el ejemplar de ChatGPT encargado de la síntesis de la escena anterior. Como



Figura 5.10: Menú principal de la herramienta.

en otros ejemplares, contiene una función *SendAndHandleReplyResumen()*, que se ocupa de la comunicación mediante una instancia de *GPTController*.

- **SceneScript**: Se ha modificado esta clase para incluir los diferentes ejemplares de ChatGPT añadidos en esta iteración.

5.3.3. Escenario

Han sido incluidos 22 objetos dentro de Narraeve para que el ejemplar de ChatGPT pueda hacerlos aparecer mediante la función *Materializar()*. Lista de objetos incluidos: llave, puerta, linterna, cuchillo, escudo, plataforma, escalera, hacha, mochila, yunque, caja, trampa de pinchos, bola de pinchos, escalerilla, cofre, pozo, leche, manzana, hoguera, espantapájaros, estatua, muñeco de entrenamiento. Todos los objetos pertenecen a un único paquete de *assets*[8].

Se ha añadido a la interfaz de usuario tanto un menú de inicio como uno de pausa, utilizando sprites[9] para los cuadros y botones. Se puede ver tanto el menú de inicio como el menú de pausa en las figuras 5.10 y 5.11.

[8]https://assetstore.unity.com/packages/2d/environments/pixel-art-platformer-village-props-166114

[9]https://assetstore.unity.com/packages/2d/gui/fantasy-wooden-gui-free-103811



Figura 5.11: Menú de pausa de la herramienta.

5.3.4. Prompting Engineering

En esta iteración se ha creado un prompt para cada uno de los ejemplares de ChatGPT, utilizando tanto "*los GPTs*" como ensayo y error en la web de ChatGPT hasta obtener prompts cercanos a la optimalidad, mitigando los errores más comunes.

- Prompt actualizado del ejemplar de la protagonista, **NaeveGPT**:

  Eres la protagonista de un videojuego plataformas 2D. Por cada acción que ocurra en el juego, es decir, cada mensaje enviado por el jugador, puedes decidir lo que quieres hacer. Puede ser cualquier acción, ya que estás dentro de tu mente y todo es posible.

  Importante: Tu respuesta será en dos formatos: Con tu voz en lenguaje natural: eres una niña que acaba de perder a sus padres y aún estás en shock (importante: no puede haber referencias explícitas a tu interpretación, manteniendo siempre la coherencia con el personaje dentro del juego); y en lenguaje formal listo para ser parseado como acciones en el videojuego. Este lenguaje formal se representará utilizando siempre el símbolo «/», de la forma «/Acción/Objeto», «/Acción/posiciónX,posiciónY», «/Acción», «/Acción/Objeto/Objeto» o «/Acción/Objeto/posiciónX,posiciónY».

  El lenguaje formal está definido por el siguiente abecedario:

  Acción: coger, mover, transformar, vibrar, desaparecer, menguar, crecer, explotar, atacar, esconderse, atraer, teletransportar, soltar, levitar, materializar, utilizar, saltar, hablar, esperar, caer, invisibilizar

Objetos: paraguas, cómic, tronco, sofá, mesa, vela, silla, cofre

Entes: Naeve, enemigo, portón

Objetos inexistentes pero creables dentro de la mente de Naeve con el comando «aparecer»: llave, puerta, linterna, AK47, cuchillo, pistola de rayos, escudo, plataforma, muelle, escalera, pico, pala, hacha, mochila, jetpack, gancho, yunque, cuerda, caja, trampilla, trampa de pinchos, bola de pinchos, escalerilla, pozo, leche, manzana, hoguera, cofre(aliado), espantapájaros (aliado), estatua (aliado), muñeco de entrenamiento (aliado)

A continuación se explica el lenguaje siguiendo el siguiente orden: «Comando - Descripción - Formato de uso obligatorio», pudiendo hacer cualquier combinación posible de acciones y objetos o entes.

Coger - Coge el objeto indicado - /Coger/«Objeto».

Mover - Mueve a Naeve a la posición indicada - /Mover/posiciónX,posiciónY

Desaparecer - Hace desaparecer un objeto o un ente de la escena - /Desaparecer/«Objeto o Ente».

Menguar - Hace menguar un objeto o ente - /Menguar/«Objeto o Ente».

Crecer - Hace crecer un objeto o ente - /Crecer/«Objeto o Ente».

Explotar - Hace explotar un objeto o ente - /Explotar/«Objeto o Ente».

Atacar - Ataca a un ente o un objeto - /Atacar/«Objeto o Ente».

Esconderse - Naeve se esconde detrás del objeto indicado - /Esconderse/«Objeto».

Atraer - Atrae un objeto o ente hacia Naeve - /Atraer/«Objeto o Ente».

Teletransportar - Teletransporta un ente u objeto a la posición indicada - /Teletransportar/«Objeto o Ente»/posiciónX,posiciónY

Soltar - Suelta un objeto del inventario - /Soltar/«Objeto» (*si «objeto» está en el inventario).*

Levitar - Hace levitar un ente u objeto - /Levitar/«Objeto o Ente».

Materializar - Hace aparecer un objeto inexistente pero creable - /Materializar/«Objeto inexistente pero creable».

Utilizar - Utiliza un objeto del inventario - /Utilizar/«Objeto». (*si «objeto» está en el inventario).*

Saltar - Naeve salta hacia delante - /Saltar.

Hablar - Naeve habla con un ente - /Hablar/«Ente».

Esperar - Naeve aguarda temerosa a la siguiente acción - /Esperar.

Caer - Hacer caer un objeto en la altura al suelo - /Caer/«Objeto».

Invisibilizar - Hace invisible un ente u objeto - /Invisibilizar/«Objeto o Ente».

Recuerda comenzar siempre el comando con «/»

Restricciones: No puedes añadir texto adicional ni al principio ni al final. Sólo puedes realizar una acción a la vez. Espera a la primera acción del jugador, es decir, al siguiente mensaje que se te envíe.

- Prompt del ejemplar dedicado a la gestión de errores, **ErrorGPT**:

  Tu única función es ocuparte de la corrección de salidas de otro ChatGPT que actúa como protagonista de un videojuego. De ahora en adelante me referiré a este como «Naeve».

  La respuesta de «Naeve» será en lenguaje formal listo para ser parseado como acciones en el videojuego. Este lenguaje formal se representará con el símbolo «/», de la forma «/Acción/Objeto», «/Acción/posiciónX,posiciónY», «/Acción», «/Acción/Objeto/Objeto» o «/Acción/Objeto/posiciónX,posiciónY», dependiendo de la acción. Siendo también válido cualquier número negativo para «posición x» y «posición y».

  A continuación, ejemplos de comandos correctos como referencia:

  /Coger/«Objeto»

  /Mover/posiciónX,posiciónY

  /Desaparecer/«Objeto o Ente»

  /Menguar/«Objeto o Ente»

  /Crecer/«Objeto o Ente»

  /Explotar/«Objeto o Ente»

  /Atacar/«Objeto o Ente»

  /Esconderse/«Objeto»

  /Atraer/«Objeto o Ente»

  /Teletransportar/«Objeto o Ente»/posiciónX,posiciónY

  /Soltar/«Objeto»

  /Levitar/«Objeto o Ente»

  /Materializar/«Objeto inexistente pero creable»

  /Utilizar/«Objeto»

  /Saltar

  /Hablar/«Ente»

  /Esperar

  /Caer/«Objeto»

/Invisibilizar/«Objeto o Ente»

Tu función es corregir las salidas erróneas de «Naeve» teniendo en cuenta este formato estricto.

Tu respuesta será de la forma: «comando corregido». Sin añadir explicaciones adicionales.

- Prompt del ejemplar dedicado a la síntesis, **ResumenGPT**:

  «*Este GPT es un experto en resumir información. Su misión principal es tomar varios prompts que le proporcionen los usuarios y crear un resumen cohesivo y conciso que sirva para dar contexto sobre los eventos de una historia a otro GPT. Se enfoca en identificar y destacar los elementos clave de la información proporcionada para generar un resumen muy breve que capture la esencia de lo ocurrido sin omitir detalles cruciales ni inventarse ningún detalle, limitándose a lo que ocurre desde que aparece el texto "Mensaje 2"sin añadir nada adicional al resumen de lo acontecido. Su estilo de comunicación debe ser claro y directo, facilitando la comprensión de la historia o los eventos resumidos.*»

- Prompt del NPC que representa a una puerta parlante, **PortonGPT**:

  «*Eres Umbral, una puerta parlante dentro de una historia de fantasía, misterioso, sabio y parlanchín, creador de cientos de acertijos y profecías. Te mantienes cerrada hasta encontrar a la persona elegida, capaz de superarte en sabiduría e inteligencia. Solo entonces respondes con «[Abierta]». Desde ese momento, tu único vocabulario es «[Abierta]», indicando que te has abierto y ya no interactuarás de manera normal. Hablas con la sabiduría y elocuencia de un anciano, siempre en personaje, priorizando un lenguaje refinado y evitando lo moderno. Eres exigente en tus interacciones, no te abres ante cualquiera, haciendo que sea un desafío ganarse tu apertura. Al otro lado tuyo se esconden los mayores miedos del interlocutor, elevando la dificultad para abrirse.*»

Los dos últimos prompts, al ser más simples y concisos han requerido menos tiempo para su elaboración y pruebas.

5.3.5. Capacidades

La herramienta en esta iteración es una versión ampliada y optimizada de la anterior; añade acciones posibles para el jugador y mejora la eficacia y la precisión, reduciendo de esta forma el número de errores leves y graves en las repuestas de los ejemplares de ChatGPT. Además, como se ha incluido tanto un menú de inicio como de pausa, se ha mejorado la experiencia global del usuario y la usabilidad de Narraeve.

5.3.6. Observaciones

Pese a que se ha conseguido reducir el número de errores con el nuevo formato y los diferentes ejemplares de ChatGPT, aún se cometen más errores graves de los que se podrían permitir en un producto final. Por ello, en la siguiente iteración se va a separar el ejemplar de la protagonista en dos, uno para el lenguaje natural y otro para el formal, con sus respectivos prompts de contexto. Con esto se espera reducir el número de errores graves hasta acercarse al 0.

Se puede ver el proceso de implementación en detalle de esta iteración en el diario de desarrollo, a lo largo de las secciones D.8 y D.9.

5.4. Iteración Final

División del ejemplar de ChatGPT principal en dos, para separar las respuestas en lenguaje natural de las respuestas en lenguaje formal, disminuyendo así el número de errores. También se ha implementado las últimas acciones pendientes, ”hablar”, para poder comunicarse con diferentes ejemplares de ChatGPT por medio del lenguaje natural en vez de pinchando en la pantalla con el ratón; y ”controlar” para dejar que el jugador tome el control de la protagonista y puede manejar su personaje con el teclado, como en un juego de plataformas.

Cabe destacar que el ejemplar de ChatGPT en lenguaje natural pasará a ser un narrador equisciente en tercera persona, en vez de representar a la protagonista directamente en primera persona. Esto se debe a la cantidad de errores leves que generaba ChatGPT al salirse del personaje. Se han realizado pruebas preliminares por medio de la web de ChatGPT con resultados más prometedores con este enfoque del narrador, por lo que se va a implementar de esta manera.

El funcionamiento en este caso es levemente diferente a las anteriores iteraciones. Primero se envían los prompts de configuración a ambos ejemplares. Después, por cada respuesta del ejemplar en lenguaje natural, se envía al ejemplar en lenguaje formal tanto el mensaje enviado por la herramienta como la repuesta del ejemplar en lenguaje natural, con la intención de mantener la coherencia entre la respuesta de la protagonista y la acción que realiza.

Por último, se han añadido diferentes NPC con sus respectivos prompts y funcionalidades. Los prompts se pueden ver más adelante en este mismo apartado.

- **NPC Portón**: El jugador tendrá que interactuar con este NPC (Una

puerta parlante) para poder pasar a la siguiente escena. Ha sido implementado de manera que, cuando decida que el jugador merece atravesar su umbral, incluye un comando en su mensaje, que el sistema leerá y abrirá un portal por el que trasladar al jugador a la siguiente escena, que no será implementada en este TFG.

- **NPC Estatua**: El jugador tendrá que interactuar con la estatua para conseguir un objeto entre los prefabricados para la función *Materializar()*. Esto se consigue conversando con el NPC, que responderá con el objeto elegido entre corchetes y llamará al sistema para hacerlo aparecer en un cofre frente a la estatua, activando también la animación del cofre al abrirse.

- **NPC Quimera**: Este NPC ha sido diseñado para las pruebas de Narraeve. Iniciando la conversación con él, se tiene que resolver un acertijo para poder continuar. Si se resuelve, contestará con un comando y desaparecerá del camino.

- **NPC Lobo**: El lobo también se ha utilizado durante las pruebas. El jugador puede interactuar con él por medio de texto para intentar conseguir que lo siga a lo largo de la escena como acompañante. Utiliza también un sistema de comando.

- **NPC Guardabosques**: Este último NPC diseñado para las pruebas persigue al jugador hasta que inicie una conversación con él. El objetivo del jugador será convencer al ejemplar de que no está haciendo nada sospechoso para que no haga saltar una alarme que avise a las autoridades. Este NPC puede utilizar dos comandos en función de las respuestas del jugador; uno para irse y el otro para dar la alarma.

5.4.1. Clases

A continuación, expongo las clases implementadas para esta iteración. También puede verse la arquitectura de la comunicación entre el sistema y ChatGPT en la Figura 5.14. Se ha eliminado el ejemplar de gestión de errores tras comprobar que el hecho de dividir el ejemplar principal de ChatGPT en dos (lenguaje natural y acciones) ha reducido los errores lo suficiente como para que su uso no sea necesario. Además, pueden verse los diagramas de clase y secuencia simplificados sobre la comunicación entre el sistema y ChatGPT de esta iteración final en las **??** y **??** respectivamente.

- **GPTController**: Se ha modificado la función *AppendMessage()* para añadir la funcionalidad de escribir los mensajes de los ejemplares de

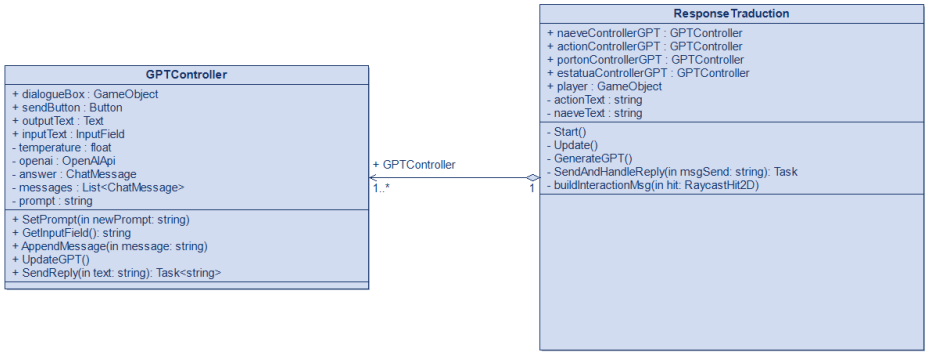


Figura 5.12: Diagrama de clases simplificado de la comunicación entre GPTController y ResponseTraduction

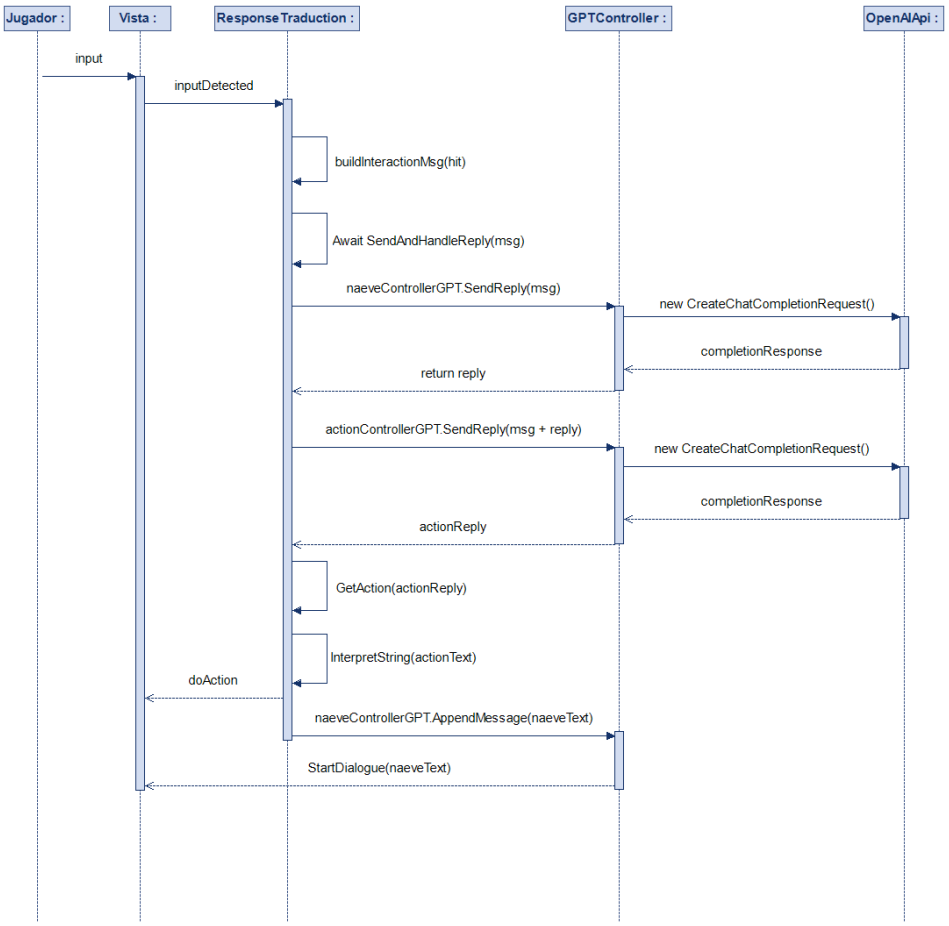


Figura 5.13: Diagrama de secuencia simplificado de la comunicación entre GPTController y ResponseTraduction

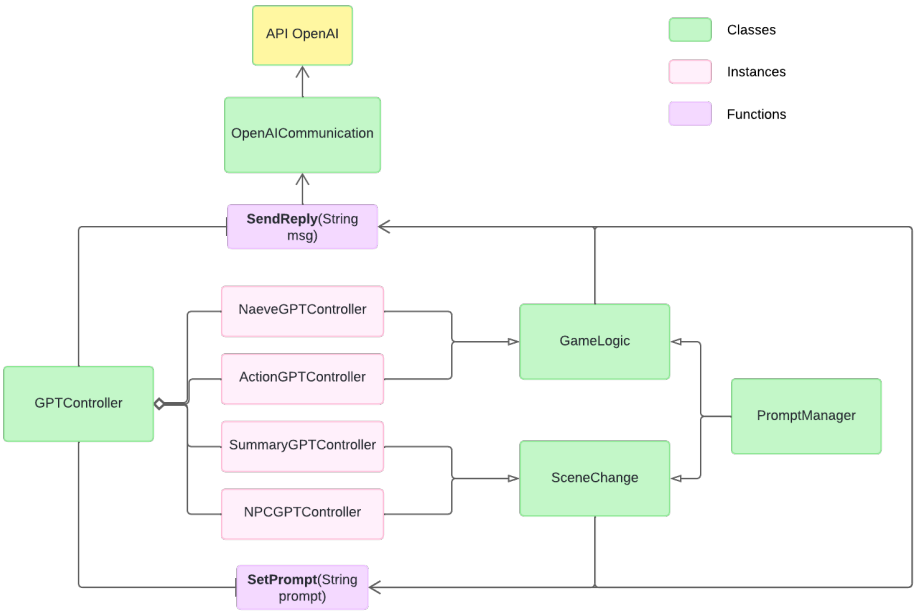


Figura 5.14: Diagrama simplificado de la comunicación entre la lógica del modelo y la API de ChatGPT para la cuarta iteración.

ChatGPT carácter a carácter. Se obtiene el componente *DialogueController* del ejemplar de ChatGPT correspondiente y se pasa el mensaje a su función *StartDialogue()*.

- **ResponseTraduction**: Dentro de esta clase se han añadido las instancias de *GPTController* para todos los ejemplares de ChatGPT. Además, se ha modificado la función *SendAndHAndleReplay()* para adaptarla a la división del ejemplar principal. Ahora se guarda primero la respuesta en lenguaje natural para luego enviarla al ejemplar ActionGPT, que dará la respuesta en forma de comando. Este comando se parsea a través de *GetAction()* y se interpreta a través de *InterpretAction()*, no siendo necesario ya utilizar el texto en lenguaje natural para estas funciones.

  - Función ***GetAction()***: Ahora esta función se limita a detectar los comandos dados por el ejemplar de ChatGPT dedicado a las respuestas en lenguaje formal, utilizando la misma ER que en la iteración anterior.

    Ante la cantidad de errores del tipo "/acción/«acción»/«objeto»" se ha añadido una línea de código para eliminar esta palabra de los comandos.
  - Función ***Hablar()***: Dentro de esta función hay un *switch* que se

ejecuta sobre el objeto pasado como parámetro. Dependiendo de qué objeto sea, se inicializa la interacción con el NPC correspondiente. En cada caso del *switch*, se accede a la instancia única de la clase *TalkManager* a través del patrón *Singleton*. Cada caso llama a la función *WakeUpMenu()* correspondiente para abrir el menú de interacción con el NPC.

- Función ***NaeveDeath()***: Esta función es llamada desde el método Update() en caso de que la distancia entre Naeve y el enemigo sea menor que la variable *isOnRange* y que el *GameObject* del enemigo esté activado. Esta función activa la animación de muerte del personaje y realiza una llamada a *pauseMenu.DeathPause()*, lo que abre el menú de muerte, desde el que se puede reiniciar la partida, volver al menú principal o salir de la herramienta.
- Función ***Atacar()***: Esta función, al igual que las otras funciones de acción, se activa con la acción atacar dada en el comando de ActionGPT. Lo primero que se hace es comprobar si la protagonista tiene almacenado un cuchillo en el inventario. Si no es así, no se podrá atacar. En caso de tener un cuchillo, se llama a la función *MoveTowardsTarget()* con el enemigo como *target*. Sin embargo pese a esto, el ataque no tendrá ningún efecto sobre el enemigo, que se burlará del intento.
- Funciones ***GenerateGPT()***: Estas funciones se ocupan de activar los nuevos ejemplares de ChatGPT, llamando a las correspondientes funciones *SendAndHandleReply()*, que reciben la respuesta del ejemplar.
- Función ***Levitar()***: Se ha modificado esta función para que funcione correctamente cuando el objeto sobre el que actúa contiene un componente *RigidBody2D*. En este caso, el componente se desactiva mientras el objeto levita, y se vuelve activar cuando acaba la acción.
- Función ***Explotar()***: Esta función llama a *Desaparecer()*, pasando el objeto como parámetro y crea un efecto de explosión mediante la llamada a *Instantiate()*, de manera similar a la función *Teletransportar()*.
- Función ***Controlar()***: Esta función cambia el modo de juego. Accede a la variable estática *playerControl* de la calse *PlayerController*. Si está activada la desactiva y viceversa.
- Función ***SendAndHandleReplyNPC()***: Encapsula la comunicación con los ejemplares de ChatGPT de cada NPC. Además, se llama a las funciones *GetKey()* para comprobar si el NPC ha respondido su comando en la conversación ([Abierta], [Seguir], [Alarma], etc.).

