TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Juegos de aventuras conversacionales mejorados con sistemas de diálogo.

Autor: José Antonio Laserna Beltrán

Grado: Grado en Ingeniería Informática

Director: Arturo Montejo Ráez
Departamento del director: Departamento de informática

Fecha: 01/07/2024

CREA

(Página intencionalmente en blanco)

Universidad de Jaén

Departamento de Informática

Don Arturo Montejo Ráez y, tutor del Trabajo Fin de Grado titulado: '**Juegos de aventuras conversacionales mejorados con sistemas de diálogo**', que presenta Don José Antonio Laserna Beltrán, otorga el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Julio de 2024

El alumno:	El tutor:
José Antonio Laserna Beltrán	Firmado por MONTEJO RAEZ ARTURO - ***2653** el día 02/07/2024 con un certificado emitido Arturo Montejo Ráez

(Página intencionalmente en blanco)

Agradecimientos

A mi abuela Matilde.

A mi madre Matilde.

A mi padre, tocayo y mentor.

A Mara, Jose, Mario, Sky, Lola y Jose Manuel, mi buen vecino.

Y, por último, todos los que han ayudado en la elaboración de este trabajo

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Tecnologías de la Información
Mención (solo TFG)	Sin mención
Idioma	Español
Tipo	Específico
TFT en equipo	No
Autor/a	José Antonio Laserna Beltrán
Fecha de asignación	03/11/2023
Descripción corta	Las aventuras conversacionales son un género de juegos por ordenador con una larga historia. El proyecto propone la mejora de la experiencia de juego mediante la aplicación de sistemas de diálogo avanzados que posibiliten mayor libertad de expresión en el proceso de comunicación jugador-juego. Para ello se estudiará la viabilidad de sistemas basados en intenciones dentro de la estrategia conversacional.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

NORMAS UTILIZADASCOMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
Estas normas se han utilizado como base o referencia para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como proyecto la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.	

Contenido

Índice de ilustraciones

Índice de tablas

1 ESPECIFICACIÓN DEL TRABAJO

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado **¡Error! No se encuentra el origen de la referencia.** (**¡Error! No se encuentra el origen de la referencia.**).

1.1 Introducción

Las aventuras conversacionales fueron uno de los primeros formatos de videojuegos que existieron, siendo la primera aventura conversacional (o al menos, la primera con renombre) *Colossal Cave Adventure*, lanzada en 1976 por Will Crowther y ampliada posteriormente por Don Woods. Estos videojuegos en su forma más pura se basan exclusivamente en el texto. El jugador dispone de una serie de verbos con los que interactuar con el mundo del juego, recibiendo de él las descripciones y narraciones que le permiten avanzar.

Las aventuras conversacionales requieren de un procesamiento del lenguaje para su funcionamiento. Deben ser capaces de lidiar con todo tipo de fallas provocadas por el usuario y comunicarle al mismo qué ha salido mal. Así pues, el procesamiento del lenguaje natural es esencial. Y el resultado es sorprendentemente humano.

Sin embargo, la tecnología actual, las técnicas en procesamiento del lenguaje natural y las técnicas de inteligencia artificial han avanzado lo suficiente para poder ofrecer al jugador la posibilidad de emplear un lenguaje natural (casi) absoluto desplazando el uso de órdenes de juego.

1.2 Objetivos del trabajo

Este trabajo se ha desarrollado teniendo en cuenta el cumplimiento de los siguientes objetivos:

1. Conocer cómo funcionan las aventuras conversacionales y las plataformas/tecnologías existentes.
2. Diseñar un sistema de diálogo que flexibilice la comunicación entre el jugador y la plataforma.
3. Integrar el sistema de diálogo en una arquitectura para aventuras conversacionales popular.
4. Evaluar, mediante un grupo usuarios de prueba, la validez de la solución.

1.3 Antecedentes y estado del arte

Aunque el uso del PLN es indisociable de una aventura conversacional, este uso suele reducirse a un correcto *parseado* del texto que introduce el jugador a órdenes que entienda el juego. Debido a esto, la tolerancia al error que cometa el usuario al escribir es casi inexistente. Si el jugador desconoce los verbos que maneja el juego o no conoce su formato, el juego enviará un mensaje que indica que no entiende la entrada. Esto desemboca inevitablemente en una interfaz muy "primitiva" y restringe el grado de inmersión del usuario en