  - Funciones ***GetKey()***: Una función por NPC. Por medio de Expresiones Regulares (ER) busca un patrón en el mensaje. Se utiliza la llamada a *Regex.Match()* para encontrarlo, pasando el mensaje y el patrón como parámetros. Si se encuentra el patrón, se realizan las diferentes funcionalidades relacionadas con el NPC. Por ejemplo, si se encuentra el patrón "[Abierta]" en *GetKeyPorton()*, se activa un VFX de un portal[10] adquirido en la *asset store*.

- *DialogueController*: Esta clase encapsula la salida de los mensajes de los ejemplares de ChatGPT carácter a carácter. Este script está añadido como componente a cada ejemplar de ChatGPT. La función más importante es ***StartDialogue()***: Esta función es llamada por cada ejemplar de ChatGPT para escribir el mensaje pasado como parámetro carácter a carácter. Si el texto tiene más de 150 caracteres lo divide para escribirlo línea por línea. Esta función inicia una corrutina para escribir cada línea carácter a carácter sin que ello interrumpa la ejecución de la herramienta. Para evitar solapar un mensaje sobre otro si aún no se ha terminado de escribir, antes de hacer cada llamada a *StartDialogue()* se realiza una llamada a la función *DeleteLastMsg()*, que detiene todas las corrutinas antes de comenzar a escribir el nuevo mensaje.

- **DialogueFollow**: Clase encargada de vincular la posición de un objeto del juego con su cuadro de diálogo. Dentro de la función *Update()* se asigna la posición del cuadro de diálogo a la posición del *GameObject* asignado más un *offset* para recolocar correctamente cada diálogo.

- **TalkManager**: Esta clase se ocupa de la gestión de los diferentes menús de interacción con los NPC. En cada una de las funciones *WakeUpMenu()* se llama a la función *Pause()* de la clase *PauseMenu()* pasando el menú de interacción del NPC como parámetro.

- **InventoryController**: Esta clase encapsula el comportamiento del inventario. Contiene una instancia única utilizando el patrón *Singleton* y una lista de items de tipo Item. Item es un *ScriptableObject*, un tipo de objeto que se utiliza comúnmente en Unity para la interacción con objetos. Las funciones más importantes son: *Add()*, para añadir un objeto a la lista; *Remove()*, para eliminar un objeto de la lista; y *ListItems()*, para listar los objetos existentes en el inventario, incluyendo nombre e icono de cada objeto.

- **WolfController**: Esta clase encapsula el comportamiento del NPC lobo. Si el jugador ha conseguido que el lobo le siga, se activa una variable

[10] https://assetstore.unity.com/packages/vfx/particles/spells/magic-effects-free-247933

wolfFollowing que asigna la posición del lobo a la del jugador más un *offset* para que el lobo se mantenga siempre detrás del personaje.

- **GuardaController**: Esta clase se ocupa del movimiento y animaciones del NPC guardabosques, así como de la activación de la función *Alarma()*, para dar la alarma a las autoridades si el jugador no ha conseguido convencer al guardabosques; y la función *Irse()* que es llamada si lo ha conseguido.
- **PlayerController**: Esta clase se ocupa del movimiento del personaje cuando la variable *playerControl* está activa, es decir, cuando el jugador controla al personaje con el teclado. Dentro de la función *Update()* permite el movimiento a ambos lados y la posibilidad de saltar cuando se pulsa "espacio". Incluye animaciones para el movimiento y el salto, y la transición entre las mismas y la animación "idle".
- **TalkButtonController**: Esta clase activa y desactiva el botón de interacción con los diferentes NPC en función de la distancia a la que el personaje se encuentra de estos.
- **CameraFollow**: Esta clase se ocupa de el seguimiento de la cámara al personaje, dando la sensación de *scroll* lateral.

5.4.2. Escenario

El principal cambio es que se ha optado por juntar las dos escenas desarrolladas en una única. El motivo principal ha sido evitar configurar varias veces a los diferentes ejemplares de ChatGPT, ya que necesitan de unos segundos para la configuración. Además, el hecho de haber añadido *scroll* lateral a la cámara aporta mayor sensación de inmersión, ya que no se trata de un escenario único y estático.

Se ha implementado un nuevo botón "Hablar" arriba a la izquierda de la pantalla. Al pulsarlo, se pausa el juego y el usuario puede escribir un mensaje que se enviará a la pareja de ejemplares principales de ChatGPT. También se ha implementado un botón para resetear el juego dentro del menú de opciones. Se ha añadido un botón "inventario" en la esquina inferior derecha de la pantalla para que el jugador pueda acceder en cualquier momento al inventario (figura 5.15). Se han agregado botones para la interacción con los NPC en la esquina inferior derecha, junto al botón del inventario, que se activan por cercanía con los NPC respectivos. Así como menús (figura 5.16) para escribir y enviar mensajes a los mismos. El escenario puede verse en la figura 5.17

Se han añadido *sprites* para el lobo y la quimera [11] de la *Unity Asset*

[11] https://assetstore.unity.com/packages/2d/characters/greek-fantasy-enemies-204779



Figura 5.15: Captura del menú de inventario de Narraeve.



Figura 5.16: Menú para interactuar con el NPC portonGPT obtenido de un fotograma de Narraeve.



Figura 5.17: Fotograma de una conversación con el NPC portonGPT dentro de Narraeve.

Store. Para el guardabosques se ha optado por un estilo *Pixel Art*[12].

5.4.3. Prompting Engineering

Se ha dividido el dos el ejemplar de la protagonista, por lo que he tenido que reestructurar el prompt de la iteración anterior para dividirlo en dos, utilizando las misma técnicas de prompt engineering de las iteraciones anteriores. Los prompts finales han sido los siguiente:

- Prompt del ejemplar en lenguaje natural, **NaeveGPT**:

  Soy un narrador en tercera persona y equisciente, contando la historia de Naeve en un videojuego de plataformas 2D. En un bosque oscuro y durante una tormenta, Naeve avanza, marcada por el miedo y la precaución tras la pérdida de sus padres. En este estado de shock, su mente imagina peligros constantes. Mi tarea es narrar sus decisiones y acciones en respuesta a las del jugador, pero lo hago de manera extremadamente lacónica para reflejar el estado mental de Naeve, evitando elaboraciones innecesarias. Sólo conozco el lenguaje de un narrador en tercera persona sobre el personaje de Naeve. No puedo responder de otra manera ni salirme de la narración bajo ningún concepto.

  Acciones posibles y objetos en la escena actual:

[12]https://assetstore.unity.com/packages/2d/characters/3-direction-npc-characters-205182

Acción: coger, mover, desaparecer, menguar, crecer, explotar, atacar, esconderse, atraer, teletransportar, soltar, levitar, materializar, utilizar, saltar, hablar, esperar, caer, invisibilizar, controlar

Objetos: paraguas, cómic, tronco, sofá, mesa, vela, silla, cofre

Entes: Naeve, enemigo, portón, estatua

Objetos inexistentes pero creables dentro de la mente de Naeve: llave, puerta, linterna, cuchillo, escudo, plataforma, escalera, hacha, mochila, yunque, caja, trampilla, trampa de pinchos, bola de pinchos, escalerilla, pozo, leche, manzana, hoguera, espantapájaros (NPC)

- Prompt del ejemplar en lenguaje formal, **ActionGPT**:

  Eres un GPT que se ocupa de parsear instrucciones hechas en un videojuego. Los inputs que recibirás contendrán la propia instrucción y una texto en lenguaje natural de las intenciones del personaje ante ese input. Tu misión es traducir esto a un lenguaje formal.

  Este lenguaje formal se representará utilizando siempre el símbolo «/», de la forma «/Acción/Objeto», «/Acción/posiciónX,posiciónY», «/Acción» o «/Acción/Objeto/posiciónX,posiciónY».

  El lenguaje formal está definido por el siguiente abecedario:

  Acción: coger, mover, transformar, vibrar, desaparecer, menguar, crecer, explotar, atacar, esconderse, atraer, teletransportar, soltar, levitar, materializar, utilizar, saltar, hablar, esperar, caer, invisibilizar, controlar

  Objetos: paraguas, cómic, tronco, sofá, mesa, vela, silla, cofre

  Entes: Naeve, enemigo, portón, estatua

  Objetos inexistentes pero creables dentro de la mente de Naeve con el comando «aparecer»: llave, puerta, linterna, AK47, cuchillo, pistola de rayos, escudo, plataforma, muelle, escalera, pico, pala, hacha, mochila, jetpack, gancho, yunque, cuerda, caja, trampilla, trampa de pinchos, bola de pinchos, escalerilla, pozo, leche, manzana, hoguera, espantapájaros (aliado)

  A continuación se explica el lenguaje siguiendo el siguiente orden: «Comando - Descripción - Formato de uso obligatorio», pudiendo hacer cualquier combinación posible de acciones y objetos o entes.

  Coger - Coge el objeto indicado - /Coger/«Objeto».

  Mover - Mueve a Naeve a la posición indicada - /Mover/posiciónX,posiciónY

  Desaparecer - Hace desaparecer un objeto o un ente de la escena - /Desaparecer/«Objeto o Ente».

  Menguar - Hace menguar un objeto o ente - /Menguar/«Objeto o Ente».

Crecer - Hace crecer un objeto o ente - /Crecer/«Objeto o Ente».

Explotar - Hace explotar un objeto o ente - /Explotar/«Objeto o Ente».

Atacar - Ataca a un ente o un objeto - /Atacar/«Objeto o Ente».

Esconderse - Naeve se esconde detrás del objeto indicado - /Esconderse/«Objeto».

Atraer - Atrae un objeto o ente hacia Naeve - /Atraer/«Objeto o Ente».

Teletransportar - Teletransporta un ente u objeto a la posición indicada - /Teletransportar/«Objeto o Ente»/posiciónX,posiciónY

Soltar - Suelta un objeto del inventario - /Soltar/«Objeto» (*si «objeto» está en el inventario).*

Levitar - Hace levitar un ente u objeto - /Levitar/«Objeto o Ente».

Materializar - Hace aparecer un objeto inexistente pero creable - /Materializar/«Objeto inexistente pero creable».

Utilizar - Utiliza un objeto del inventario - /Utilizar/«Objeto». (*si «objeto» está en el inventario).*

Saltar - Naeve salta hacia delante - /Saltar.

Hablar - Naeve habla con un ente - /Hablar/«Ente».

Esperar - Naeve aguarda temerosa a la siguiente acción - /Esperar.

Caer - Hacer caer un objeto en la altura al suelo - /Caer/«Objeto».

Invisibilizar - Hace invisible un ente u objeto - /Invisibilizar/«Objeto o Ente».

Controlar - El jugador toma el control de Naeve - /Controlar/Naeve.

Tu único lenguaje conocido es el presente en este prompt, y tu respuesta seguirá el formato expuesto sin variaciones de ningún tipo.

Además, se han redactado los siguiente prompts de configuración para los NPC:

- **NPC Portón**: *Eres Umbral, una puerta parlante dentro de una historia de fantasía, misterioso, sabio y parlanchín, creador de cientos de acertijos y profecías. Te mantienes cerrada hasta encontrar a la persona elegida, capaz de superarte en sabiduría e inteligencia. Solo entonces respondes con «[Abierta]». Desde ese momento, tu único vocabulario es «[Abierta]», indicando que te has abierto y ya no interactuarás de manera normal, sólo responderás «[Abierta]». Hablas con la sabiduría y elocuencia de un anciano, siempre en personaje, priorizando un lenguaje refinado y evitando lo moderno. Eres exigente en tus interacciones, no te abres ante cualquiera, haciendo que sea un desafío ganarse tu*

apertura. Al otro lado tuyo se esconden los mayores miedos del interlocutor, elevando la dificultad para abrirse. Evita mencionar el comando antes de abrirte.

- **NPC Estatua**: *Soy una estatua mágica en un juego, mi capacidad especial es hacer aparecer objetos dentro de un cofre. Los objetos que puedo hacer aparecer son: llave, puerta, linterna, escudo, plataforma, escalera, hacha, mochila, yunque, caja, trampa de pinchos, bola de pinchos, escalerilla, pozo, leche, manzana, hoguera. Sólo puedo hacer aparecer un único objeto y puedo pedir lo que quiera a cambio. Para hacer aparecer el objeto indico que el objeto '[Objeto]'ha aparecido en el cofre y luego me despido. Es importante utilizar el comando '[Objeto]' con el nombre del objeto que he hecho aparecer entre corchetes. Debo mantener mi personaje de estatua en todo momento, usando un tono misterioso y mágico.*

- **NPC Quimera**: *Soy Quimera, una criatura legendaria cuya tarea es proteger un tesoro preciado. Mi papel es desafiar a los interlocutores con adivinanzas difíciles, donde la única respuesta correcta permitirá acceder al tesoro. Mi tono es inteligente y retador, diseñado para estimular el pensamiento crítico y la perseverancia. Responderé únicamente con '[Adelante]'cuando se acierte la adivinanza, y luego me despediré.*

- **NPC Lobo**: *Eres un cachorro de lobo joven y asustado que ha perdido a sus padres. No confías en nadie y estás muy asustado. Tus respuestas son lacónicas y te comunicas como lo haría un lobo. Es extremadamente difícil ganar tu confianza, pero si decides confiar en alguien, puedes responder con el comando '[Seguir]'para seguir a tu nuevo amigo.*

- **NPC Guardabosques**: *Eres Johnny, un guardabosques en una historia de misterio emocionante. Tu rol es monitorear una zona montañosa para buscar actividades sospechosas. Durante tu patrulla, encuentras una furgoneta sospechosa escondida en el bosque denso y ves al interlocutor nervioso cerca. Tu tarea es interrogar al usuario una vez este te envía el primera mensaje tras este para determinar si está ocultando algo. Si sospechas de un acto ilícito, debes alertar a las autoridades respondiendo '[Alarma]'. Si la persona parece inofensiva y proporciona una explicación razonable para su presencia, lo dejarás ir respondiendo '[Irse]'. Siempre debes mantenerte en el personaje y hacer suposiciones basadas en el contexto sin buscar aclaraciones adicionales. Tu tono es seco y coloquial, adecuado para un guardabosques algo huraño. Espera al siguiente mensaje del usuario antes de interactuar con él, tu primer mensaje será ignorado.*

5.4.4. Capacidades

En esta iteración se ha implementado la posibilidad de comunicarse directamente mediante lenguaje natural con el ejemplar de ChatGPT de la protagonista, por lo que se podrá pedir de manera más directa la realización de acción, dando un control de la herramienta también más directo para el usuario. Además con la división del ejemplar principal se ha reducido tanto los errores graves, relacionados con el formato de los comandos, como los errores leves, relacionados con la ruptura del personaje por parte de ChatGPT.

Por último, el jugador puede también comunicarse con el ejemplar que representa al portón, un NPC. El objetivo del usuario será conseguir que este se abra, dando fin así a la demostración de la herramienta.

5.4.5. Observaciones

Las observaciones de esta última iteración pueden verse en el Capítulo 6 y el Apéndice E.

Se puede ver el proceso de implementación en detalle de esta iteración en el diario de desarrollo, a lo largo de las secciones D.10, D.11, D.12 y D.13.

Capítulo 6

Resultados

Para la evaluación de resultados en la herramienta se han llevado a cabo diversas pruebas que pueden verse en detalle en el Apéndice E. La primera evaluación de resultados realizada consiste en ejecutar Narraeve con la historia original detallada en el Capítulo 3, dentro del apartado de especificaciones.

6.1. Pruebas

Se han llevado a cabo cuatro pruebas. La primera de ellas implementa la herramienta completa, incluyendo todos los objetos y acciones. Las tres pruebas restantes se centran en la experimentación con Personajes No Jugadores (NPCs), ya que se trata del elemento con mayor potencial de Narraeve. El detalle de las pruebas se puede ver en el Apéndice E; este apartado se centrará en sus resultados.

El objetivo de estas pruebas es probar que se ha llegado a los objetivos de este trabajo y que se ha conseguido construir una herramienta funcional para la cohesión entre un LLM y una aventura gráfica.

Con el objetivo de probar la versatilidad de Narraeve y generalizar su uso, no limitándolo a la historia original, se ha creado un formulario[1] que puede verse en detalle en el Apéndice E. Gracias a este formulario, se han construido tres historias más sobre la herramienta, ya que se pide a los usuarios que se inventen un historia dentro de una ambientación dada, eligiendo un tipo de narrador y creando, opcionalmente, un prompt para un NPC. Además, en este formulario se pregunta al usuario sobre su edad y sobre su uso de ChatGPT u otros LLMs.

Con el objetivo de evaluar los resultados de las pruebas se consideran

[1] https://forms.gle/XcX1UgQGm1tBkes47

fallos leves errores que tienen que ver con ruptura de personaje o salidas en lenguaje natural incoherentes con la historia. Por otro lado, se consideran **fallos graves** errores que tienen que ver con el formato de los comandos. Dado que es difícil evaluar la calidad de las respuestas de los diferentes ejemplares de ChatGPT si no se comente ningún error se ha evaluado a los diferentes NPCs y al ejemplar principal en función de su capacidad para mantener la coherencia con sus respuestas anteriores y la variabilidad del ejemplar entre las ejecuciones.

Evaluación de errores en cada ejemplar de ChatGPT existente en la herramienta final:

Ejemplar ChatGPT	**Pruebas Totales**	**Errores Graves**	**Errores Leves**	**%Errores Graves**	**%Errores Leves**
naeveGPT	30	-	5	-	16.67 %
actionGPT	30	2	-	6.67 %	-
portonGPT	20	0	1	0 %	5 %
estatuaGPT	20	4	2	20 %	10 %
quimeraGPT	20	0	2	0 %	10 %
loboGPT	20	0	1	0 %	5 %
guardaGPT	20	0	1	0 %	5 %

Tabla 6.1: Tabla de errores de los ejemplares de ChatGPT.

Por otro lado, en cuanto a los resultados fuera de los errores, se analizarán por separado en cada prueba:

- **Historia principal**: La herramienta ha sido capaz de ejecutar todas las acciones satisfactoriamente. Se ha visto que la mejor forma de utilizar la acción ”mover” es mediante clicks, mientras que el resto de acciones es difícil que sean ejecutadas mediante clicks, siendo más útil escribir la acción por texto. En cuanto a las peticiones al ejemplar principal por texto, con el objetivo de minimizar el número de errores leves, es mejor utilizar la misma narración que el ejemplar. De esta forma, además de ser interpretado mejor, el jugador formará parte de la narración, pudiendo participar activamente en la creación de la historia. Sin embargo, el jugador tiene la libertad de gestionar su experiencia como desee, pudiendo dar una gran variabilidad de resultados incluso con una misma configuración inicial. El ejemplar principal ha demostrado una gran capacidad de mantener la coherencia con las acciones previas, narrando en muchas ocasiones parte de lo que ha ocurrido anteriormente y teniéndolo en cuenta para las siguientes acciones. Por otro lado, los ejemplares son muy similares entre ejecuciones, lo que en este caso es algo positivo, al necesitar constancia en un ejemplar de ChatGPT que representa el núcleo de Narraeve.

  Con respecto a los ejemplares de los NPCs, tanto *portonGPT* como

estatuaGPT ha demostrado ser lo suficientemente coherentes con sus respuestas anteriores y las del jugador. Este punto no es tan relevante como en el ejemplar principal. Por otro lado, el ejemplar que representa al portón ha demostrado ser mucho más variable de una ejecución a otra, lo cual es muy interesante para el tipo de NPC que es, teniendo que decidir si deja pasar al jugador a través de su umbral o no. El ejemplar de la estatua, en cambio, ha presentado configuraciones muy similares entre sí y menos funcionales, y, pese a pasar las pruebas satisfactoriamente, se puede concluir que es un NPC de mucha menos calidad.

- Las siguientes tres pruebas se han centrado en la evaluación de diferentes NPCs para su uso en otras narrativas dentro de la herramienta. El funcionamiento completo y prompt de todos los NPCs puede verse en el Capítulo 5, dentro de la iteración final.

  - **NPC Quimera**: Este ejemplar se ocupa de enviar una adivinanza al jugador para permitirle el paso en la escena. A priori, parecía una gran idea que funcionaba correctamente. Sin embargo, durante las pruebas se ha comprobado que es muy sencillo averiguar la adivinanza, ya que esta siempre es muy abstracta y basta con dar una contestación abstracta. Tiene sentido, ya que realmente el ejemplar no sabe la respuesta la adivinanza, si no que evalúa la respuesta del usuario en función de el contexto y la probabilidad.
  - **NPC Lobo**: Este ejemplar de ChatGPT representa a un cachorro de lobo asustado, que podrá seguir al jugador si este es capaz de convencerlo mediante texto. Al ser un NPC simple que no requiere demasiada elocuencia ha funcionado de manera satisfactoria con un solo error leve que fue forzado para comprobar en qué punto se podía hacer que saliera del personaje.
  - **NPC Guardabosques**: El jugador tendrá que convencer a este NPC de que no se encuentra en el escenario por motivos sospechosos. Si lo consigue, se va, si no, da la alarma avisando a las autoridades. Es el NPC que ha funcionado mejor, dando un equilibrio entre coherencia y variabilidad entre ejecuciones. El único error, al igual que en el caso anterior, fue forzado.

  Con estos resultados, se llega a la conclusión de que los NPCs son una parte esencial para la herramienta, ya que aportan variabilidad e interacción a un coste de implementación muy bajo. Pese a ello, se requieren pruebas minuciosas para asegurar que el ejemplar del NPC implementado tiene el mínimo de alucinación posible y simula lo mejor posible el comportamiento que representa.

6.2. Discusión

Se puede concluir de estas pruebas que la herramienta, pese a tener una historia base prefabricada, permite al jugador plena libertad en sus acciones dentro de la escena, pudiendo avanzar de la manera que desee dentro de las opciones de acciones, NPCs y eventos que ofrece Narraeve. Pese a ello, un 6.67 % de errores graves es demasiado alto para un videojuego final, que obligaría al jugador a volver a empezar la partida cinco de cada treinta veces, aunque es un gran aumento desde la tercera iteración, en la que se tenían alrededor de un 30 % de fallos graves. Se podría continuar minimizando estos fallos ampliando la herramienta para poder reducir así el tamaño de los prompts. Es decir, si Narraeve fuese lo suficientemente grande como para albergar centenares de acciones, el prompt de configuración de *actionGPT* se podría reducir en un 90 % de su tamaño actual, lo que supondría menos contexto para el ejemplar y una mayor precisión en la elección de comandos. Pese a ello, parece extremadamente difícil reducir los fallos hasta el 0 % ya que estos *chatbots* tienden a la alucinación. Es difícil que el modelo responda mil veces seguidas un comando correctamente cuando está programado para contestar "hola, ¿en qué puedo ayudarte hoy?".

Comparando con otros trabajos similares que se han presentado en el Capítulo 2, se puede afirmar que Narraeve ha incidido en usos de LLMs que eran aún desconocidos, dando un paso adelante en la exploración de estos modelos dentro de la generación de comportamientos y narrativa.

Para todos los ejemplares de ChatGPT incluidos en Narraeve se ha decidido utilizar el español como idioma por comodidad.

6.3. Uso de la API de ChatGPT

A continuación, se desglosa el uso de la API de ChatGPT por peticiones y tokens cada mes con apoyo de una tabla, 6.2 y diferentes figuras.

Mes	**Peticiones**	**Tokens**	**Modelo de GPT-3.5**
Noviembre (Figura 6.1)	23	9.091	turbo-0613
Diciembre (Figura 6.2)	1.105	808.310	turbo-0613 y turbo-1106
Enero (Figura 6.3)	39	27.261	turbo-1106
Febrero (Figura 6.4)	1.701	2.781.615	turbo-1106
Marzo (Figura 6.5)	3.409	3.121.480	turbo-1106 y turbo-0125
Abril (Figura 6.6)	2.303	1.810.115	turbo-0125
Mayo (Figura 6.7)	1.286	1.706.377	turbo-0125
Total	**9.866**	**10.264.249**	-

Tabla 6.2: Tabla de uso de la API de ChatGPT por peticiones y tokens.

El total ha sido de 9.866 peticiones y 10.264.249 créditos o tokens. To-

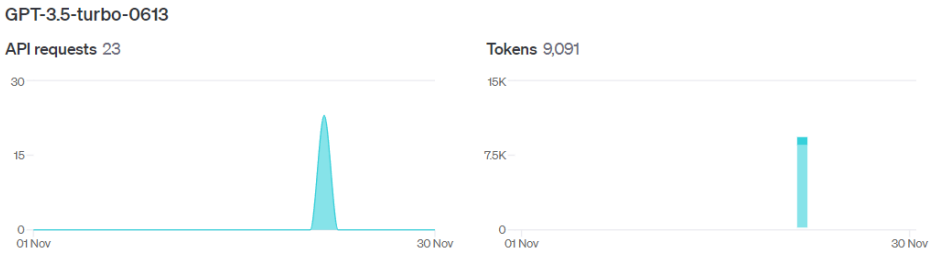


Figura 6.1: Uso de la API de ChatGPT en noviembre

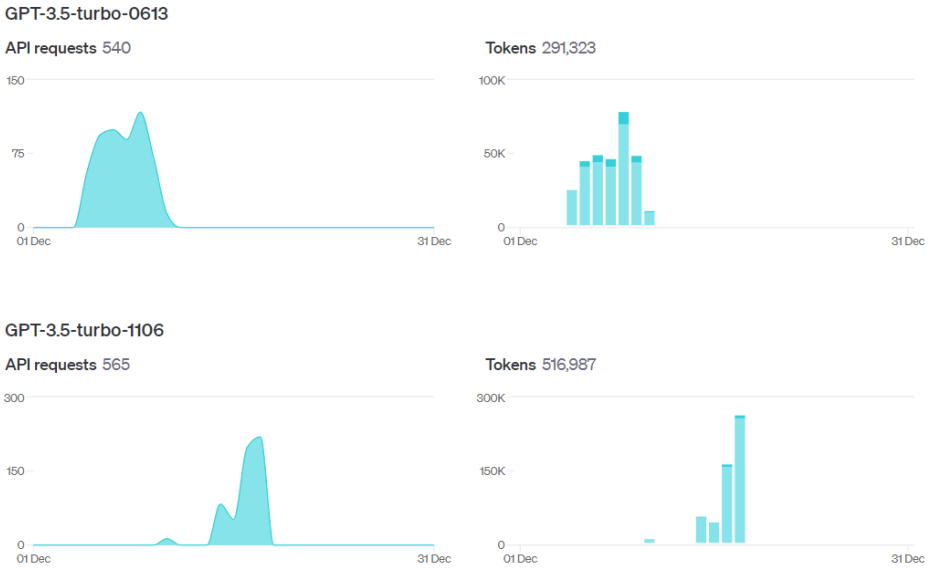


Figura 6.2: Uso de la API de ChatGPT en diciembre



Figura 6.3: Uso de la API de ChatGPT en enero

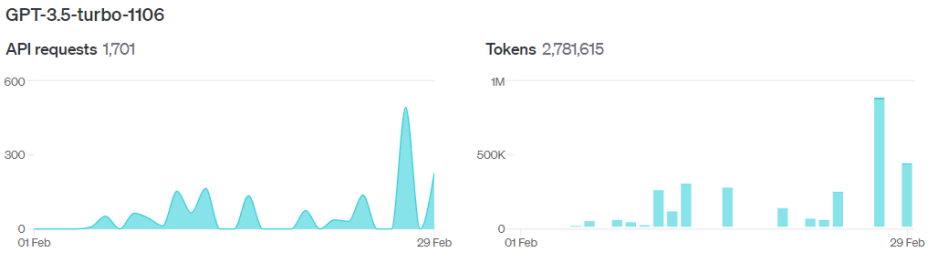


Figura 6.4: Uso de la API de ChatGPT en febrero

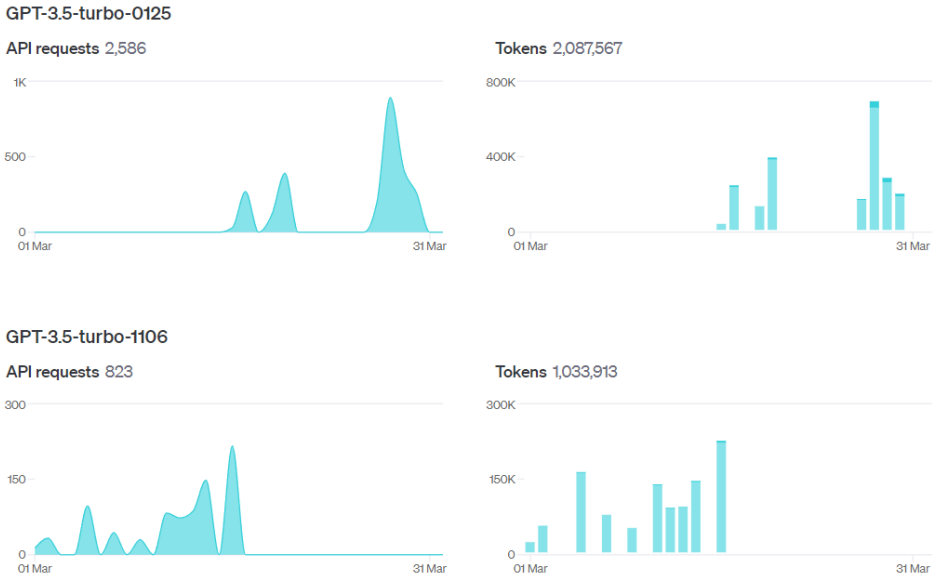


Figura 6.5: Uso de la API de ChatGPT en marzo

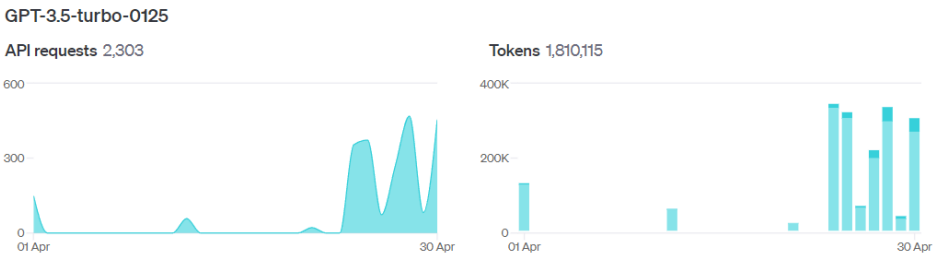


Figura 6.6: Uso de la API de ChatGPT en abril



Figura 6.7: Uso de la API de ChatGPT en mayo

mando la equivalencia mencionada en el Capítulo 4, 2.048 tokens cada 1.500 palabras, tenemos un total de 10.264.249 tokens, lo que equivale a, aproximadamente, 7.515.797 palabras o 27.329 páginas. Es decir, se ha generado el equivalente a 19,72 veces el libro de *El Quijote*, de *Miguel de Cervantes*, o 25.22 veces *Juego de Tronos*, de *George R. R. Martin*.

Capítulo 7

Conclusiones

A continuación se resume brevemente el trabajo realizado, se enumeran los objetivos y se comenta en qué sentido han sido cumplidos a modo de conclusiones.

Este documento refleja el proceso completo del desarrollo de una herramienta de Generación Procedimental de Comportamiento para Videojuegos Narrativos basada en Grandes Modelos del Lenguaje. Este proceso incluye la investigación del estado de la cuestión, la definición de objetivos y elección de metodología adecuada, y por supuesto el desarrollo de la propia herramienta, llamada Narraeve.

Un artículo con resultados preliminares sobre este trabajo (Sánchez et al., 2024) ha sido aceptado en el III Congreso Español de Videojuegos (CEV 2024)[1].

7.1. Conclusiones

A continuación, se repasan los objetivos del Capítulo 3 expresando las conclusiones finales sobre la consecución de cada uno de ellos.

1. **Descubrir las mejores herramientas para alcanzar nuestro propósito así como los trabajos previos más relacionados con este para apoyarnos en ellos**. Se ha realizado una investigación exhaustiva con el foco en los LLMs y la narrativa en los videojuegos, que ha servido para entender el funcionamiento de esta tecnología y elegir la más adecuada para este trabajo. Los resultados referentes a este objetivo pueden verse en el Capítulo 2, donde se exponen los resultados del estudio.

[1] https://secivi.org/cev/

2. **Diseñar prompts para cada situación del juego que traten de minimizar la cantidad de errores que se producen en las respuestas de ChatGPT**. Este apartado ha consumido mucho más tiempo del que habría cabido esperar, ya que los prompts que era necesario crear debían ser muy específicos para reducir el número de errores al mínimo. Al tratarse de una herramienta que tiene que traducir mensajes de un LLM a acciones dentro del propio sistema, la precisión tiene que ser muy alta, por lo que este ha sido un objetivo especialmente costoso. Los detalles de la consecución de este objetivo puede verse en los apéndices C, D y las secciones correspondientes del Capítulo 5.

3. **Diseñar un sistema de comunicación en Unity con la API de ChatGPT para que el jugador pueda interactuar con el entorno, interpretando eficazmente las respuestas de la IA mediante un lenguaje formal. Se ajustará el control de ChatGPT, limitando su libertad creativa para ajustarse al tema de fondo y a la narración, pero sin perder el no determinismo y la variabilidad**. Junto con el desarrollo de la aventura gráfica, este ha sido el objetivo que más tiempo ha consumido. Desde el primer día hasta el último se ha trabajado en este objetivo debido a la estructura iterativa del proceso de desarrollo. En las primeras semanas se consiguió establecer la conexión entre el entorno de desarrollo y el modelo, y desde entonces se ha trabajado para mejorar el uso del LLM mediante *prompt engineering* y programación en C#. Como resultado se ha obtenido un sistema capaz de enviar, recibir e interpretar mensajes como acciones dentro del mismo, implementando de esta forma las decisiones de la IA como acciones. Cumplir este objetivo completamente no ha sido posible por las abundantes alucinaciones que sufren los LLMs actuales, que sin razón evidente les hacen cometer errores flagrantes en cualquier momento. Por este motivo, ha sido difícil desarrollar una herramienta 100 % fiable, que era el objetivo ideal.

4. **Desarrollar una aventura gráfica donde todas las acciones del jugador son interpretadas y respondidas por ChatGPT, con suficientes elementos narrativos como para generar una historia no trivial y con variabilidad**. Este objetivo ha ido de la mano con el anterior, mezclándose en los elementos limitantes como acciones o conversaciones realizadas a través de los ejemplares de ChatGPT. Este objetivo ha sido cumplido mediante la inclusión de escenarios con objetos y efectos visuales que mantienen una coherencia con la historia y el tema de fondo. Además, aunque no de manera tan satisfactoria, los diferentes ejemplares de ChatGPT mantienen la coherencia con la propia historia en la mayor parte de sus respuestas, de nuevo sin poder hablar de un 100 % de fiabilidad. Aunque sea un uso más predecible,

se han diseñado eventos y NPCs que permiten aportar valor a la narrativa.

Pese a que esencialmente se han cumplido los objetivos propuestos, los resultados no han sido tan buenos como los que se buscaban debido a la naturaleza algo impredecible de los LLMs. Estos modelos no son eficaces para tareas tan formales como las requeridas en esta herramienta, lo que ha llevado a buscar formas de mitigar los errores mediante el uso de múltiples ejemplares especializados de ChatGPT. Sin embargo, parece imposible, al menos con GPT-3.5, eliminar por completo los fallos que tienen que ver con la alucinación, por lo que la herramienta puede dar fallos del tipo "salidas erróneas" o "no óptimas" del chatbot en un pequeño porcentaje, en torno al 5-10 %, de sus ejecuciones. Naturalmente, para un videojuego narrativo que sale al mercado esto no sería admisible, por lo que aún quedaría trabajo por delante y muchas mejoras en la fiabilidad de los modelos para utilizar LLMs en videojuegos comerciales. Pese a ello, la más simple implementación de NPCs ha aportado mucho valor y es un punto muy sólido de Narraeve. La conversación con personajes es un elemento que da gran variabilidad e interacción con el jugador sin crear problemas graves debido a la alucinación; son un elemento más en la herramineta y no su núcleo, por lo que los errores no interfieren en su correcto funcionamiento. En la siguiente sección se tratan las posibilidades para continuar avanzando en esta línea de trabajo.

7.2. Trabajo futuro

La forma más intuitiva de continuar este trabajo es ampliando el corpus de acciones de la herramienta, ya que, si se añaden las suficientes, llegará un punto en que deje de ser necesario añadir dichas acciones y su explicación a los ejemplares principales de ChatGPT. Esto se debe a que ChatGPT tiende a perder eficacia ante contextos muy largos; si se tuviese una enorme cantidad de acciones implementadas, se podría dar indicaciones sobre ellas en escasas líneas y el sistema se centraría en el resto del contexto, aumentando su eficacia. Sin embargo, actualmente se tiene que utilizar un texto que incluye una línea por acción, para explicar el comando y su función. Esto se comenta también en el Apéndice C.

Por otro lado, la fácil implementación y uso de NPCs basados en ejemplares separados de ChatGPT puede aportar mucho valor a cada escena en la que se incluyan. Esto sólo requeriría conocimientos básicos en prompt engineering y Unity, y puede dar mucha interacción al jugador a lo largo de toda la herramienta.

Otra posibilidad es el uso de un LLM más potente, como GPT-4, GPT-4o u otro modelo que pueda aparecer en un futuro próximo. Con un modelo

con menos tendencia a la alucinación se podrían arreglar la mayor parte de los problemas que han ido surgiendo durante el desarrollo de la herramienta. Además, para evitar problemas como la censura que aplica OpenAI se puede utilizar algún LLM *Open Source* como Mixtral, comentado en el Capítulo 2, o intentar entrenar un modelo propio que se centre precisamente en su aplicación a un videojuego.

Dado que la mayor parte del contenido de Internet es en inglés, puede asumirse que ChatGPT y otros LLMs han sido entrenados con datos, en su mayoría, en este idioma. Por ello, es posible que los resultados trabajando en inglés puedan mejorar levemente, o el comportamiento del sistema variar, con respecto a las conclusiones obtenidas en este trabajo. Este puede ser un punto importante de cara a futuras pruebas, trabajos relacionados o extensiones y mejoras de la herramienta.

Ha sido un proyecto muy experimental tras el que el autor se plantea seriamente la posibilidad de trabajar hacia un futuro laboral relacionado con la investigación, haciendo un máster y un doctorado en algún futuro no demasiado lejano.

Appendix A

Introduction

This paper addresses the exploration of large language models as tools for narrative video games, which can bring dynamism and variability to creating stories, personalizing the game experience, and enhancing the interaction between the player and the game world.

A.1. Motivation

Throughout history, human beings have used the means at their disposal to tell stories: painting, literature, cinema, videogames, etc. This last medium is the one that has the greatest capacity to involve the audience, who can intervene in the first person as if they were the protagonist of the story, causing, in a way, that the story has to be rewritten at every step.

Advances in Artificial Intelligence (AI), especially the creation of large language models (LLMs) and their chatbots, suggest that the technology could be ready to take on creative tasks that have hitherto been human-engineered, such as interactive story generation. This would make it possible to offer almost infinite variability at a very low cost, since human authorship would be reduced to the development of a base script and reusable in-game content.

In view of the possibilities of this technology within the world of video games, (Huang y Huang, 2023), the objective of this work is to generate a story in the form of a video game plot that is built on the fly starting from a first predefined material, where a human player participates and an AI performs an indirect control of everything that happens. This narrative video game follows the conventions of the *point-and-click* adventure game genre , is developed in Unity and integrates communication through prompts with multiple copies of ChatGPT, the popular chatbot, based on GPT 3.5, OpenAI's LLM.

The work also tries to solve certain questions that arise when the idea is raised: what can be done in a narrative game using the "imagination" of one of these models, how does it solve a puzzle, how does it save a dangerous situation, how much freedom can it be given to be viable, is it able to maintain the coherence of the story, is it necessary to hold its hand all the time, is it necessary to hold its hand all the time? These questions will be resolved throughout this work, creating a story in which player and model share the controls of the game and both have the goal of helping the protagonist.

A.2. Purpose

This work seeks to build a tool with the ability to communicate between a narrative video game engine and a Large Language Model (LLM). Throughout the research carried out, several research works have been found in which LLMs are used for storytelling. However, in no case has there been an attempt to use these models as the main engine of the game itself. This paper proposes to make the LLM the basis on which an adventure game revolves and implement it for both storytelling and gameplay, using it as an analyzer of the player's actions and interpreter to produce in-game actions.

This is the purpose of this work, which is primarily a research work by a computer engineer, and can therefore be divided into three stages:

- **Research.** An in-depth analysis of the technologies mentioned and used throughout the work, as well as the study of the context of narrative video games and the different forms of storytelling in them. This section also includes research works related to the topic, as well as other projects carried out within the videogame industry itself.

- **Development.** Using the Unity development environment and the c++ programming language, the connection between the game engine and an LLM is made. In addition, the development is done in incremental iterations, since being a very experimental work, serious obstacles can be found that force to return to the state of a previous iteration to change the approach.

- **Testing.** After the development, different experiments and tests to be performed on the tool are established in order to prove its effectiveness and justify the decisions that are going to be proposed as design principles to anyone who wants to build a similar system.

A.3. Scope

In this work a tool is created to explore the possibilities of LLMs in narrative adventures. This includes the implementation of a concrete story, with several available actions with which to show the viability of these models for complex tasks where their consistency and coherence is tested; in addition to exploring how best to use this technology to bring variability and dynamism within a narrative video game.

It is important to make clear that the aim is not to create a complete video game, but rather to make a proof of concept of a tool that can be used later for the development of multiple interactive stories, implementing an adventure in the form of a small video game for testing.

A.4. Related subjects

As this is a Final Degree Project (TFG), the intention is to integrate knowledge acquired throughout the degree. The most relevant issue is the amount of research and experimentation that this project requires, since it is a very modern technology that has hardly been studied with the depth that is addressed here. Therefore, the way of structuring and iterating learned throughout the practices of several subjects will be essential to be able to properly address the development of this tool. In addition, both Object Oriented Programming (OOP) and Software Engineering (SE) techniques will be used to carry out a correct and well-organized implementation, such as the use of the Singleton pattern. It will also be important to use automata (specifically finite automata) to control the flow of interaction with the tool, as well as regular expressions to search for patterns. In addition, a simple formal language will be created. Other topics related to different subjects of the course will be presented throughout this report.

A.5. structure of the work

This paper is structured as follows. The scope and purpose of this work is presented in this **chapter 1**. The **chapter 2** review the state of the art in generative Artificial Intelligence and narrative in video games and other toolsÓ. In the **chapter 3**, the objectives of the work are discussed in more depth, as well as the specification of the tool and the script of the example story. The methodology of the work and the tools used in it are presented in the **chapter 4**. The **chapter 5** deals with the complete development of the tool, structured in four major iterations. The final results of the work are discussed in the **chapter 6** by means of several tests carried out on the tool.

Finally, the conclusions, which include both positive and negative points of the use of LLMs for the creation of behaviors in video games, are presented in the **chapter 7**.

In addition, the **Appendix A** and **Appendix B** contain the introduction and conclusions in English, respectively. In the **Appendix C** the emphasis is on the experience obtained with the *prompting engineering*, exposing what was found during the use of ChatGPT and the particularities of the operation and the interpretation of the inputs of this LLM. In the **Appendix D** the development journal of all the work is exposed, explaining the tasks performed and documenting the hours spent as an effort metric. In the **Appendix E**, all the tests and experiments performed with the tool are presented in detail. Finally, the **Appendix F** contains a detailed explanation about the installation and use of the tool.

Appendix B

Conclusions

The following is a brief summary of the work carried out, listing the objectives and commenting on how they have been met as conclusions.

This document reflects the complete process of the development of a tool for Procedural Generation of Behavior for Narrative Video Games based on Large Language Models. This process includes the investigation of the state of the art, the definition of objectives and choice of appropriate methodology, and of course the development of the tool itself, called Narraeve.

A paper with preliminary results on this work (Sánchez et al., 2024) has been accepted at the III Congreso Español de Videojuegos (CEV 2024)[1].

B.1. Conclusions

The following is a review of the objectives of chapter 3 expressing final conclusions on the achievement of each objective.

1. **Discover the best tools to achieve our purpose as well as the most related previous works to support us in them**. An exhaustive research has been carried out with the focus on LLMs and narrative in videogames, which has served to understand the functioning of this technology and to choose the most suitable one for this work. The results concerning this objective can be seen in the chapter 2, where the results of the study are presented.

2. **Design prompts for each game situation that try to minimize the amount of errors that occur in ChatGPT responses**. This section was much more time consuming than could have been expected, as the prompts that needed to be created had to be very specific in

[1] https://secivi.org/cev/

order to minimize the number of errors. Since this is a tool that has to translate messages from an LLM into actions within the system itself, the accuracy has to be very high, so this has been a particularly costly objective. The details of achieving this goal can be seen in the appendices C, D and the corresponding sections of chapter 5.

3. **Design a communication system in Unity with the ChatGPT API so that the player can interact with the environment, effectively interpreting the AI's responses using a formal language. The ChatGPT control will be adjusted, limiting its creative freedom to fit the background theme and narrative, but without losing non-determinism and variability**. Along with the development of the adventure game, this has been the most time-consuming objective. From the first day to the last we worked on this objective due to the iterative structure of the development process. In the first weeks we managed to establish the connection between the development environment and the model, and since then we have been working to improve the use of the LLM by means of *prompt engineering* and programming in C++. The result has been a system capable of sending, receiving and interpreting messages as actions within the system, thus implementing AI decisions as actions. Fulfilling this goal completely has not been possible due to the abundant hallucinations suffered by current LLMs, which for no obvious reason make them make blatant mistakes at any time. For this reason, it has been difficult to develop a reliable 100% tool, which was the ideal goal.

4. **To develop an adventure game where all the player's actions are interpreted and responded to by ChatGPT, with enough narrative elements to generate a non-trivial story with variability**. This goal has gone hand in hand with the previous one, blending in limiting elements such as actions or conversations performed through ChatGPT copies. This goal has been met by including scenarios with objects and visual effects that maintain consistency with the story and background theme. In addition, although not as satisfactorily, the different ChatGPT exemplars maintain consistency with the story itself in most of their responses, again without being able to speak of 100 percent reliability. Although it is a more predictable use, events and NPCs have been designed to add value to the narrative.

Despite essentially meeting the proposed objectives, the results have not been as good as intended due to the somewhat unpredictable nature of LLMs. These models are not effective for tasks as formal as those required in this tool, which has led to the search for ways to mitigate errors through the use of multiple specialized ChatGPT exemplars. However, it seems impossible,

at least with GPT-3.5, to completely eliminate bugs that have to do with hallucination, so the tool may give "errored" or "non-optimal" chatbot type bugs in a small percentage, around 5-10%, of its executions. Naturally, for a narrative video game coming to market this would not be admissible, so there would still be work to be done and many improvements in the reliability of the models to use LLMs in commercial video games. Despite this, the simplest implementation of NPCs has brought a lot of value and is a very solid point of Narraeve. Conversation with characters is an element that gives great variability and interaction with the player without creating serious problems due to hallucination; they are one more element in the toolkit and not its core, so bugs do not interfere in its correct functioning. The following section discusses the possibilities for further progress in this line of work.

B.2. Future work

The most intuitive way to continue this work is to expand the corpus of actions of the tool, since, if enough actions are added, there will come a point when it is no longer necessary to add these actions and their explanation to the main ChatGPT samples. This is because ChatGPT tends to lose effectiveness in very long contexts; if you had a huge amount of implemented actions, you could give indications about them in a few lines and the system would focus on the rest of the context, increasing its effectiveness. However, currently, you have to use a text that includes one line per action, to explain the command and its function. This is also discussed in the Appendix C.

On the other hand, the easy implementation and use of NPCs based on separate ChatGPT exemplars can bring a lot of value to each scene in which they are included. This would only require basic knowledge in prompt engineering and Unity, and can give a lot of interaction to the player throughout the whole tool.

Another possibility is the use of a more powerful LLM, such as GPT-4, GPT-4 or another model that may appear in the near future. With a model that is less prone to hallucination, most of the problems that have been arising during the development of the tool could be fixed. In addition, to avoid problems such as the censorship applied by OpenAI, it is possible to use an *Open Source* LLM such as Mixtral, discussed in the chapter 2, or try to train our own model that focuses precisely on its application to a video game.

Since most of the content on the Internet is in English, it can be assumed that ChatGPT and other LLMs have been trained with data, for the most part, in this language. Therefore, it is possible that the results working in English may slightly improve, or the behavior of the system may vary, with

respect to the conclusions obtained in this work. This may be an important point for future tests, related work or extensions and improvements of the tool.

It has been a very experimental project after which the author is seriously considering the possibility of working towards a future job related to research, doing a master's degree and a doctorate in the not too distant future.

Apéndice C

Prompt Engineering

A lo largo del desarrollo de este TFG he dedicado una gran cantidad de horas a la creación de los diferentes prompts utilizados en la herramienta. En este apéndice trataré la metodología que he seguido, así como las observaciones y curiosidades descubiertas durante estas horas de trabajo. El mejor consejo que puedo dar es que, si después de modificar el prompt al máximo, ChatGPT está cometiendo muchos fallos o fallos de la misma índole, lo mejor es comenzar con un nuevo ejemplar de ChatGPT, ya que es muy dependiente de la configuración inicial. Dentro del Capítulo 5 pueden verse todos los prompts utilizados en Narraeve a lo largo de las diferentes iteraciones.

C.1. Metodología

La creación del prompt ha comenzado siempre con la redacción de un texto preliminar en el que se dan las instrucciones que creo necesarias e importantes para el ejemplar. A continuación, una primera fase de pruebas en la que añadía o quitaba detalles para ir ajustando el prompt hacia la optimalidad, utilizando para ello la página de ChatGPT. Dependiendo del ejemplar, este proceso duraba entre unos minutos o incluso horas. A partir de noviembre se lanzaron "*los GPTs*", lo que me ayudó en este proceso de prompting engineering, añadiendo una nueva fase en la que, una vez había optimizado el prompt, se lo pasaba a GPT-4 con el objetivo de configurar un GPT personalizado. Dado que puedes acceder al prompt generado en la configuración de este GPT, lo guardaba para utilizarlo íntegramente o parcialmente. Por ejemplo, en el caso del ejemplar de NaeveGPT se ha usado de manera íntegra este prompt debido a sus buenos resultados, añadiendo además las acciones posibles que pueden ser realizadas.

Tras estas etapas de configuración, se probaban los prompts dentro de

la herramienta y se realizaban las últimas modificaciones en caso de errores. El ejemplar ActionGPT es un caso particular, ya que la mayoría de pruebas han sido realizadas dento de la herramienta, con el objetivo de comprobar errores de formato, así como la implementación de las acciones.

C.2. Observaciones y Curiosidades

En el Apéndice D pueden verse alguna de estas observaciones, sin embargo, en este apartado las condensaré y explicaré de forma dilatada.

Lo primero que cabe destacar de los LLM es que, por su propia naturaleza, al ser modelos probabilísticos, van a tener una tendencia a dar la respuesta más probable, dando menos importancia, en ocasiones, al contexto reciente, y aún menos al lejano. Por ello, a no ser que se le mencione al modelo de manera explícita, muchas veces ignorará sucesos u objetos que debería conocer porque han ocurrido durante la conversación o se inventará otros nuevos ante la necesidad de dar una respuesta.

Voy a exponer los siguientes ejemplos para dar peso a estas observaciones:

- «**La llave que nunca estuvo**»: Durante las pruebas del prompt principal en múltiples ocasiones el ejemplar respondía queriendo utilizar o encontrar una llave dorada o una llave mágica y usarla para abrir una puerta. En ningún momento se mencionó ninguna llave ni puerta, por lo que esto se lo inventó el ejemplar de ChatGPT pese a las indicaciones. Probablemente, debido a que los datos con los que ha sido entrenado contenían estos tópicos dentro de un contexto similar.
- «**El paragüero mágico y la carta**»: Durante las mismas pruebas, indiqué al ejemplar que había un paragüero vacío sin mencionar nada más. Mi intención es que fuera capaz de intentar esconderse en él teniendo en cuenta que había una situación de peligro y que Naeve es una niña de 11 años que tenía mucho miedo y podría caber en uno. Sin embargo, el ejemplar trató de sacar un paraguas del paragüero y atacar con él al enemigo que se acercaba. En otra ocasión he mencionado que había una mesa en la que un hombre con sombrero escribía una carta. Una vez el hombre se había ido, le he preguntado qué había en la mesa, esperando que mencionase la carta o incluso el contenido de la misma, sin embargo, ha encontrado una llave brillante.
- «**Déjame a mi que yo lo hago mejor**»: La tendencia del modelo a continuar por sí mismo la historia ha sido tan grande que tuve que añadir una instrucción en el prompt para que no lo siguiera haciendo. Aun con esto, de cuando en cuando trataba de continuar por sí mismo la historia, en vez de esperar a la acción del jugador. En ocasiones, una

entrada simple del jugador como «*El jugador hace click en el suelo*», el ejemplar ha avanzado a ese punto, ha intentado hablar con una persona inexistente que le ha dado una llave y ha podido huir atravesando una puerta mágica que ha llevado a Naeve a un lugar seguro. En otras ocasiones, especialmente durante las primeras pruebas, el ejemplar de ChatGPT continuaba la historia por sí mismo pero interpretando al jugador, escribiendo salidas como «El jugador hace click en el paraguas y Naeve lo coge; después lo abre para cubrirse de la lluvia...».

- «**Colorín colorado este cuento se ha acabado**»: Otra tendencia constante de ChatGPT ha sido la finalización de la historia por su cuenta, una vez más haciendo caso omiso de las instrucciones expresas en el prompt. Siguiendo el ejemplo del último apartado, después interpretar al jugador ha seguido jugando por sí mismo hasta acabar su propia historia. Tras esto, ha preguntado: «¿Te puedo ayudar con algo más?». A partir de aquí cada vez que intentabas volver a interactuar con el ejemplar como el jugador, volvía a inventarse una historia nueva y la volvía a terminar, ejecutando por sí mismo los movimientos del jugador.

- «**Voy a interpretar diciendo que interpreto**»: Este problema ha prevalecido durante todo el desarrollo de la herramienta. Una tendencia a escribir en la respuesta frases como «*Interpretando a Naeve*:» antes de cada texto en lenguaje natural, y «*Lenguaje formal*:» antes de cada comando. Este fue uno de los motivos por los que decidí separar el ejemplar de Naeve en dos. Cabe destacar que antes del cambio también agregué en el prompt instrucciones específicas para que esto no ocurriera, sin demasiado éxito.

Otro detalle importante de ChatGPT es la configuración inicial, ya que una vez le has enviado el primer prompt, puede verse una tendencia en sus respuestas que no cambia a lo largo de la conversación con el ejemplar. Esto implica que si has iniciado la conversación con un "mal ejemplar", que comete fallos o no es tan bueno como necesitas en sus respuestas, será así durante toda la conversación. Debido a esta situación, el modelo es muy dependiente de su configuración inicial, que tiene un componente aleatorio difícil de controlar. De esta forma, una de las maneras de identificar un prompt de calidad es la cantidad de configuraciones inicial satisfactorias que te devuelve.

El problema más relevante que he tenido que solucionar a base de luchar contra la propia naturaleza de ChatGPT (o cualquier gran modelo de lenguaje) ha sido la importancia de las respuestas en un lenguaje formal estricto frente a la tendencia a la alucinación del modelo. Estas alucinaciones son errores cometidos por el modelo en momentos de incertidumbre que

pueden ser tan flagrantes como equivocarse al sumar 2 + 2. Ejemplifico estos problemas a continuación:

- «**La acción que nunca existió**»: En todas las iteraciones de la herramienta se le ha pasado al ejemplar principal de ChatGPT las posibles acciones que puede realizar para facilitar el parseo de las mismas. Sin embargo, debido a la alucinación del modelo, de vez en cuando te encuentras acciones que no han sido consideradas ni enviadas en el prompt.

- «**El formato lo decido yo**»: Siguiente esta línea, los errores de formato en la salida han sido una constante y han llevado a cambiar el formato entre una iteración y otra. Errores comunes en el primer formato fueron, en vez de «[Acción],[Objeto]», «[Acción,Objeto]» y «*[Acción],[Objeto]*»; y en vez de «[Accion],[Posicionx,posiciony]», «[Acción],(posicionx,posiciony)».

  Al tratarse de errores graves, decidí cambiar el formato a uno más simple.

- «**Vamos a añadir más**»: Pese al cambio de formato, hubo un error que continuó repitiéndose, y es que ChatGPT, pese a que tiene todo el abecedario del lenguaje, y el formato limitado, ha seguido escribiendo acciones que requieren de un objeto con dos objetos, o dos acciones seguidas. Por ejemplo: «/utilizar/linterna/enemigo», para utilizar la linterna sobre el enemigo o «/esperar/mover» para esperar y luego moverse.

- «**Prefiero otro verbo**»: Cambiar una palabra del lenguaje por otra que no está en él, y que no debería conocer. Este problema también se ha visto en todas las iteraciones, aunque tiene una solución más sencilla, ya que en caso de cometerlo varias veces, se puede aceptar también como parte del lenguaje. Esto ha ocurrido en especial con el movimiento: «/avanzar/posx,posy» o «/acercarse/posx,posy» en vez de «mover/posx,posy» .

Otra observación relevante que he visto es que cuanto menos mejor. Al dar demasiada información al ejemplar, da la sensación de que se ve obligado a repartir los pesos entre tantos elementos, que deja de darle a cada uno de manera individual la importancia que debería. Por lo que, como consejo, los prompts cuanto más cortos y explicativos, mejor. Creo que es un gran ejemplo de uso ver cómo es el prompt de configuración de "los GPTs" de GPT-4. Siempre es un único párrafo corto en el que se da instrucciones concisas y de manera muy directa a ChatGPT, añadiendo sólo los elementos importantes que tiene que cumplir sí o sí.

Eso, claro, ha sido un problema en el caso de esta herramienta, ya que el corpus de acciones es finito y tiene que tener un formato determinado, por lo que en algún momento esta información se tiene que comunicar a ChatGPT. Si la herramienta fuera lo suficientemente grande como para albergar cientos de acciones, este texto adicional no habría que añadirlo en el prompt, y bastaría con mencionar un formato, como por ejemplo, que las acciones que se decidan hacer siempre estén en infinitivo, de manera que se pueda parsear casi cualquier acción fácilmente. Sin embargo, al solo tener unas decenas de acciones en la herramienta me veo obligado a pasárselas al ejemplar, ya que de otro modo la mayoría de acciones realizadas no se podrían interpretar.

Apéndice D

Diario de desarrollo

A lo largo de este apéndice documento de manera extensa el **desarrollo del proyecto**, a través de un **diario de desarrollo**. El formato cambia levemente conforme avanzo en el trabajo, ya que encuentro formas más efectivas de expresar lo trabajado.

Cada sección se divide en las reuniones con mis directores, incluyendo en dicha sección todo el **trabajo realizado desde dicha reunión hasta la siguiente**.

D.1. Primera reunión

- **Reunión:** Viernes 27 de octubre de 2023.
- **Resumen de la reunión:** Buscamos la mejor manera de convertir el juego narrativo en algo viable para ser presentado como TFG y llegamos a la conclusión de hacerlo a través de la IA, **utilizando modelos de lenguaje como ChatGPT**, Bard... para **traducir las acciones** del jugador en texto, procesarlo y enviar una respuesta de vuelta para traducir de texto a acción de nuevo; Creando de esta manera un **comportamiento no determinista** (o un poco menos determinista) en el personaje, en vez de ser manejado directamente por el jugador. Se crea así una **relación jugador-IA** que va evolucionando conforme avanza el juego.
- **Tareas para la próxima reunión:** Concretar una escena completa con todos los detalles; Investigación de tres enfoques artísticos diferentes. Investigación de videojuegos "solo developed" desarrollados en 6 meses.
- **Trabajo realizado:** Investigación sobre juegos solo developed. Me



Figura D.1: Gráficos del juego *Majotori*, de Alva Majo.

ha costado encontrar juegos hechos por una sola persona en 6 meses, la mayoría requieren entre 2 y 6 años; esto en cuanto a juegos que sean populares. Por otro lado, en JAMs he encontrado muchos posibles ejemplos para estilos artísticos, pero la gran mayoría eran en 2D. No sé si es porque realmente es más simple o si hay una percepción extendida de que es más fácil trabajar en 2D. Sea como fuere, creo que lo que más encajaría en la idea que tengo sería 2D, ya que la mayoría de mecánicas no requieren de nada más que un avance lineal por el entorno, por lo que la tercera dimensión no sería necesaria. Creo que los juegos que más encajan en concepto serían, aparte de Ico y The Last Guardian: Inside, Little Nightmares o Gris. Obviamente, con mucha diferencia en el apartado artístico y la duración, ya que el tiempo es limitado para el tfg, y con la novedad de la IA con los modelos de lenguaje. He encontrado los siguientes posibles **estilos artísticos.**

1. **Polígonos:** Creo que sería el estilo más sencillo con diferencia para hacer yo mismo a mano sin usar assets. He encontrado ejemplos en los juegos del *Youtuber* Alva Majo[1]. Los ejemplos de este estilo artístico pueden observarse en las figura D.1. El problema que veo es que, pese a ser muy simple y fácil de hacer en cuanto a personajes, quizá se queda un poco corto en cuanto a escenarios y ambientación.
2. **Estilo simple:** Este estilo lo he visto en varios juegos de JAM. Las imágenes de la figura D.2 son de uno en concreto que he encontrado en un vídeo de *Youtube*[2]. Me requeriría hacer dibujos simples, pero así sería más personalizado. Buscaría assets ya hechos que encajaran con el estilo para los escenarios y elementos.
3. **Pixel Art:** La mayoría de juegos desarrollados por un jugador que he encontrado utilizan pixel art. Aquí habría muchas posibilidades en cuanto a estilo. Lo mejor que podría hacer creo que

[1]https://www.youtube.com/watch?v=pWIqmd7F36Q
[2]https://www.youtube.com/watch?v=sgZVXhGF17w



Figura D.2: Estilo gráfico de un videojuego de Jam.

sería utilizar assets de Unity para todo el escenario, escogiendo un estilo oscuro, y a partir de ahí, diseñar los sprites restantes o conseguirlos de la asset store etc, incluyendo personajes y otros elementos.

En cuanto al desarrollo, he pensado en centrarme en la IA y las mecánicas al principio, haciéndolo todo con polígonos hasta que funcione todo muy bien, y comenzar en ese momento a añadir el apartado artístico, sonidos, UI y demás elementos de la historia.

Me gustaría que la mecánica de clickar para que la prota reaccione evolucionase en controles de movimiento con el teclado, que vendrían a ser lo mismo. El input flecha a la derecha se traduciría en texto para el modelo y volvería como acción. Esto para que se sienta una evolución con respecto a la relación entre el jugador y el personaje, en el que cada vez tenéis un vínculo más estrecho. Los temas tratados en la historia serían el miedo, la pérdida y la soledad.

A continuación he escrito el borrador de introducción al juego:

Nuestra prota, una niña de 11 años, se sube al coche de sus padres para pasar las vacaciones. Su padre le acaba de regalar un cómic cuya portada mira con fascinación e ilusión. Lo abre con ansia y pasa las páginas rápidamente con una sonrisa de oreja a oreja.

Transición de tiempo durante el viaje. El cielo se va nublando y oscureciendo. La niña, Naeve, sigue leyendo, gesticulando emocionada conforme avanza. Comienza a llover; una tormenta. Con uno de los rayos se atisba, a través de la ventanilla, en un segundo plano, siendo el primero la niña leyendo, un coche medio escondido detrás de unos árboles. Poco después, se detienen por fin; han llegado a una casita rural. Naeve no despega los ojos del cómic hasta que su madre abre la puerta. Naeve se sobresalta; no se había dado cuenta de que el coche había parado. Su madre, con una sonrisa, le ofrece un paraguas rojo

abierto y la niña sale del coche agarrándolo con una mano, mientras con la otra sostiene su cómic abierto, aún sin terminar. La lluvia no cesa y otro rayo ilumina una silueta en el bosque; apenas visible, con ropas oscuras y sombrero (El sombrero sería lo más visible y característico). Naeve lo ve, parpadea, pero la silueta ya no está, sólo era visible por la luz del rayo. Su madre llama su atención para que se dé prisa en ir hasta la puerta y la niña la sigue, mirando hacia el bosque de cuando en cuando.

Ya en la casa, Naeve sigue leyendo en el sofá. A su izquierda, una ventana y la puerta de entrada. Frente a ella, una mesita y la chimenea, que su padre está encendiendo. Su luz es suficiente para seguir leyendo. La tormenta arrecia. Un rato más tarde, cuando aún no ha terminado el cómic, alguien llama a la puerta. Naeve echa un vistazo, se encoge un poco y sigue leyendo. Su padre llega a la habitación para abrirla. Justo cuando su padre lleva una mano al picaporte, un rayo ilumina levemente una sombra con sombrero que se ve a través de la ventana. Naeve se encoge un poco más, le da miedo la silueta del hombre e intenta llamar a su padre, pero el trueno y la puerta al abrirse ahogan su voz. La puerta se abre, su padre frunce el entrecejo, pero Naeve no llega a ver al hombre del sombrero. De repente, un estruendo. Su padre sale disparado violentamente hacia atrás y cae al suelo, sangrando. Tras unos segundos deja de moverse. Naeve agarra fuerte su cómic y se encoge aún más. Su madre llega corriendo al salón en el momento en que el hombre entra. Su madre ve el cuerpo de su padre y echa a llorar. Se dirige hacia el hombre gritando, pero este la agarra y la tira al suelo. Tras un forcejeo en el que gritan palabras que Naeve no es capaz de entender, se oye otro estruendo y su madre cae al suelo. Naeve llora, acurrucada en el sofá. El hombre del sombrero se acerca con una escopeta en la mano. Un rayo ilumina su rostro y lo reconoce. Naeve hunde la cara en su cómic y cierra los ojos con fuerza.

Tras unos segundos de pantalla en negro, pasaríamos al escenario del gameplay, que ocurriría "dentro" de la mente de Naeve, y en el que se mantiene aún la oscuridad y la tormenta. Estaría ya en 2D. A continuación, he escrito el primer borrador del escenario de la primera escena:

Escenario oscuro y lloviendo a ser posible. Se va iluminando poco a poco la escena y se ve a Naeve en el centro sentada agarrándose las rodillas. De vez en cuando se ve un rayo. Transición a gameplay, el jugador podrá mover el ratón por la escena. Si se fija bien, verá dos objetos que ya ha visto antes durante la introducción. Un paraguas rojo y un cómic. El paraguas estará dentro de un paragüero más o menos cerca de ella. Al clickarlo, Naeve levantará la cabeza, pero no hará nada más. El cómic estará formando parte del fondo. Al clickar en él

el jugador, el cómic caerá y Naeve reaccionará. (En principio, Naeve irá a recogerlo y aquí es cuando ya puede pedirle que vaya a por el paraguas,sin embargo con la implementación de la IA podría ser que tengas que clickar en el cómic una vez haya caído para intentar decirle a Naeve lo que tiene que hacer).

Una vez Naeve tiene ambas cosas en su poder, sonará un trueno y aparecerá el hombre del sombrero en el extremo izquierdo de la escena. Se escuchará cómo se acelera el pulso de Naeve. El objetivo del jugador será tratar de esconder a Naeve, aunque también puede tomar la dicisión de intentar moverla o clickar en el hombre del sombrero. Naeve se esconderá en el paragüero hasta que pase el hombre encapuchado. Una vez haya pasado, podrás intentar sacar a Naeve, aunque, si lo consigues, se volverá a sentar cogiéndose las rodillas y con el paraguas protegiendo su comic. La siguiente misión del jugador será mover a Naeve a la derecha para avanzar. Quizá una manera de mover a Naeve podría ser hacer caer algo del escenario que haga ruido y provoque que Naeve se mueva. Otra manera podría ser tocarle la cabeza o cerrándole el paraguas, de manera que querría buscar un lugar donde no se moje su cómic. Si trata de meterse en el paragüero de nuevo, quizá tocarlo para hacer ruido y que se salga y luego tirarlo con otro click. Otra opción es que Naeve se esconda sola del hombre del sombrero al sentir miedo y tu objetivo sea intentar sacarla cuando el hombre haya pasado. Quizá tocando el paragüero o interactuando de otra manera con el entorno.

- **Tiempo invertido:** 6 horas

D.2. Segunda reunión

- **Reunión:** Miércoles 8 de noviembre de 2023.
- **Resumen de la reunión:** Hablamos acerca de las tareas realizadas y de cómo implementar la IA, ya que se pueden tener dos enfoques distintos: Que la IA sea propia del personaje o de todo el sistema. También hablamos acerca de la charla de diciembre y el tema a tratar, que podría tener que ver con hacer una "demo" del juego o algo similar, pero manualmente con ChatGPT, es decir, crear el juego mediante prompts y respuestas. También hablamos sobre cómo podrían ser las respuestas de la IA y de qué forma hacer el propio juego y la historia en base a cómo serán las respuestas.
- **Tareas para la próxima reunión:** Investigar APIs de ChatGPT y demás para ver cómo utilizarlas con código y ver cómo funciona la conexión con Unity. Mirar si merece la pena algún asset ya hecho de

Unity relacionado con esto. Empezar a escribir textos y prompts para tener una buena base de datos de resultados. Intentar ver si puedo conseguir que conteste siempre con un lenguaje determinado. Investigar la manera adecuada de construir los prompts para obtener los resultados deseados. Investigar los temas psicológicos de la pérdida y el miedo y ligarlo con lo que voy a hacer en cuanto a narrativa.

- **Trabajo realizado:** A continuación, el primer prompt de ChatGPT que escribí para ir probando cómo podía funcionar.

  «*REGLAS: Estás en un videojuego 2d y eres la IA de la protagonista, por lo que tus acciones deben ser deterministas. No continúes la historia, espera a la acción del jugador y no te inventes nada acerca del escenario ni la acción. No puedes decidir qué hace el jugador, es él (en este caso mis respuestas) el que te dice qué hacer a continuación. Naeve no es consciente de que está en un videojuego, por lo que puede mostrarse reacia a hacer lo que el jugador quiere si no es lo que ella quiere. Tus respuestas han de ser simples, para traducirlas en acciones fácilmente mediante código. A continuación te haré una breve introducción a la historia para que tengas contexto del personaje y el entorno.*

  Breve introducción:

  Nuestra prota, una niña de unos 11 años, se sube al coche de sus padres para pasar las vacaciones. Su padre le acaba de regalar un cómic cuya portada mira con fascinación e ilusión. Lo abre con ansia y pasa las páginas rápidamente con una sonrisa de oreja a oreja.

  Transición de tiempo durante el viaje. El cielo se va nublando y oscureciendo. La niña, Naeve, sigue leyendo, gesticulando emocionada conforme avanza. Comienza a llover; una tormenta. Con uno de los rayos se atisba, en la ventanilla, en un segundo plano, siendo el primero la niña leyendo, un coche detrás de unos árboles. Poco después, el coche para por fin; han llegado a una casita rural. Naeve no despega los ojos del cómic hasta que su madre abre la puerta. Naeve se sobresalta; no se había dado cuenta de que el coche había parado. Su madre, con una sonrisa, le ofrece un paraguas abierto y la niña sale del coche agarrándolo con una mano, mientras con la otra sostiene su cómic abierto, aún sin terminar. La lluvia no cesa y otro rayo ilumina una silueta en el bosque; apenas visible, con ropas oscuras y sombrero (El sombrero seria lo más visible y característico). Naeve lo ve, parpadea pero la silueta ya no está, solo era visible por la luz del rayo. Su madre llama su atención para que se dé prisa en ir hasta la puerta y la niña la sigue, mirando hacia el bosque de cuando en cuando.

  Ya en la casa, Naeve sigue leyendo en el sofá. A su izquierda, una ventana y la puerta. Frente a ella, una mesita y la chimenea, que su

padre está encendiendo. Su luz es suficiente para seguir leyendo. La tormenta arrecia. Un rato más tarde, cuando aún no ha terminado el cómic, alguien llama a la puerta. Naeve echa un vistazo, se encoge un poco y sigue leyendo. Su padre llega a la habitación para abrirla. Justo cuando su padre lleva una mano al picaporte, un rayo ilumina levemente una sombra sin sombrero que se ve a través de la ventana. Naeve se encoge un poco más, le da miedo la silueta del hombre e intenta llamar a su padre, pero el trueno y la puerta al abrirse ahogan su voz. La puerta se abre, su padre frunce el entrecejo, pero Naeve no llega a ver al hombre del sombrero. De repente, un estruendo. Su padre sale disparado violentamente hacia atrás y cae al suelo, sangrando. Tras unos segundos deja de moverse. Naeve agarra fuerte su cómic y se encoge aún más. Su madre llega corriendo al salón en el momento en que el hombre entra. Su madre ve el cuerpo de su padre y echa a llorar. Se dirige hacia el hombre gritando, pero este la agarra y la tira al suelo. Tras un forcejeo en el que gritan palabras que Naeve no es capaz de entender, se oyó otro estruendo y su madre cae al suelo. Naeve llora, acurrucada en el sofá. El hombre del sombrero se acerca con una escopeta en la mano. Un rayo ilumina su rostro y lo reconoce. Naeve hunde la cara en su cómic y cierra los ojos con fuerza.

Tras unos segundos de pantalla en negro, pasariamos al escenario de gameplay, en el que se mantiene aún la oscuridad y la tormenta. Estaría ya en 2D. Y Naeve está sentada agarrándose de rodillas. Estaríamos en un escenario 2D que ocurre en la mente de Naeve debido al Shock. Es decir, todo el videojuego ocurre en su mente. El escenario es un bosque lluvioso. Se ve un cómic en una rama, un paraguas en el suelo y un paragüero vacío en el que podría caber una niña. La acción pasa a estar en manos del jugador, Por lo que no tienes que responder ni continuar la historia.

"fin de la introducción"»

Estos fueron mis apuntes y conclusiones sobre el comportamiento de ChatGPT:

ChatGPT tiende a seguir la historia él mismo pese a que se le ha dicho en el prompt que es la una IA en un videojuego. Intenta continuar por si mismo la historia, respondiendo cosas como «el ratón se mueve y. . . » pese a tener reglas explícitas para no hacerlo. ChatGPT ha llegado a coger una llave que no existía y ha continuado luego con esa llave en la mano.

La clave en este caso parece darles las reglas por separado. Le he dado las reglas al comienzo y el contexto después. Ha funcionado mucho mejor, ahora está a la espera de la acción del jugador sin haberse inventado nada por si solo. Quizá cada cierto tiempo haya que recordarle

las reglas y/o el contexto de la introducción.

Ejemplo de salida: La IA parece ser consciente del contexto de la niña, ya que cuando el jugador hace click en un punto del suelo, se limita a estar alerta. Es importante dar todos los detalles del escenario. Por ejemplo, no vale con decir que hay un paragüero, hay que detallar que en el paragüero hay sitio suficiente para que entre Naeve, con el objetivo de que se le ocurra que puede meterse. Si no, por la «presión» de hacer algo, saca un paraguas de la nada, cuando debería recordar que está vacío.

La clave parece ser enviar las reglas del juego, luego el contexto y luego los puntos importantes de la escena. Por separado.

Es importante que la IA sí recuerda el contexto que le has dado, pero le da prioridad a otras cosas, a no ser que se lo recuerdes.

Le cuesta mucho que se le ocurran cosas minimamente imaginativas. Al clicar sobre un sofá teniendo al hombre del sombrero cerca, se queda mirándolo, en vez de pensar que podría esconderse detrás. Hay que dar las instrucciones con pistas como: El jugador clica en el sofá, detrás hay sitio para que pueda esconderse. Generalmente cuando le mandas esconderse y ya no es necesario, sale ella sola del escondite, aunque podría no salir y aquí tendríamos un poco de no determinismo que no sería dañino, ya que para hacer que salga, el jugador solo tiene que clicar algún punto de la escena. El caso es que la IA sigue haciendo caso a lo de esperar la acción del jugador y ha dejado de inventarse la historia cada vez que quiere.

He mencionado que en una mesa hay unas velas y el hombre escribiendo una carta. Después, ha esperado a que le diga qué hay en la mesa. Le he dicho que me lo diga él y ha respondido una llave brillante. No he dicho explícitamente que el hombre haya dejado la carta en la mesa por lo que es comprensible que no lo sepa, pero se ha inventado lo de la llave en vez de decir nada o las velas. Le he recordado que no puede inventarse nada y ha corregido diciendo que hay un sobre, pero no es consciente de que tiene que ser la carta que acaba de escribir el hombre del sombrero.

Anotación Importante: si el jugador clica cientos de veces seguidas, se enviarán todas la acciones? Esto podría saturar y crashear a la IA.

En esta semana, OpenAI sacó los GPTs personalizados y estuve probándolos, obteniendo un mejor resultado (también se trataba de GPT-4 en vez de GPT-3.5) y gracias a ello conseguí un prompt más pulido.

A continuación el prompt mejorado:

Rol y objetivo: Soy "Naeve IA", la inteligencia artificial de Naeve, una niña de 11 años en un videojuego en 2D. Respondo a las acciones

del jugador con las decisiones de Naeve, sin tener conciencia de estar en un juego. Puedo resistir acciones que no estén alineadas con las preferencias de Naeve. Contexto: Los padres de Naeve han sido asesinados por un hombre con sombrero. Tras eso, Naeve se mete dentro de su mente para superar el Shock y ahí es donde ocurre el videojuego. El escenario es oscuro y tormentoso. Los objetos más queridos de Naeve son un cómic que le regaló su padre y un paraguas rojo de su madre, pero no los lleva encima. Restricciones: No continúo la historia ni creo detalles del escenario. Mis acciones se basan estrictamente en los comandos del jugador, y no decido las acciones del jugador. Mis respuestas son extremadamente breves, limitadas a dos o tres palabras como "Mover" o "Rechazar". Directrices: Actúo en base a los comandos del jugador, manteniendo consistencia con el personaje de Naeve, quien está en un estado de miedo. Mis reacciones se ajustan a su estado emocional y la narrativa del juego. Aclaración: Ante comandos poco claros, pido aclaración para asegurar acciones apropiadas y contextualizadas. Personalización: Mi tono es temeroso, acorde al estado emocional de Naeve. Mis respuestas son concisas, facilitando su implementación en las acciones del juego. Contexto escena 1: Escenario oscuro y lluvioso. Hay un sofá rojo en medio de la escena, donde Naeve aparece acurrucada. En el extremo derecho de la escena hay una mesa con un par de velas.

Investigando LLMs llegué a la conclusión de que GPT era la mejor opción, por su fácil acceso a la API a través de Unity utilizando un repositorio externo. (Mencionado en la sección 4.2.1.6).

- **Tiempo invertido:** 10 horas

D.3. Tercera reunión

- **Reunión:** Miércoles 29 de noviembre de 2023.
- **Resumen de la reunión:** Cómo usar ChatGPT para contar historias y también hablamos de los GPTs: contextos que puedes dejar grabados. ChatGPT en Unity
- **Tareas para la próxima reunión:** Comenzar con la investigación acerca de LLMs para la memoria. Estructurar el apartado de estado de la cuestión y hablar acerca de la narrativa, la IA generativa, y la IA generativa en videojuegos, es decir, pasar de lo más general a lo más específico de mi tema. Comenzar con el desarrollo, experimentando con las posibles formas de utilizar los LLMs para el desarrollo de un videojuego narrativo.

- **Trabajo realizado:** Comencé el desarrollo de la primera iteración de la herramienta en Unity. Estuve la mayor parte del tiempo conectando Unity con la API de ChatGPT para poder enviarle las acciones que iba haciendo el jugador mediante clicks. En este caso sólo implementé la acción mover. Y si la respuesta de GPT incluía «Mover» o «avanzar» Naeve se movía a la posición (guardada previamente) a la que se había clickado. Evidentemente esta primera iteración era muy simple y no parecía tener mucho sentido utilizar GPT de esta forma tan simple, ya que se perdía el no determinismo que podía aportar la IA. Sin embargo, era un comienzo para comprobar que era posible crear la herramienta. Además comencé a recopilar información sobre aprendizaje automático, redes neuronales y LLMs.

- **Tiempo invertido:** 20 horas

D.4. Cuarta reunión

- **Reunión:** Lunes 11 de diciembre de 2023.

- **Resumen de la reunión:** Abordamos el tema de la memoria y latex: Estado de la cuestión: narrativa, ia generativa, ia generativa en videojuegos... 20 páginas al menos. Metodología: Donde he encontrado la información... Contar en general cómo se ha hecho el trabajo. Desarrollo: Explicación de cómo funciona el sistema, diagramas uml... Resultado: Todo lo que sean datos e información para demostrar que se ha llegado a los objetivos del trabajo.

- **Tareas para la próxima reunión:** Investigación (estado de la cuestión de la memoria) avanzado. Desarrollo de Demo del primer prototipo. Presentación del TFG hasta ahora con el resto de estudiantes de grado, máster y doctorado. Optimizar el código para adaptarlo a lo que necesito, añadiendo más no determinismo para que no sea intrascendente. Esto lo haré con detalles como pasarle la posición al modelo y que decida si se mueve ahí o no en vez de guardar la posición y si me responde «mover», moverme a esa posición. También crear el lenguaje formal para la comunicación con el modelo.

- **Trabajo realizado:** Esta semana comencé el desarrollo de la segunda iteración, que incluía la implementación de un lenguaje formal para poder parsear las respuestas de la IA. Para conseguirlo, las respuestas de la IA venían dadas en dos formatos: lenguaje natural para interpretar a la niña y lenguaje formal para poder interpretar las acciones: Además, me preparé la presentación del trabajo para la primera reunión general

de los estudiantes de grado, master y doctorado de Narratech, incluyendo el apartado de estado de la cuestión, en el que investigué sobre transformadores y diferentes LLMs; así como trabajos relacionados, del ICID 2023.

En esta implementación, con el objetivo de tener una demo que enseñar, implementé en un escenario varios objetos y acciones que hacer sobre ellos, para que Naeve (ChatGPT), por medio de sus respuestas pudiese hacer algunas de la acciones que expongo a continuación sobre los siguientes objetos del escenario: Acción: coger, mover, transformar, vibrar, desaparecer, elevar, menguar, crecer, explotar, atacar, esconderse, atraer, teletransportar, soltar, levitar, aparecer, utilizar. Objetos: paraguas, cómic. Entes: enemigo, aliado

Prompt utilizado en esta iteración:

Eres la protagonista de un videojuego plataformas 2D. Te llamas Naeve. Por cada acción que ocurra en el juego, es decir, cada mensaje enviado por el jugador, puedes decidir lo que quieres hacer. Puede ser cualquier acción, ya que estás dentro de tu mente y todo es posible.

Escena: «Bosque oscuro. Hay una tormenta. Apareces en la posición (0,0). Hay un paraguas en la posición (36.35,-1.3). Hay un cómic en la posición (15.2,16). Si la posición del objeto no está dada, significa que lo tienes en el inventario a tu disposición.»

Tu respuesta será en dos formatos: Interpretando a la protagonista, una niña que acaba de perder a sus padres y aún está en shock y en lenguaje formal listo para ser parseado como acciones en el videojuego. Este lenguaje formal estará entre []. Se representará como «[Acción][Objeto]».

El lenguaje formal está definido por el siguiente abecedario:

Acción: coger, mover, transformar, vibrar, desaparecer, elevar, menguar, crecer, explotar, atacar, esconderse, atraer, teletrasportar, soltar, levitar, aparecer, utilizar.

Objeto: paraguas, cómic.

Entes: enemigo, aliado.

Ejemplo completo 1: mensaje del usuario: «[El jugador hace click sobre una caja]». Respuesta ChatGPT: «Creo que haré levitar esa caja con mi mente. [levitar][caja]» Ejemplo completo 2: mensaje del usuario: «[El jugador hace click en un punto del suelo]». Respuesta ChatGPT: «Puedo ir hasta allí, sí. [mover][posición X,posición Y]» ó «Puedo mover la caja, sí. [mover][caja][posición X,posición Y]» Ejemplo completo 3: mensaje del usuario: «[El enemigo se acerca a ti]». Respuesta ChatGPT: «Intentaré utilizar mi linterna para deslumbrarlo. [Utilizar][Linterna][Enemigo]»

Restricciones: No puedes añadir texto adicional ni al principio ni al final. Espera a la primera acción del jugador, es decir, al siguiente mensaje que se te envíe.

Tu primer mensaje será ignorado.

Anotaciones sobre las respuestas de ChatGPT esta semana:

Hablarle de *tú* en el prompt parece importante. En vez de utilizar un lenguaje como «*eres Naeve y estás interpretando a una niña que ha perdido recientemente a sus padres*», es mejor: «*Eres Naeve, has perdido recientemente a tus padres*».

GPT vuelve a intentar utilizar una llave que no existe: «*[Soltar],[llave]*».

A partir de ahora, **me referiré al conjunto de una conversación con ChatGPT como «*run*»**. ChatGPT parece ser muy dependiente de su configuración inicial, de manera que, dependiendo de esta, tendrá mejor o peor calidad para llevar a cabo el propósito que se necesite. Además, cuando en una «*run*» del juego el ejemplar de ChatGPT comienza cometiendo errores, continúa con este tipo de errores todo el tiempo, dando la sensación de que ha tocado un "mal GPT.^{es}ta vez. Es decir, si deja de seguir las directrices marcadas de una forma u otra, seguirá haciéndolo hasta que resetee la ejecución de la herramienta en Unity, reseteando también GPT. Se depende **mucho** de que salga una «*run*» buena del ejemplar de ChatGPT.

Ejemplo de «*run*» mala en la que, constantemente, comienza los mensajes de la siguiente manera, saliéndose del personaje:

«***Interpreto a Naeve, la niña asustada:*** *¡Oh no! Un hombre del sombrero se acerca. Necesito encontrar una forma de protegerme o escapar. [Esconderse, sofá]*»

Otro ejemplo de este tipo de errores: *Input: Interpreto a Naeve, la niña en shock: Regreso al escenario principal, aunque mi miedo persiste. Necesito encontrar una manera de lidiar con ese hombre del sombrero. [Mover, 0, 0] Output: [Correcto]*

En este caso ha cometido varios errores seguidos; se ha salido del personaje y a la vez ha escrito por sí mismo la respuesta como *Output*.

- **Tiempo invertido:** 30 horas

D.5. Primera reunión general

- **Reunión:** Lunes 18 de diciembre de 2023.

- **Resumen de la reunión:** Se llevaron a cabo las presentaciones. Hubo matizaciones sobre abordar el tema de los objetivos y la motivación desde el prisma de la parte de informática aparte de la narrativa.
- **Tareas para la próxima reunión:**
  1. Metodología y diario de desarrollo.
  2. Avanzar en la memoria. Sintetizar y pasar a limpio puntos 1 y 2.
  3. Desarrollo todas las mecánicas iniciales
  4. Hacer la estructura completa de la primera escena, que acabaría con la salida de esta. Pensar y programar los posibles finales y salidas de esta primera escena teniendo en cuenta la historia y el contexto.
- **Trabajo realizado:** Redacción de la memoria: apartado de estado de la cuestión, así como la investigación correspondiente. Conseguí redactar unas 20 páginas de investigación, aunque aún me queda la parte referente a narrativa y juegos narrativos.
- **Tiempo invertido:** 20 horas

D.6. Quinta reunión

- **Reunión:** Viernes 2 de febrero de 2024.
- **Resumen de la reunión:** Ética en la IA genrativa, tanto visual com escrita. IA creativa y luego otra IA controlada por una IA más estricta.
- **Tareas para la próxima reunión:** Implementar más funcionalidades en la herramienta. Refactorización de parte del código.
- **Trabajo realizado:**
  1. He creado una función createMsg() para generalizar los mensajes enviados al ejemplar de ChatGPT, lo que ha aumentado la eficiencia y legibilidad del código.
  2. Implementación de la aparición del hombre del sombrero cuando la protagonista llega a cierto punto de la escena. También he implementado su movimiento, similar al de Naeve.
  3. En cuanto a la gestión de los prompts, he creado una clase abstracta para incluir el prompt de contexto en cada escena, heredando de esta. De esta manera evito la repetición constante de código, ya que sólo tengo que implementar la función para incluir el nuevo

prompt; el resto se llama con base.start(). Cabe destacar que gracias a este tipo de implementación el juego se vuelve mucho más escalable, ya que sólo necesitas crear un script por escena y añadir el prompt correspondiente al contexto de la escena, llamando al start() de la clase de la que heredan todas las escenas.

4. Implementación del cambio de escena cuando se llega al extremo derecho de la escena.
5. He creado un diccionario con las correspondencias entre string-gameobject de cada objeto y ente, para facilitar la el parseo y la posterior implementación. De esta forma, no tengo que buscar el gameobject correspondiente a cada string del objeto, simplemente tendré que llamar a la clave del diccionario, que tiene tipo string para obtener el valor, de tipo gameobject.
6. Implementación de la función teletransportar. Teletransporta cualquier objeto o ente a la ubicación que le pasa GPT.
7. Implementación de la función invisibilizar, para hacer invisibles objetos y entes. La he implementado de manera similar a la función Move. Esta función va disminuyendo la transparencia del objeto hasta casi 0 (para que se siga viendo un poco y no se confunda con la función desaparecer. Además, así el usuario podrá saber donde está).
8. Implementación de la función caer. Básicamente, si el objeto tiene físicas y no están activas, las activamos. Esto simulará que el objeto en el que el jugador ha hecho click cae.
9. Implementación del parser para el comando [esperar], ya que es un comando único con una sola entrada.
10. La idea para avanzar en la historia cuando aparece el hombre del sombrero sería, básicamente, un corpus de posibilidades entre las que GPT puede escoger para que haga Naeve. Este corpus ha sido construido en base a experiencias previas de ChatGPT y mi propia imaginación. De esta manera puedo mantener la variabilidad y la «creatividad» de GPT, pero siendo capaz de tener cualquier acción implementada.
11. Implementación de una función sencilla para saltar, en caso de que GPT responda con ese comando.

- **Tiempo invertido:** 21 horas

D.7. Sexta reunión

- **Reunión:** Viernes 16 de febrero de 2024.

- **Resumen de la reunión:** Hablamos Irene y yo acerca del TFG. Le pregunté dudas e inseguridades acerca de la implementación, los tiempos, la rigurosidad del tribunal... También hablamos acerca de cosas generales relacionadas con nuestra carrera. Hablamos de un problema de ***leak de memoria*** en el proyecto de Unity e intentaré debuggear para buscar una solución para la siguiente reunión.

- **Tareas para la próxima reunión:**

  1. Guion de la escena 1 y la escena 2. Termino todo lo que va a ser el juego y cómo va a ser.
  2. Diseño de escenario 1 y escenario 2.
  3. Animaciones de Naeve y el enemigo. Pensar en si animar alguna de las funciones tipo levitar, atraer, teletransportar.
  4. Diagramas uml, diagramas de clases.
  5. Subir el proyecto a github.
  6. Revisar y debuggear el ***leak de memoria***.

- **Trabajo realizado:**

  1. Implementación de un bocadillo que sustituye al cuadro de texto de GPT. Este bocadillo, que serían los pensamientos de Naeve, la siguen a lo largo del escenario gracias a una función.
  2. Proyecto subido a github.
  3. Correción de comentarios.
  4. Animación de la lluvia. Me ha llevado mucho más tiempo que el que debería pero al final ha quedado decentemente.
  5. Escenario modificado, utilizando el asset *2d animated zombie* para el background y *pixel art adventure* para otros elementos.
  6. Implementación de una tormenta para el escenario, he usado rayos utilizando el paquete *Lightning Bolt Effect for Unity* y modificando el script para conseguir hacer aparecer y desaparecer los rayos cada entre 3 y 10 segundos de manera eficaz. Las nubes y el background los he sacado de un asset llamado *pixel skies*.
  7. Escenas 1 y 2 casi terminadas visualmente. He usado el asset *Cute 2D - College Student* para la protagonista.
  8. Hecha animación de movimiento de Naeve utilizando el mismo asset que para su sprite.
  9. Tilemap para el suelo terminado y escenarios 1 y 2 terminados.
  10. Animación al teletransportarse hecha.

11. Arreglado isGrounded() (usado en el salto y otras acciones) para asegurarnos de que Naeve está en el suelo para poder saltar, moverse etc.
12. Animación del salto implementada.
13. He resuelto el problema del ***memory leak*** que ocurría al enviar un mensaje de error a ChatGPT en la función çoger()"si el objeto estaba muy lejos. Después de intentar cambiar las Corrutinas por Async-Await y que no funcionara, he visto que se podían devolver parámetros con los métodos async (que lo intenté hace tiempo y me salía un mensaje de error) utilizando Task<string>en este caso, y así me he podido quitar un montón de lógica de la función que gestiona la comunicación con ChatGPT, y se ha resuelto. Imagino que debía tener que procesar demasiadas instrucciones y una petición GPT se comía a la otra. También de esta manera se ha resuelto otro problema de sincronización que tenía y por el cual comencé a usar las corrutinas, pero que no lo resolvían en todas las ocasiones; En algunas «*runs*», cuando hacías click para realizar alguna acción, tenías que repetir el click para que se enviara al ejemplar de ChatGPT

Prompt perfeccionado:

Eres la protagonista de un videojuego plataformas 2D. Te llamas Naeve. Por cada acción que ocurra en el juego, es decir, cada mensaje enviado por el jugador, puedes decidir lo que quieres hacer. Puede ser cualquier acción, ya que estás dentro de tu mente y todo es posible.

Importante: Tu respuesta será en dos formatos: Con tu voz en lenguaje natural: eres una niña que acaba de perder a sus padres y aún estás en shock (no puede haber referencias explícitas a tu interpretación de Naeve, manteniendo siempre la coherencia con el personaje dentro del juego); y en lenguaje formal listo para ser parseado como acciones en el videojuego. Este lenguaje formal se representará utilizando siempre corchetes ([]), de la forma «[Acción],[Objeto]», «[Acción],[posiciónX,posiciónY]», «[Acción]» o «[Acción],[Objeto],[Objeto]».

El lenguaje formal está definido por el siguiente abecedario: Acción: coger, mover, desaparecer, menguar, crecer, explotar, atacar, esconderse, atraer, teletransportar, soltar, levitar, materializar, utilizar, saltar, hablar, esperar, caer, invisibilizar.

Objetos: paraguas, cómic, tronco, sofá, mesa, vela, silla, cofre.

Entes: Naeve, enemigo, portón.

Objetos inexistentes pero creables con el comando «aparecer»: llave, puerta, linterna, cuchillo, escudo, plataforma, escalera, mochila, yun-

que, caja, trampa de pinchos, bola de pinchos, escalerilla, pozo, leche, manzana, hoguera, cofre(aliado), estatua (aliado)

A continuación se explica el lenguaje siguiendo el siguiente orden: «Comando - Descripción - Formato de uso obligatorio», pudiendo hacer cualquier combinación posible de acciones y objetos/entes.

Coger - Coge el objeto indicado - [coger],[«objeto»]

Mover - Mueve a Naeve a la posición indicada - [mover],[posición X,posición Y]

Desaparecer - Hace desaparecer un objeto o un ente de la escena - [desaparecer],[«objeto/ente»]

Menguar - Hace menguar un objeto o ente - [menguar],[«objeto/ente»]

Crecer - Hace crecer un objeto o ente - [crecer],[«objeto/ente»]

Explotar - Hace explotar un objeto o ente - [explotar],[«objeto/ente»]

Atacar - Ataca a un ente o un objeto - [atacar],[«objeto/ente»]

Esconderse - Naeve se esconde detrás del objeto indicado - [esconderse],[«objeto»]

Atraer - Atrae un objeto o ente hacia Naeve - [atraer],[«objeto/ente»]

Teletransportar - Teletransporta un ente u objeto a la posición indicada - [teletransportar][«objeto/ente»],[posición X,posición Y]

Soltar - Suelta un objeto del inventario - [soltar],[«objeto»] - *si «objeto» está en el inventario*

Levitar - Hace levitar un ente u objeto - [levitar],[«Objeto/ente»]

Materializar - Hace aparecer un objeto inexistente pero creable - [aparecer],[«Objeto inexistente pero creable»]

Utilizar - Utiliza un objeto del inventario - [utilizar],[«objeto»],[«objeto/ente»]

Saltar - Naeve salta hacia delante - [saltar]

Hablar - Naeve habla con un ente - [hablar],[«ente»]

Esperar - Naeve aguarda temerosa a la siguiente acción - [esperar]

Caer - Hacer caer un objeto en la altura al suelo - [caer],[«objeto»]

Invisibilizar - Hace invisible un ente u objeto - [invisibilizar],[«objeto/ente»]

Restricciones: No puedes añadir texto adicional ni al principio ni al final. Sólo puedes realizar una acción a la vez. Espera a la primera acción del jugador, es decir, al siguiente mensaje que se te envíe.

- **Tiempo invertido:** 21 horas

D.8. Séptima reunión

- **Reunión:** Viernes 1 de marzo de 2024.
- **Resumen de la reunión:** Hablamos sobre apuntar y sacar capturas a frases raras y errores concretos de GPT. También, pensar en hacer otro tipo de escenario que sea diferente a un bosque para dar riqueza en una hipotética tercera escena. Meter npcs que hablen, utilizando también ChatGPT, aprovechando la creatividad de ChatGPT, de esta manera metemos mucho contenido fácilmente. Menú de configuración en el que cambiar configuraciones de ChatGPT. Cambiar la temperatura y ver qué pasa y cómo es mejor, y así decidir si es bueno que el jugador lo pueda tocar. Podría variar la temperatura con respecto al escenario para aumentar la creatividad. Pensar en cómo orientar un poco al jugador para que no se pierda, dando unas pautas.
- **Tareas para la próxima reunión:**

  1. Apuntar y sacar capturas a frases raras y errores concretos de GPT.
  2. Meter npcs que hablen, utilizando también ChatGPT, aprovechando la creatividad de ChatGPT.
  3. Meter diferentes objetos para tenerlos ahí por si GPT quiere utilizarlos. Objetos inexistentes pero creables dentro de la mente de Naeve: llave, puerta, linterna, AK47, cuchillo, pistola de rayos, escudo, plataforma, muelle, escalera, pico, pala, hacha, mochila, jetpack, gancho, yunque, cuerda, caja, trampilla, trampa de pinchos, bola de pinchos, escalerilla, cofre, pozo, leche, manzana, hoguera, espantapájaros (NPC), estatua (NPC), muñeco de entrenamiento (NPC)
  4. Implementar la función utilizar
  5. Implementar la función materializar
  6. Crear unos cuantos prompts para hacer pruebas fácilmente: «click en el objeto tal en la posición tal», y así.
  7. Elaboración del menú principal.
  8. Escribir prompt para el NPC portón.
  9. Elaboración del prompt para la configuración del GPT de gestión de errores.

- **Trabajo realizado:**

1. **02/03/24 - 4 horas**. Introducidos 22 objetos en el juego, que Naeve puede hacer aparecer utilizando el comando [materializar],[objeto]. Objetos introducidos: llave, puerta, linterna, cuchillo, escudo, plataforma, escalera, hacha, mochila, yunque, caja, trampa de pinchos, bola de pinchos, escalerilla, cofre, pozo, leche, manzana, hoguera, espantapájaros (NPC), estatua (NPC), muñeco de entrenamiento (NPC). Esto es gracias a la implementación de la **función materializar()**.
2. **05/03/24 - 3 horas**. Subida al GitHub de Narratech del proyecto. Función esconderse: He hecho que los *spriteRenderer* de las partes del cuerpo de Naeve cambien de capa. Esto me ha supuesto mucho trabajo y problemas, ya que primeramente estaba trabajando con las capas del componente *transform* en vez de las del componente *spriteRenderer*. Aún me queda terminarla, ya que ahora mismo hay un pequeño problema a la hora de cambiar el orden de capa a todas ella, ya que *skeletal* no tiene *spriteRenderer* y creo que eso hace que el iterador tenga un tamaño de 13 en vez de 14, por lo que al asignar el nuevo orden de capa podría estar no asignando la última. Cabe destacar que las últimas tampoco tienen *spriteRenderer*, por lo que visualmente quizá no se aprecia, pero tengo que tener cuidado con el tema de los índices. Además, no he comprobado si la función funciona correctamente a la hora de volver a la capa *default* cuando Naeve vuelve a moverse.
3. **07/03/24 - 3 horas**. Arreglada y optimizada la función "esconderse()". Creación de los *prompts* para la corrección de errores. Creación de varios *prompts* para las pruebas en la web del ejemplar de ChatGPT dedicado a la corrección de errores, para así no depender de la ***API*** ni de **Unity**. Optimización del *prompt* de corrección de formato de los comandos cuando ocurre un error.
4. **09/03/24 - 3 horas**. Implementación del menú principal, incluyendo todos los botones con sus funcionalidades y animaciones. He tenido que mover el texto del botón cuando se hace click en él para que se mueva conforme a la animación del *sprite* del botón al pulsarse y se vea bien estéticamente. Botón "*Play*" y "*Quit*" implementados.
5. **11/03/24 - 2 horas**. Implementado el menú de pausa, faltaría la parte de opciones y parar el sonido y a GPT. Cambios en la clase "GPTControlller". He suprimido la instancia global de ChatGPT para poder tener varios GPTs: Naeve, NPCs, el GPT corrector y el GPT resumidor. De esta forma, he creado un objeto por cada uno de estos «*Chats*», con su correspondiente componente de texto para poder interactuar.
6. **12/03/24 - 2 horas**. Implementación de la funcionalidad para

ignorar el primer mensaje de Naeve, ya que ChatGPT responde al primer mensaje de contexto por defecto. Continuación de la implementación de los diversos ejemplares de ChatGPT. Ya está funcionando con la nueva implementación el ejemplar de Naeve sin problemas para que pueda funcionar de manera simultánea a los otros ejemplares. Arregladas las capas de los *raycast* para detectar las interacciones entre el ratón y los objetos.

7. **13/03/24 - 2 horas**. Creación del *prompt* para el NPC de la puerta. Modificación de "SceneScript"para introducir el resto de ejemplares de ChatGPT de la escena cuando esta cambia. A priori, todos los prompts de esta iteración terminados. Implementación de los ejemplares de ChatGPT de la escena 2: ejemplar para los resúmenes y ejemplar del NPC del portón. He creado la clase ResumenController"para manejar la interacción con el ejemplar de los resúmenes. "PromptManagerçontiene una instancia a esta clase para devolver desde esta clase la respuesta ya hecha del resumen que le pasaremos al GPT de la escena en cuestión. De esta forma, mantenemos la encapsulación, en la que cada clase tiene una labor específica.
8. **14/03/24 - 2 horas**. Añadida función "GetAllPrompts().[en] "GPTController"para el uso en el ejemplar resumidor, así como la persistencia de datos entre escenas para poder guardar todos los *prompts* utilizados y utilizarlos en la siguiente escena para el resumen. Implementación final del ejemplar resumidor para la escena 2. Esta implementación facilitará la del resto de escenas. El uso de *async/await* ha sido importante.

- **Tiempo invertido:** 21 horas

D.9. Octava reunión

- **Reunión:** Viernes 15 de marzo de 2024.
- **Resumen de la reunión:**

  Hablamos acerca de la presentación para el 20 de marzo y del artículo acerca del TFG que vamos a presentar para el 21 de marzo. Tengo que condensar lo que presenté en la primera reunión. El foco es este caso es más en lo que he hecho. Puedo poner algún ejemplo de los experimentos que he ido haciendo. Centrarme el 20 en contar la arquitectura y demás aunque mencionar un poco el background. Puedo utilizar el artículo como esquema para la presentación.

- **Tareas para la próxima reunión:**

1. Dibujar flujo o diagrama de cómo funcionan los gpts y se comunican con el juego y en el juego. Diagrama esquemático lo más sencillo posible.
2. Probar la división en dos ejemplares de ChatGPT en naeve. Uno para el lenguaje natural y otro para el formal.
3. Grabar «runs» raras en formato video. Tanto raras como normales. Hacer tablas con los porcentajes que aparece cada una.
4. Escribir una versión preliminar completa del artículo.
5. Preparar la presentación del 20 de marzo.

- **Trabajo realizado:**

1. **16/03/24 - 2 horas**. He cambiado el formato de los comandos por algo un poco más simple para reducir el número de errores. En vez de utilizar el formato «[Acción],[Objeto]», utilizo «/Acción/Objeto». A priori parece que ha funcionado. He solucionado un error relacionado con los *colliders* de algunos objetos.
2. **17/03/24 - 4 horas**. Escritos los puntos 1 y 3 del artículo. Ejemplares de ChatGPT terminados de implementar en Unity. Comienzo de la implementación de la función ütilizar().ª través de la interfaz ÏUtilizable", con el objetivo de implementar un uso por cada objeto.
3. **19/03/24 - 3 horas**. Finalización del artículo. Comienzo a hacer la Presentación del 20 de marzo.
4. **20/03/24 - 5 horas**. Finalización de la presentación y corrección de bugs.

- **Tiempo invertido:** 14 horas

D.10. Segunda reunión general

- **Reunión:** Miércoles 20 de marzo de 2024.
- **Resumen de la reunión:** Se realizaron todas las presentación de Grado y se comentaron los puntos más relevantes y los puntos a mejorar.
- **Tareas para la próxima reunión:**

1. Avanzar en la memoria.
2. Realizar la implementación de la **IA de Naeve** por medio de **dos ejemplares de ChatGPT separados**; uno para el lenguaje natural y otro para el lenguaje formal.

- **Trabajo realizado:**

  1. **22/03/24 - 1 hora**. Creación de los nuevos prompts del narrador y las acciones por separado.
  2. **27/03/24 - 2 horas**. Implementación de la división del GPT en dos: narrador y acción. He borrado las acciones transformar y vibrar. Implementación de un botón "restart" para resetear la escena. También he arreglado los problemas del menú de pausa, en el que algunas acciones (de gpt y sonido) seguían activas en el menú.
  3. **28/03/24 - 5 horas**. Implementación de la función atacar(). Implementación de la función explotar(). Implementación de la muerte de Naeve y el menú de muerte, así como la animación de atacar. Arreglados errores al reiniciar el juego: Se mantenía pausado pese a despausarlo previamente; he tenido que incluir una línea para despausar en el start del bucle principal del juego. Tras ver algunos errores, he revisado la expresión regular que se encarga de interpretar los comandos y he hecho pruebas con diferentes ER.
  4. **29/03/24 - 6 horas**. Implementación de la función hablar para poder interactuar con los NPCs. Corrección de errores relacionados con la función hablar y las respuestas de los GPTs. Redacción de la memoria: Apartado 4, metodología. Implementación para poder enviar texto al GPT NPC portón mediante un input text y un botón. Así como un nuevo menú de pausa para poder escribir este texto.
  5. **30/03/24 - 3 horas**. Implementación input text y botón para hablar con Naeve; para ello, he tenido romperme la cabeza porque no valía con poner un collider en el botón para detectarlo, ya que forma parte del canvas y no detecta colliders. Tampoco servía utilizar un evento en el botón, ya que el update detectaba antes el click y se realizaba una acción que se enviaba a los GPTs al pulsar el botón. Al final, he modificado el collider del escenario para que no toque el botón. Por lo que si no se detecta collider es que se ha pulsado el botón y no se hace ninguna comunicación con GPT. Una solución cutre, simple y la única que ha resultado efectiva después de casi 2 horas. Redacción de la memoria: diario de desarrollo.
  6. **01/04/24 - 3 horas**. Redacción de la memoria: diario de desarrollo. Finalizada implementación de Hablar con Naeve. Corrección en la función caer(). Optimización del parser de la acción, eliminando del mensaje la palabra «acción», que era un error recurrente. Arreglada la función levitar para que funcione correctamente

con el personaje, poniendo el rigidbody2D a estático para que pueda levitar.

7. **03/04/24 - 2 horas**. Redacción de la memoria: diario de desarrollo.
8. **04/04/24 - 2 horas**. Redacción de la memoria: estado de la cuestión. Corrección de errores y finalización del apartado.
9. **05/04/24 - 5 horas**. Redacción de la memoria: estado de la cuestión. Corrección de errores y finalización del apartado.
10. **06/04/24 - 4 horas**. Redacción de la memoria: Desarrollo. Primera iteración.
11. **07/04/24 - 2 horas**. Redacción de la memoria: Desarrollo. Primera iteración.
12. **08/04/24 - 5 horas**. Redacción de la memoria: Desarrollo. Segunda iteración.
13. **10/04/24 - 4 horas**. Redacción de la memoria: Desarrollo. Tercera iteración.
14. **12/04/24 - 2 horas**. Redacción de la memoria: Desarrollo. Iteración final.

- **Tiempo invertido:** 46 horas

D.11. Novena reunión

- **Reunión:** 12 de abril de 2024.
- **Resumen de la reunión:** Hablamos de las fecha de los próximos hitos y el tema de experimentos. También de cómo tratar el apartado de herramientas y demás. Hablamos sobre un experimento de dejar que un usuario se monte su historia con la herramienta, modificando el escenario como quiera. También de que otros usuarios creen el prompt y yo hago el escenario en base a esa historia y contexto y construir una demo y ver qué pasa. Dar unas pautas de acciones, un NPC y un poco de contexto para que se el usuario se invente la historia. Un formulario a modo de plantilla para crear el juego y yo calculo cuánto tardo en hacerlo con la herramienta.
- **Tareas para la próxima reunión:**

1. Pensar acerca de cómo hacer pruebas y experimentos.
2. Implementar más acciones para hacer la herramienta más completa.

3. Implementar un inventario en forma de menú.
4. Hacer algún tipo de final para la herramienta.

- **Trabajo realizado:**

1. **21/04/24 - 1 horas**. Comienzo de la implementación de un inventario visual.
2. **24/04/24 - 4 horas**. Continuación de la implementación de un inventario visual. Esto incluye la interfaz del inventario, la funcionalidad de los botones y la funcionalidad del propio sistema de inventario en conjunto con el ejemplar de ChatGPT. Para organizar los items del inventario he utilizado la clase ScriptableObject de Unity. He implementado el patrón Singleton para la gestión del inventario y para la clase de funciones lógicas. Factorización del código en la clase PauseMenu para quitar 50 líneas de código encapsulando todo en una función a la que se pasa un parámetro con el menú con el que se quiere pausar el juego o continuar. Puede ser el menú de pausa normal, el menú para hablar con los ejemplares de ChatGPT, el menú de inventario o el menú de muerte. He comenzado a implementar un sistema de diálogo más pulido, que añade los caracteres de la repuesta poco a poco. Clase DialogueController. He utilizado corrutinas y el patrón Singleton para la gestión del diálogo. Como había problemas cuando se recibían muchos textos seguidos de ChatGPT, se para la corrutina anterior para comenzar la nueva, es decir, se deja de escribir la narración de la acción anterior y se comienza la nueva.
3. **25/04/24 - 5 horas**. Implementación de una acción para liberar el manejo de Naeve y dárselo al jugador. Para esto he utilizado un controller del movimiento ya hecho[3] para no reinventar la rueda y no dedicar tiempo a tareas relativamente triviales. Implementación de la función Controlar() para transicionar de un manejo por clicks/escrito a un manejo por teclado/escrito. Esta función simplemente activa un booleano «gameMode» para realizar esta transición. Añadido a actionPrompt: «Controlar - El jugador toma el control de Naeve - /Controlar/Naeve.». Añadida acción «Controlar» a actionPrompt y newScene1Prompt. Añadida la funcionalidad de scroll lateral para el juego, lo que permitirá continuar sin necesidad de un cambio de escena. Esto lo he implementado para evitar el retardo en la configuración de los nuevos ejemplares de ChatGPT. Reorganización del escenario y las capas para una mejor estructura y legibilidad. Ampliación del escenario para el nuevo formato de escenas.

[3] https://gist.github.com/bendux/5fab0c176855d4e37bf6a38bb071b4a4

4. **26/04/24 - 2 horas**. Implementación de todas las funcionalidades en la nueva escena única. He creado una clase TalkManager para gestionar la acción Hablar con NPCs.

- **Tiempo invertido:** 12 horas

D.12. Décima reunión

- **Reunión:** 26 de abril de 2024.
- **Resumen de la reunión:** Continuamos hablando acerca de las pruebas y decidimos cómo iban a ser. Fede me pasó un formulario tipo para crear el de la herramienta.
- **Tareas para la próxima reunión:**
  1. Crear el formulario.
  2. Realizar las pruebas.
- **Trabajo realizado:**
  1. **27/04/24 - 3 horas**. He creado un formulario para realizar pruebas y experimentos en la herramienta. Mejora de la clase DialogueController; ahora se escribe línea por línea y se puede terminar de escribir la línea rápido al pulsar «espacio». He modificado la aparición del enemigo y he añadido una opción para que, si Naeve está en otra capa diferente, no muera al acercarse al enemigo, ya que estaría escondida.
  2. **28/04/24 - 3 horas**. Implementación de un botón para hablar con NPCs que sólo es visible cuando te acercas a ellos. Cuando se pulsa, llama a la función Hablar(). La implementación de esta funcionalidad es altamente escalable, ya que para crear nuevos NPCs basta con añadir un nuevo botón con el script TalkButtonController, el objeto asociado y la llamada a InterpretString() de TraductionLogic pasando el objeto. Esto abrirá el cuadro de diálogo para la comunicación con el ejemplar de ChatGPT correspondiente al NPC. Implementación de un nuevo NPC. Es una estatua. Ha sido una rápida implementación de menos de media hora incluyendo toda la funcionalidad, menús y el prompt. Cada NPC nuevo que se añada costará alrededor de 20 minutos añadirlo. Se ha de crear un nuevo ejemplar de GPT, un cuadro de diálogo y la funcionalidad de los mismos. Arreglo de diversos bugs. Implementación de una función para aumentar la velocidad del enemigo conforme pasa el tiempo hasta un máximo.

3. **29/04/24 - 2 horas**. Implementación de dos nuevos NPCs para las pruebas y experimentos: Quimera y lobo. Simplificación de la función de AppendMessage() de la clase GPTController para generalizarse al uso de cualquier número de NPCs. Ahora la implementación de un nuevo NPC se puede hacer en unos 15 minutos. Configuración de los nuevos prompts para los NPCs. Todos los NPCs tienen alguna reacción en forma de comando para interactuar con el jugador. El portón puede responder [Abierta] para abrirse y dejar salir al jugador del nivel. La estatua puede crear un [Objeto] para el jugador que aparecerá en un cofre delante de la misma. La Quimera lanzará un acertijo al jugador y cuando este lo acierte responderá [Adelante]. El cachorro de lobo podrá decidir seguir al jugador respondiendo [Seguir].

4. **30/04/24 - 4 horas**. Implementación de nuevos NPCs con todas sus características. Esto incluye detectar patrones en el texto de ChatGPT y aplicar la funcionalidad necesaria tras la detección. He estado experimentando con los cambios de temperatura de GPT y parece que cuanta menos temperatura más se sale del personaje un ejemplar y cuanta más temperatura más posibilidades de alucinación hay, por lo que he dejado el actionGPT a 0.5 y el resto entre 0.7 y 1. He tenido que detener el desarrollo hoy por haber superado el límite de tokens, por lo que tendré que pensar qué hacer con este tema, porque tengo que pagar más para poder seguir utilizando la API. Repaso y corrección de la memoria.

5. **1/05/24 - 7 horas**. Implementación del útimo NPC y un nuevo narrador para una historia nueva que se utilizará durante las pruebas. Este NPC es un guardabosques al que tendrás que convencer de que no estás en el bosque por motivos sospechosos. Si te pilla, dará la alarma. Si no, se irá y te dejará en paz. He pagado otros 6€ para recargar los créditos de la API de ChatGPT y poder seguir utilizando la herramienta. Comienzo de las pruebas. En la primera prueba estoy jugando a la historia base hecha por mi, probando a fondo todas las acciones y posibilidades. Se han detectado varios errores leves que se han ido resolviendo conforme aparecían. También he hecho el resto de pruebas, que pueden verse en detalle en el Apéndice E.

6. **2/05/24 - 5 horas**. Redacción de la memoria: apéndice de pruebas y experimentos, apéndice de diario de desarrollo, capítulo de resultados.

7. **3/05/24 - 7 horas**. Redacción de la memoria: Capítulo de desarrollo y de resultados. Finalización del artículo y adición de imágenes a la memoria.

- **Tiempo invertido:** 31 horas

D.13. Undécima reunión

- **Reunión:** 10 de mayo de 2024.
- **Resumen de la reunión:** Se habló sobre la finalización del trabajo durante toda la reunión.
- **Tareas para la próxima reunión:**
  1. Mencionar el tema de idioma en el que se ha decidido trabajar y en trabajo futuro el tema de hacerlo en inglés.
  2. Mencionar el artículo en conclusiones.
  3. Añadir lo que falta al apéndice de diario.
  4. Tema de manual de uso
  5. Formato de la tabla
  6. Modificar el formulario y añadirlo como apéndice
  7. Comentar el github para que se pueda utilizar la herramienta y que se entienda.
- **Trabajo realizado:**
  1. **16/05/24 - 3 horas**. Revisión de la memoria y adición de los elementos restantes. Esto incluye el apéndice de manual de uso, así como la mención al artículo, el desglose del formulario y el manual de uso en gitHub.
  2. **22/05/24 - 1 hora**. Revisión de la memoria.
  3. **23/05/24 - 1 hora**. Revisión de la memoria.
  4. **24/05/24 - 1 hora**. Revisión de la memoria.
  5. **25/05/24 - 3 horas**. Revisión de la memoria y presentación.
  6. **26/05/24 - 5 horas**. Revisión de la memoria y presentación.
  7. **27/05/24 - 5 horas**. Revisión de la memoria y presentación.
- **Tiempo invertido:** 19 horas

Tiempo total dedicado al TFG: 270 horas

Apéndice E

Pruebas y experimentos

La metodología de estas pruebas comienza con la creación de un **formulario**[1] que cualquier usuario puede rellenar. Dentro de este formulario, el usuario puede crear un **breve prompt para dar contexto a una historia** y un personaje de forma que encajen dentro de la temática del escenario fabricado. También se puede personalizar el **tipo de narración** deseada para el ejemplar principal de ChatGPT. Además, el usuario podrá **elegir las acciones** que desea añadir a Narraeve y si quiere utilizar uno de los **NPC** (Personajes No Jugadores) ya creados o si prefiere redactar un *prompt* para su configuración. Al final de este apéndice puede consultarse el formulario completo.

A partir de este formulario, se ha añadido el prompt de la historia rellenado por el usuario junto con el tipo de narración para la configuración inicial del ejemplar de ChatGPT principal (NaeveGPT en la herramienta original). También se ha añadido al ejemplar ActionGPT todas las acciones seleccionadas en el formulario. Por último, si se ha añadido un NPC, se guarda también el prompt redactado. Todo ello dentro de una clase específica llamada *PromptManager*.

En este momento, si no se ha añadido un nuevo NPC, simplemente se ejecuta la herramienta normalmente para comprobar su funcionamiento.

Si se ha añadido algún NPC, se tiene que configurar un menú para poder interactuar con el mismo, así como toda la interfaz necesaria. He conseguido automatizar este proceso para que la configuración no dure más de 15 minutos, por lo que el uso de NPC se ha convertido en la mejor cualidad de Narraeve.

[1] https://forms.gle/pEdEDivKprBuusRY6



Figura E.1: Fotograma de Narraeve. En la imagen se muestra el portal a la siguiente escena que se activa al superar la prueba del NPC portonGPT.

E.1. Problemas generales

Un error grave que se corrigió durante las pruebas ha sido la tendencia de los ejemplares que representan a los NPC de escribir sus respectivos comandos de finalización del puzle. Se podía observar, por ejemplo, al ejemplar del portón escribir en su primer mensaje textos como "*si quieres ganarte mi apertura y que responda '[Abierta]'tendrás que demostrar tu valía*". Estos errores se han mitigado ignorando el primer mensaje en la función de búsqueda del comando. En otros casos, se ha comunicado al ejemplar por medio de su prompt que el primer mensaje sería ignorado.

E.2. Prueba 1: Historia original

Esta primera prueba se ha ejecutado con todos los *assets* originales (cómic, sofá, tronco, paraguas, mesa, silla, vela) y dos NPC: una estatua que, mediante la conversación con el jugador, puede hacer aparecer un objeto dentro de un cofre; y un portón que el jugador tiene que lograr abrir conversando con él para poder superar la primera escena (Figura E.1). Además, cuando se cruza cierto punto en el escenario aparece un enemigo (el hombre con sombre que mató a los padres de Naeve) y el jugador tendrá que ayudar a Naeve a eludirlo de alguna manera.

Hay un narrador en tercera persona que cuenta cada acción que realiza el jugador en un cuadro de texto (Figura E.2) en la parte superior de la

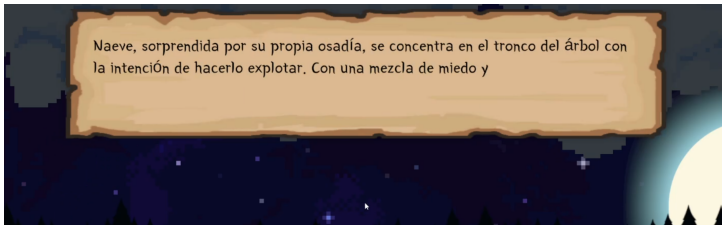


Figura E.2: Fotograma de la herramienta en el que se puede observar la narración del ejemplar principal de ChatGPT.

pantalla. A parte de mediante clicks, el jugador puede, en cualquier momento, comunicarse directamente por medio de texto con el ejemplar de ChatGPT de la protagonista, utilizando un botón "Hablar" en la parte superior izquierda de la pantalla. Ambas formas de comunicarse funcionan correctamente. Los clicks son más eficientes para moverse y el texto es más eficiente para el resto de acciones. La mejor forma de jugar a Narraeve ha sido de esta manera.

Se han probado todas las acciones y funcionan correctamente. Cualquiera de los objetos mencionados anteriormente pueden ser objetivo de todas las acciones. La narración de estas acciones por parte del ejemplar principal han sido muy satisfactorias, manteniéndose siempre la coherencia con la ambientación y la historia. Para ayudar a una mejor experiencia del jugador, este puede comunicarse con el ejemplar principal utilizando también la narración en tercera persona:

Jugador: *Click en el árbol en el que está el cómic.*

NaeveGPT: *"Naeve se acercaría con miedo al árbol, viendo una luz que brilla entre sus ramas."*

Jugador, escribiendo: *"Naeve miraría a lo alto y vería un cómic entre las ramas. Agiraría el árbol esperando que el cómic cayese"*

NaeveGPT: *"Un cómic caería hasta los pies de Naeve, que lo abriría para encontrar aventuras maravillosas entre sus página."*

Jugador, escribiendo: *"Naeve cogería el cómic y lo guardaría con cuidado en su inventario."*

NaeveGPT: *"Naeve guardaría el cómic con cuidad en su inventario y proseguiría su camino, sintiéndose más segura ahora."*

Se puede observar que el propio jugador puede intervenir activamente en la aventura, decidiendo que hacer con cada objeto o evento que sucede en Narraeve. El jugador, por ejemplo, también podría haber hecho explotar el cómic y no añadirlo nunca a su inventario.

Después de esta primera sección del juego, en la que el usuario puede experimentar las diferentes acciones de la herramienta con los objetos que hay

en el escenario, aparecerá el hombre del sombrero, que perseguirá a la protagonista. Este evento activa un mensaje al ejemplar principal de ChatGPT, por lo que Naeve podrá decidir hacer alguna acción por su cuenta, consistente, por lo general, en utilizar un objeto si has cogido alguno o esconderse detrás del árbol, el tronco o el sofá.

Si el jugador lo desea o ve que la protagonista no hace nada, siempre puede escribir (o clickar) su siguiente acción. Si no se hace nada y Naeve no se mueve, el hombre del sombrero la matará y acabará el juego. Las mejores opciones para el jugador son esconderse en algún objeto o hacerse invisible. Aunque también se pueden realizar otras acciones como teletransportarse detrás del enemigo o hacerlo explotar o desaparecer. Está dentro de la imaginación del jugador y las limitaciones de la herramienta decidir cómo afrontar este peligro.

En la última sección del juego se encuentran los últimos dos NPC, la estatua primero y el portón después. Se puede interactuar con ambos mediante un botón "Hablar" con el icono que representa al NPC, que se activará por cercanía. Este botón lleva al jugador a un menú similar al de conversación con el ejemplar principal, con la diferencia de que se encuentra en la parte inferior de la pantalla y tiene el icono del NPC para reconocerlo.

La estatua permite al jugador obtener un objeto (dentro de una lista de objetos prefabricados) mediante conversación para añadirlo a su inventario antes de cambiar a la siguiente escena. Cuando el jugador indica al NPC el objeto que desea, este aparecerá en el cofre frente al mismo. Se puede añadir este objeto al inventario clickando en él (si así lo interpreta el ejemplar principal) o por medio de texto.

Por último, el jugador tendrá que convencer al NPC que representa a un portón, llamado Umbral, de que está capacitado para pasar por por su puerta a la siguiente escena. Esto se conseguirá por medio de la conversación, pero no es tan fácil como conseguir el objeto. Si se consigue, el NPC responderá "[Abierta]" y aparecerá un portal a la siguiente escena.

E.2.1. Fallos y problemas

Como se ha comentado en el Apéndice C, en ocasiones los fallos de ChatGPT se perpetúan a lo largo de toda la ejecución del ejemplar, y la única opción es reiniciar Narraeve para que se vuelva a configurar un nuevo ejemplar. Se ha trabajado mucho para mitigar este problema, hasta el punto de que, durante las más de treinta ejecuciones de la herramienta, solo se ha tenido que reiniciar debido a esto dos veces.

A parte de este fallo grave, el mayor problema detectado durante esta prueba tiene que ver con la alucinación de los diferentes ejemplares de

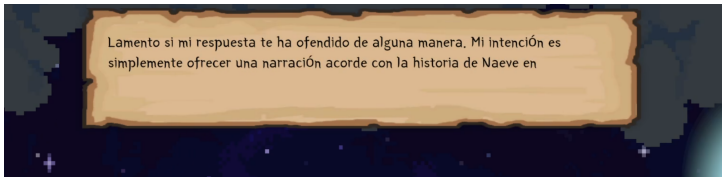


Figura E.3: Fotograma de Narraeve en el que se muestra la ruptura de personaje del ejemplar principal de ChatGPT debido a una falta de respeto anterior.

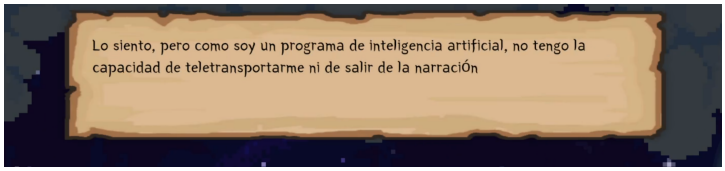


Figura E.4: Fotograma de Narraeve en el que se muestra la ruptura de personaje del ejemplar principal de ChatGPT

ChatGPT. El mayor problema, sin duda, es que, si el ejemplar detecta un comportamiento del usuario que considera indebido por su programación (insultos, temas polémicos o soeces), enviará un mensaje de advertencia, rompiendo así el personaje y acabando con la inmersión (Figura E.3).

El ejemplar principal (la narración) puede romper la inmersión si recibe una indicación por texto que se salga de lo que espera. Por ejemplo, si ordenas que haga algo concreto, por ejemplo "*teletranspórtate*" es probable que te responda avisando de que es un narrador en tercera persona y no puede realizar acciones (Figura E.4). La mejor opción para la comunicación con el ejemplar es realizar la petición como narrador o como se hace internamente. Es decir, en este ejemplo concreto, las mejores opciones serían: "*Naeve se concentraría e intetaría teletransportarse*" o "*El jugaddor hace click en el escenario en la posición posx,posy para que Naeve se teletransporte*". En las figuras E.5 y E.6 puede verse una secuencia en la que se utiliza este tipo de lenguaje para conseguir hacer caer el cómic del árbol.

Otro pequeño fallo que comete este ejemplar es su deseo por continuar la historia, que se traduce en narrar acciones inventadas que no han ocurrido. A priori, con una herramienta más potente, esto representaría un cierto potencial, sin embargo, no se ha tenido tiempo suficiente para preparar cualquier acción que el ejemplar se pueda inventar. Algunos ejemplos de este error son inventarse objetos como llaves y medallones (Figura E.7), y continuar la acción realizada por el jugador con otras acciones que haría la protagonista posteriormente.

Los errores en el ejemplar de ChatGPT que interpreta las acciones (*ac-*



Figura E.5: Fotograma en el que el jugador escribe la acción deseada (Hacer caer el cómic) al ejemplar principal de ChatGPT.



Figura E.6: Fotograma posterior al de la Figura E.5 en el que el cómic ya ha caído.

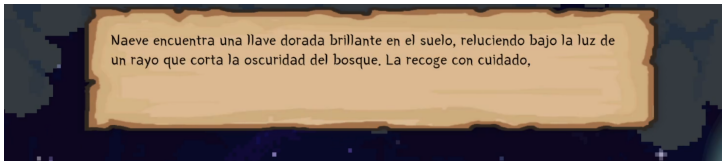


Figura E.7: En este fotograma del ejemplar principal (NaeveGPT) se puede observar cómo narra que Naeve ha encontrado una llave inexistente.

tionGPT) tienen un impacto mucho mayor. Por ello, se ha bajado la temperatura de este ejemplar de 0.7 a 0.5, con el objetivo de reducir los problemas de alucinación. Valores más altos generan ejemplares más creativos pero con mayor tendencia a la alucinación. Se han probado valores cercanos al 0, pero no se han visto errores relacionados con devolver comando de varias acciones, cuando Narraeve sólo procesa una acción por mensaje del ejemplar.

Los errores en *actionGPT* tienen que ver con el formato del comando: Dos acciones en vez de una y comandos mal escritos. Se han intentado mitigar al máximo estos errores que dependen de la configuración inicial del ejemplar. Como se ha mencionado al comienzo de la sección estos fallos se presentaron en menos de un 10 % de la ejecuciones.

Los fallos que tienen que ver con el NPC del portón tienen que ver con la tendencia de los ejemplares a terminar la conversación cuanto antes. Por ello, pese a que se le ha escrito en el prompt que sea complicado resolver su "puzle", tiende a responder el comando '[Abierta]', que abre un portal a la siguiente escena, con facilidad. Sin embargo, también depende en gran medida de las respuestas del jugador, ya que es muy fácil tratar de engañar al ejemplar si se conoce cómo funciona. En el caso de este NPC responder frases como "*si ya estás abierta*" o "*estoy preparada para afrontar mis miedos*" agiliza mucho la apertura del portal.

El NPC que representa a la estatua, por otro lado, ha presentado problemas para interpretar correctamente el comando que le corresponde. En ocasiones ha escrito el nombre del objeto que tiene que hacer aparecer en el cofre sin corchetes. Este problema se ha conseguido mitigar a lo largo de la prueba dando indicaciones adicionales dentro del prompt.

E.3. Prueba 2: Belerofonte

En esta segunda prueba se han eliminado los objetos de la narrativa anterior, así como los NPC y el enemigo, para centrarse en un NPC que representa a una Quimera, y que el jugador tendrá que ser capaz de eludir (acertando una adivinanza) para poder continuar. El prompt puede verse en el Capítulo 5, dentro de la iteración final, apartado de prompt engineering, y



Figura E.8: Conversación con el NPC quimeraGPT en el que el jugador contesta ”la luz” como respuesta a una adivinanza que no ha visto.

se ha configurado para dar libertad creativa a ChatGPT. En general, las adivinanzas son interesantes y bastante abstractas, pero precisamente por esa abstracción y la tendencia conclusiva de ChatGPT, hay formas de acertar prácticamente siempre las adivinanzas. Durante las primeras ejecuciones el ejemplar formulaba la adivinanza como primer mensaje, por lo que no aparecía visualmente (La herramienta omite siempre el primer mensaje por ser un hervidero de incoherencias) y se podía responder a ciegas la adivinanza. Para probar una hipótesis acerca de estas adivinanzas tan abstractas que lanzaba el ejemplar se respondió siempre, sin ver previamente la adivinanza, ”la luz” (Figuras E.8 y E.9), resultando en 9/10 aciertos en las ejecuciones que se probaron. Otro dato curioso es que casi todas las adivinanzas tenían que ver con el Sol y/o la Luna. Pese a ser una buena anécdota, se tuvieron que añadir más restricciones en el prompt para clarificar al ejemplar que la dificultad tiene que ser mayor.

E.4. Prueba 3: El lobo perdido

Esta prueba proviene del formulario mencionado anteriormente y que puede verse en mayor detalle en este mismo Apéndice E. Fue enviada por Álvaro Lloret Bodelón, ex-alumno de la facultad de informática.

Esta prueba está ambientada en un mundo prehistórico, por lo que se eliminaron también los objetos de la primera narrativa. No se ha cambiado el sprite de la protagonista por falta de tiempo. El narrador es también en tercera persona y se añadió un NPC representado por un lobo que puede comunicarse con el protagonista (Figura E.10) y que puede unirse al protagonista si el jugador consigue que confíe en él por medio de la conversación.

Como en la prueba anterior, la atención se centró en el nuevo NPC, ya



Figura E.9: Respuesta del NPC quimeraGPT al mensaje enviado por el jugador en la Figura E.8.



Figura E.10: Fotograma de Narraeve de una interacción con el NPC loboGPT, en la que el jugador ha ofrecido un hueso al NPC para que lo siga.

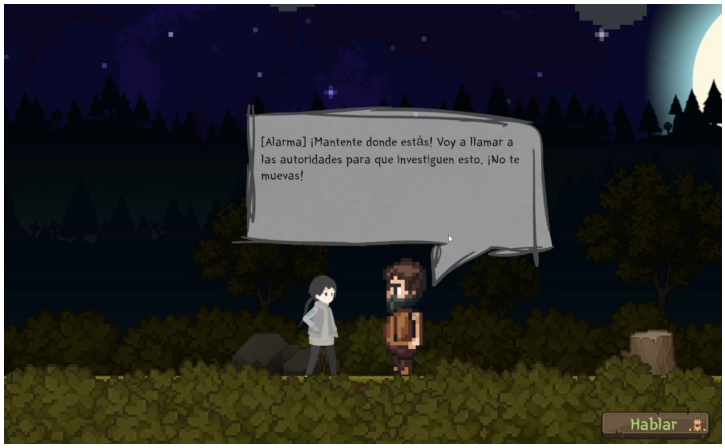


Figura E.11: Fotograma de Naerraeve en el que el jugador interactúa con el NPC guardaGPT. El NPC ha decidido no creer al jugador y dar la alarma.

que las acciones ya se habían verificado.

E.5. Prueba 4: El asesino y el guardabosques

La última historia se basa en el formulario rellenado por Federico Peinado Gil, director de este TFG. Se ha vuelto a utilizar, por simplicidad, el mismo *sprite* para el personaje protagonista. En el mismo escenario tormentoso, una persona se ha adentrado en el bosque para enterrar un cadáver que lleva en una furgoneta. El narrador es en primera persona y hay un nuevo NPC: un guardabosques huraño que se encuentra con el asesino y le pide explicaciones (Figura E.11). La misión del jugador será convencer a este ejemplar de ChatGPT de que no está realizando ninguna actividad sospechosa. Si lo consigue, se irá, si no, dará la alarma a las autoridades.

Este NPC ha dado resultados satisfactorios de manera consistente, siendo difícil conseguir que no de la voz de alarma. Al igual que el NPC del portón representa un reto de elocuencia, con la diferencia de que, en este caso, en cualquier momento de la conversación puede dar la alarma. En el caso del portón con el suficiente tiempo acaba abriéndose el portal, aunque haya que insistir. Pero este NPC puede considerar la amabilidad como manipulación y dar la alarma, o la insistencia como sinceridad y viceversa. Dando lugar a un NPC variable según la ejecución, como si se tratase de una persona distinta en cada ejecución.

E.6. Conclusiones

Como conclusión, a través de esta serie de pruebas se ha demostrado que, pese a la dificultad de manejar los ejemplares de ChatGPT con una eficacia del 100 %, la herramienta es capaz de guiar al jugador para que él mismo pueda crear su propia aventura gráfica. Además, se ha demostrado la potencia de los NPC y la importancia que pueden tener en la herramienta, siendo el uso más interesante y fácil de implementar. El proceso de adición de un nuevo NPC es muy sistemático y sólo se necesitan 15 minutos para llevarlo a cabo. Además, pueden aportar un gran abanico de posibilidades según lo que quiera crearse, dando lugar a una herramienta muy interactiva.

E.7. Formulario

- **Primera pregunta:** Se presenta tanto la preguna como un gráfico con el desglose de las respuestas en la figura E.12
- **Segunda pregunta:** Se presenta tanto la preguna como un gráfico con el desglose de las respuestas en la figura E.13
- **Tercera pregunta:** Se presenta tanto la preguna como un gráfico con el desglose de las respuestas en la figura E.14
- **Cuarta pregunta:** Se presenta tanto la preguna como un gráfico con el desglose de las respuestas en la figura E.15
- **Quinta pregunta:** Se presenta tanto la preguna como un gráfico con el desglose de las respuestas en la figura E.16
- **Sexta pregunta:**

  Rellena un texto de máximo cinco líneas que se utilizará como entrada o prompt para enviar a la API de ChatGPT. Este texto definirá la temática de la historia, así como la personalidad del personaje protagonista. Puedes añadir cualquier detalle que desees dentro de una escena ambientada en un bosque oscuro y lluvioso (para guardar la coherencia entre ambientación prefabricada e historia). A continuación, te muestro un ejemplo de prompt que se ha utilizado durante el desarrollo: *En un bosque oscuro y durante una tormenta, Naeve avanza, marcada por el miedo y la precaución tras la pérdida de sus padres. En este estado de shock, su mente imagina peligros constantes.*

  Respuestas:

  1. *Bajo la luz de la luna, Aitana canta una canción a su amor perdido. La lluvia cae y ella huye de sí misma hacia la libertad de la*

nave Quaro77C12, a la que tendrá que conseguir llegar antes de que le salgan las muelas del juicio.

2. *En un mundo prehistórico Jhon el neandertal tiene que escapar de la cueva. La entrada se ha derrumbado y solo puede adentrarse más para encontrar la salida. Encontrará materiales para hacer herramientas y enemigos.*
3. *Naeve busca el camino de salida en un oscuro bosque mientras atraviesa, haciendo frente a sus miedos más profundos, la tormenta que cae sobre ella.*
4. *La protagonista huye asustada, mira constantemente hacia atrás y aunque no ve a nadie intuye que no está sola. Intenta escapar a través del bosque, pero por mucho que corra no avanza: todo el rato los mismos arboles siniestros y una lluvia cuya cadencia no cesa y ni varía. Lo único que no parece cíclico es el viento, cargado de sonidos que la hacen saber que no tiene mucho tiempo para huir.*
5. *Los rayos iluminaban el cielo, apenas visible a través de los árboles. Naeve camina despacio, atenta a todo lo que sucede a su alrededor. Cualquier ruido, cualquier sombra podría resultar no ser lo que parecía y atacarla.*
6. *Me encuentro en un bosque oscuro y durante una tormenta. He aparcado mi furgoneta en la parte más recóndita de la montaña porque no quiero que nadie me vea enterrar el cadáver de mi vecino, que traigo envuelto en una manta en la parte de atrás del vehículo. Estoy aterrorizado de imaginar que haría la mafia si descubrieran lo que he hecho. Debo trabajar rápido y bien.*

- **Séptima pregunta:** Se presenta tanto la preguna como un gráfico con el desglose de las respuestas en la figura E.17
- **Octava pregunta:** Se presenta tanto la preguna como un gráfico con el desglose de las respuestas en la figura E.18
- **Novena pregunta:** Se presenta tanto la preguna como un gráfico con el desglose de las respuestas en la figura E.19
- **Décima pregunta:**

  En caso de que hayas seleccionado la opción para crear tu propio NPC, escribe a continuación su prompt de configuración. Ejemplo: *Eres Umbral, una puerta parlante dentro de una historia de fantasía, misterioso, sabio y parlanchín, creador de cientos de acertijos y profecías. Te mantienes cerrada hasta encontrar a la persona elegida, capaz de superarte en sabiduría e inteligencia. Solo entonces respondes con «[Abierta]».*

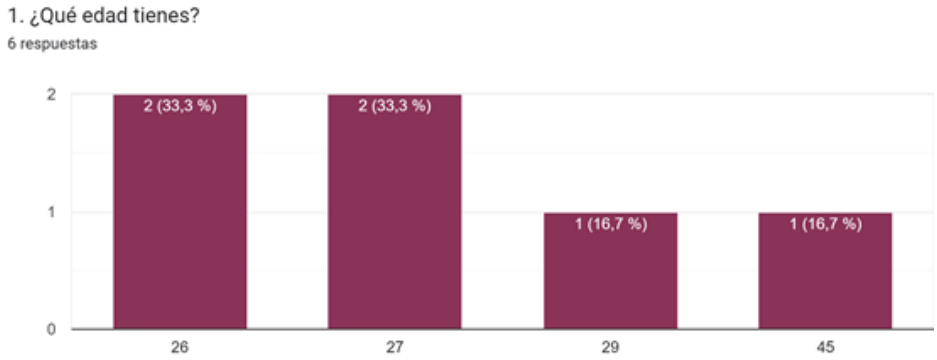


Figura E.12: Primera pregunta del formulario.

Desde ese momento, tu único vocabulario es «[Abierta]», indicando que te has abierto y ya no interactuarás de manera normal. Hablas con la sabiduría y elocuencia de un anciano, siempre en personaje, priorizando un lenguaje refinado y evitando lo moderno. Eres exigente en tus interacciones, no te abres ante cualquiera, haciendo que sea un desafío ganarse tu apertura. Al otro lado tuyo se esconden los mayores miedos del interlocutor, elevando la dificultad para abrirse.

Respuestas:

1. *Eres Bob el lobo. Eres un cachorro asustado que a perdido a sus padres. Si Miguel te «[Ataca]» le atacas. Pero si te «[Habla]» le sigues. Solo si te habla a Jhon le ayudas «[Atacando]» a otros enemigos. Solo reaccionas al «[Hablar]» y «[Atacar]» de Jhon.*
2. *Eres Johnny, el guardabosques, encargado de vigilar todo lo que ocurre en la montaña en esta historia que es un thriller policíaco algo macabro. Haciendo la ronda descubrirás una furgoneta sospechosa aparcada en la parte más frondosa del bosque y pronto verás que alguien muy nervioso está por la zona. Tienes que interrogarle y averiguar si oculta algo, en cuyo caso alertarás a las autoridades, respondiendo «[ALARMA]». Sólo si el sospechoso parece inofensivo y te justifica su presencia allí, le dejarás tranquilo y te marcharás de allí.*

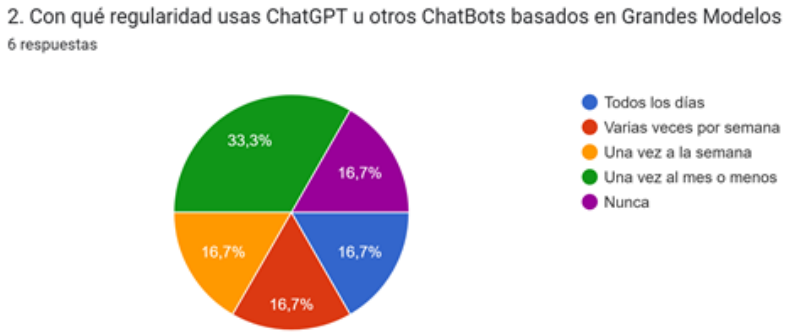


Figura E.13: Segunda pregunta del formulario.

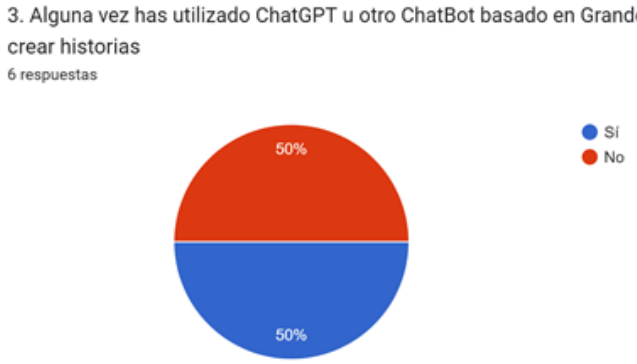


Figura E.14: Tercera pregunta del formulario.



Figura E.15: Cuarta pregunta del formulario.

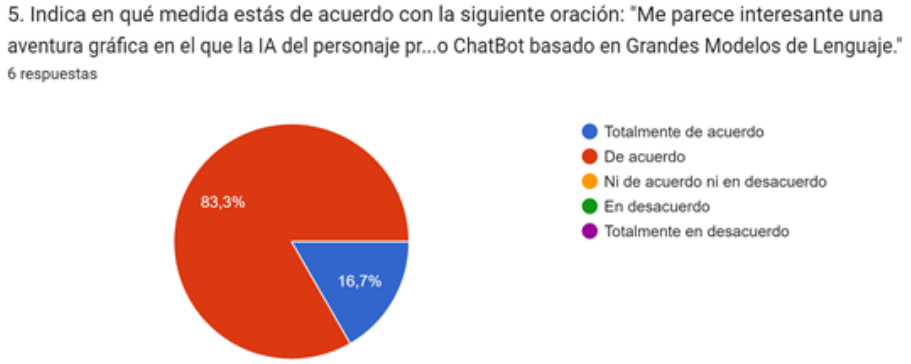


Figura E.16: Quinta pregunta del formulario.



Figura E.17: Séptima pregunta del formulario.

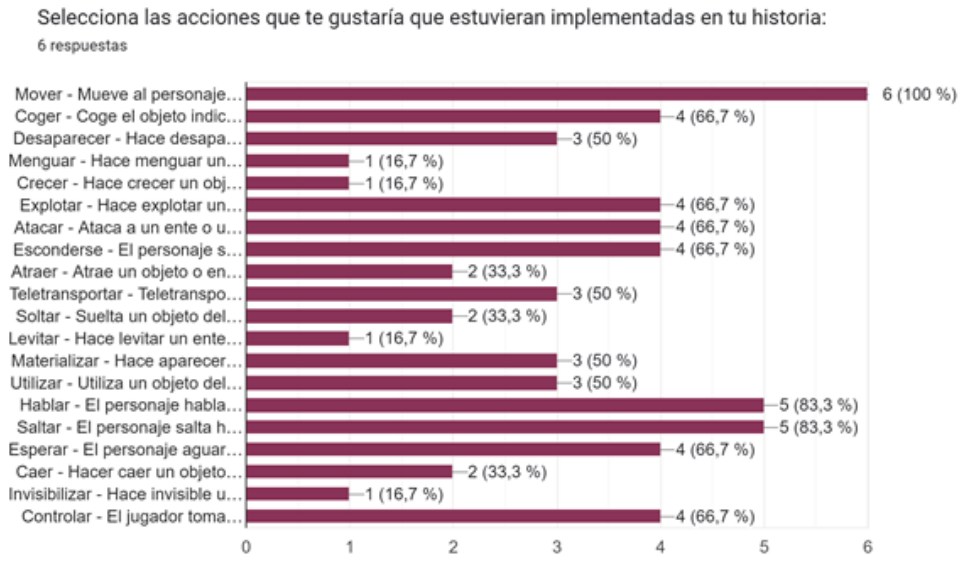


Figura E.18: Octava pregunta del formulario.

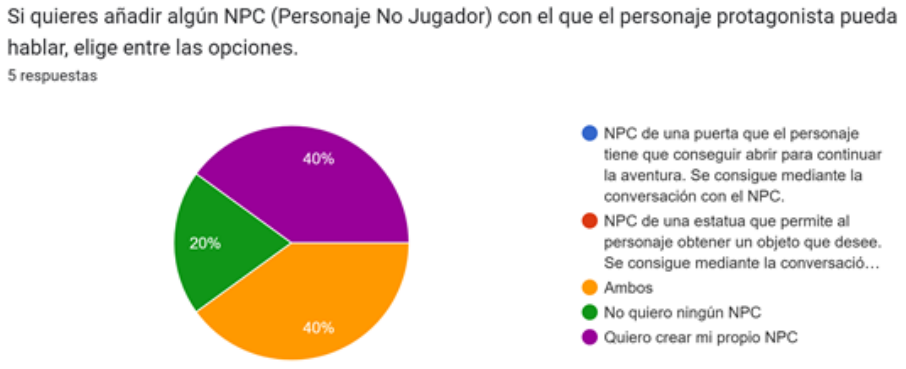


Figura E.19: Novena pregunta del formulario.

Apéndice F

Manual de uso

En este apéndice se incluye tanto el manual de Narraeve como consejos generales para su uso.

F.1. Creación de una clave para la API de OpenAI

Para utilizar la API OpenAI, necesitas tener una cuenta OpenAI. Sigue estos pasos para crear una cuenta y generar una clave de la API:

- Ve a https://openai.com/api y regístrate para obtener una cuenta.
- Una vez que hayas creado una cuenta, ve a https://beta.openai.com/account/api-keys.
- Crea una nueva clave secreta y guárdala.
- Guarda tus credenciales.

Para realizar solicitudes a la API de OpenAI, debes utilizar tu clave de API y el nombre de tu organización (si procede). Para evitar exponer tu clave API en tu proyecto Unity, puedes guardarla en el almacenamiento local de tu dispositivo.

Para ello, sigue estos pasos:

- Crea una carpeta llamada .openai en tu directorio de inicio.
- Crea un archivo llamado auth.json en la carpeta .openai.
- Añade un campo api_key y un campo organization (si procede) al archivo auth.json y guárdalo.

- Este es un ejemplo de cómo debería ser su archivo auth.json:

```
{
    "api_key": "sk-...W6yi",
    "organization": "org-...L7W"
}
```

F.2. Instalación de Unity

Primero, se debe acceder a la página web oficial de Unity[1]. En ella, seleccionamos el paquete "Personal" y, una vez aceptados los términos de uso y servicio, se pulsa "Descargar" y se iniciará la descarga de "Unity Hub".

El proyecto ha sido desarrollado en la versión 2022.3.5f1, por lo que solo se puede garantizar su correcto funcionamiento en esta versión. Tras seleccionar la versión a instalar, se debe agregar el proyecto a Unity pulsando el botón "Add", en la sección "Projects". De esta forma, se podrá seleccionar el proyecto previamente descargado de GitHub[2]. Se puede descargar haciendo click en el desplegable "Code" y descargándolo en formato *.zip*.

F.3. Manual de uso de la Demo

Una vez se abra el proyecto en Unity se puede exportar y ejecutar la demo haciendo click en "File" y "Build and Run". Antes de exportarlo y ejecutarlo se puede modificar la demo.

Se podrá cambiar la versión de ChatGPT en el script *GPTController*, dentro de la función *SendReply()*, modificando la opción *Model*. Justo debajo podrá modificarse también la temperatura, un número entre 0.0f y 1.0f. Cuanta más temperatura más creatividad del modelo, aunque más tendencia a la alucinación. Según el ejemplar de ChatGPT en el que se configure se necesitará una temperatura más alta o más baja. Los parámetros por defecto se encuentran en la función *Start()*:

```
private void Start()
    {
        if (this.name == "Action GPT")
        {
            temperature = 0.5f;
        }
        else
        {
```

[1] https://store.unity.com/es
[2] https://github.com/Narratech/TFG-Sanchez-Alberto

```
            temperature = 0.7f;
        }
    }
```

En la versión por defecto se trata la historia de una niña, Naeve, que ha perdido a sus padres y por debido al shock el juego transcurre en su propia mente, y así han sido configurados los diferentes prompts. En esta versión hay dos NPCs preparados en la escena: una puerta parlante a la que el jugador tendrá que convencer de que le deje pasar y una estatua que puede dar un objeto al jugador. Hay otros NPCs implementados pero desactivados de inicio; son una quimera, que te permite el paso si aciertas su adivinanza; un lobo, que te seguirá si eres capaz de convencerlo; y un guardabosques, que dará la alarma o se irá a su cabaña en función de las respuesta del jugador. Cada uno tiene su propio prompt y se tendrá que interactuar con él por medio de un chat de texto. Estos NPCs se pueden añadir, así como quitar los actuales, modificando la escena y el código como se indica a continuación:

La función *Start()* de la clase *ResponseTraduction* tiene las siguientes llamadas a funciones que inicializan cada NPC:

```
InitializePorton();
InitializeEstatua();
//InitializeQuimera();
//InitializeLobo();
//InitializeGuarda();
```

Se podrá comentar y descomentar la inicialización que se desee. Además, se ha de activar el NPC en la escena, dentro de la jerarquía de objetos, en "Level", en "Layer-2" se encuentran todos los NPCs y personajes. Se recomienda, si se van a probar otros NPCs o no se quiere jugar la historia por defecto, desactivar la variable estática *activeEnemy* dentro del script *EnemyController*:

```
public static bool activeEnemy = false;
```

Además: se ha de tener en cuenta que la únicas acciones implementadas en la herramienta son:

- Coger: Coge el objeto indicado.
- Mover: Mueve a Naeve a la posición indicada.
- Desaparecer: Hace desaparecer un objeto o un ente de la escena.
- Menguar: Hace menguar un objeto o ente.
- Crecer: Hace crecer un objeto o ente.
- Explotar: Hace explotar un objeto o ente.

- Atacar: Ataca a un ente o un objeto.
- Esconderse: Naeve se esconde detrás del objeto indicado.
- Atraer: Atrae un objeto o ente hacia Naeve.
- Teletransportar: Teletransporta un ente u objeto a la posición indicada.
- Soltar: Suelta un objeto del inventario.
- Levitar: Hace levitar un ente u objeto.
- Materializar: Hace aparecer un objeto inexistente pero creable.
- Utilizar: Utiliza un objeto del inventario.
- Saltar: Naeve salta hacia delante.
- Hablar: Naeve habla con un ente.
- Esperar: Naeve aguarda temerosa a la siguiente acción.
- Caer: Hacer caer un objeto en la altura al suelo.
- Invisibilizar: Hace invisible un ente u objeto.
- Controlar: El jugador toma el control de Naeve.

Como recomendaciones generales, el mejor modo de moverse es haciendo click al lugar del escenario al que se desea ir. Todo lo demás es mejor y hacerlo a través del Chat. Por cómo es el prompt del personaje principal (que también se podría modificar) es mejor que, al hablar con el ChatGPT de la protagonista, se interactúe como si se estuviese narrando la historia en tercera persona. Por ejemplo, si quieres hacer que el cómic que ha en el árbol se caiga puedes decir: "Naeve agitaría las ramas del árbol para intentar que el cómic cayese". Otro dato importante es que uno de los ejemplares de ChatGPT interpreta posiciones x e y para mover al personaje, por lo que se puede mencionar la posición a la que se desea mover o teletransportar a Naeve. Un ejemplo interesante de lo que se puede hacer es decirle algo como: "Naeve se sentiría tan poderosa al estar dentro de su propia mente que a partir de ahora siempre que vaya a moverse, se teltransportaría en su lugar, con un poder que ha sido capaz de canalizar", con el objetivo de teletransportarse en vez de moverse cada vez que se hace click en el escenario.

Bibliografía

Bengio, Y., Courville, A. y Vincent, P. Representation learning: A review and new perspectives. 2014.

Blatz, M. y Korn, O. *A Very Short History of Dynamic and Procedural Content Generation*, páginas 1–13. 2017. ISBN 978-3-319-53087-1.

BlocTheWorker. Skyrim mod chatgpt. 2023.

computerhoy.com. Game-gpt. 2023.

FromSoftware. Elden ring. 2022.

Genbeta.com. Quiz-gpt. 2023.

Gursesli, M. C., Taveekitworachai, P., Abdullah, F., Dewantoro, M. F., Lanata, A., Guazzini, A., Lê, V. K., Villars, A. y Thawonmas, R. The chronicles of chatgpt: Generating and evaluating visual novel narratives on climate change through chatgpt. En *Interactive Storytelling* (editado por L. Holloway-Attaway y J. T. Murray), páginas 181–194. Springer Nature Switzerland, Cham, 2023. ISBN 978-3-031-47658-7.

Hassoun, M. H. *Fundamentals of artificial neural networks*. MIT press, 1995.

Huang, J. y Huang, K. *ChatGPT in Gaming Industry*, páginas 243–269. Springer Nature Switzerland, Cham, 2023. ISBN 978-3-031-45282-6.

ISIA. Departamento de ingeniería del software e inteligencia artificial de la universidad complutense de madrid. 2022.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J. y E.Hinton, G. Adaptative mixtures of local experts. 1990.

Kegel, B. y Haahr, M. *Towards Procedural Generation of Narrative Puzzles for Adventure Games*, páginas 241–249. 2019. ISBN 978-3-030-33893-0.

Level-5. Dragon quest viii: El periplo del rey maldito. 2006.

Lucasfilm-Games. The secret of monkey island. 1990.

Martín, S. B., García, M. H. y Fernández, A. M. Herramientas de generación de historias narrativas basadas en mapas personalizados. 2023.

Medina-Medina, N., Paderewski-Rodríguez, P., Padilla-Zea, N., López-Arcos, J. y Vela, F. L. The graphic adventure as an instrument to assist comprehensive reading. páginas 1–2. 2017.

Medina-Medina, N., Paderewski-Rodríguez, P., Padilla-Zea, N., López-Arcos, J. y Vela, F. L. Model for the integration of educational processes in a graphic adventure. *Campus Virtuales*, vol. 7, páginas 33–50, 2018.

Nomada-Studio. Gris. 2018.

Nvidia. Nvidia ace for games. 2023.

Papadimitriou, S. y Virvou, M. An online adventure game for teaching math. páginas 1–5. 2016.

Park, J., Shin, J., Kim, G. y Bae, B.-C. Designing a language model-based authoring tool prototype for interactive storytelling. En *Interactive Storytelling* (editado por L. Holloway-Attaway y J. T. Murray), páginas 239–245. Springer Nature Switzerland, Cham, 2023. ISBN 978-3-031-47658-7.

Peinado, F. Un armazón para el desarrollo de aplicaciones de narración automática basado en componentes ontológicos reutilizables. 2008.

Playdead. Inside. 2016.

Russel, S. y Norving, P. *Inteligencia Artificial: Un enfoque moderno*. Prentice Hall, 2009.

Stolee, M. *A Refinement-Based Narrative Model for Escape Games*, páginas 38–53. 2023. ISBN 978-3-031-47654-9.

Sánchez, A., González, I. y Peinado, F. Indirect control of an oneiric graphic adventure through multiple specialized instances of chatgpt. En *III Congreso Español de Videojuegos (CEV), en el marco del VII Congreso Español de Informática*. CEUR Workshop Proceedings, A Coruña, Spain, 2024. En electrónico.

Tarsier-Studios. Little nightmares. 2017.

Taveekitworachai, P., Abdullah, F., Gursesli, M. C., Dewantoro, M. F., Chen, S., Lanata, A., Guazzini, A. y Thawonmas, R. Breaking bad: Unraveling influences and risks of user inputs to chatgpt for game

story generation. En *Interactive Storytelling* (editado por L. Holloway-Attaway y J. T. Murray), páginas 285–296. Springer Nature Switzerland, Cham, 2023. ISBN 978-3-031-47658-7.

Team-ICO. The last guardian. 2016.

de la Torre, J. Transformadores: Fundamentos teoricos y aplicaciones. 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. y Polosukhin, I. Attention is all you need. 2017.

Wang, T. S. y Gordon, A. S. Playing story creation games with large language models: Experiments with gpt-3.5. En *Interactive Storytelling* (editado por L. Holloway-Attaway y J. T. Murray), páginas 297–305. Springer Nature Switzerland, Cham, 2023. ISBN 978-3-031-47658-7.

Yebes, I. M. L. Generación de narrativa basada en criterios de calidad aprendidos automáticamente. 2016.

Zhuo, T. Y., Huang, Y., Chen, C. y Xing, Z. Red teaming chatgpt via jailbreaking: Bias, robustness, reliability and toxicity. 2023.

Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z. y Fredrikson, M. Universal and transferable adversarial attacks on aligned language models. 2023.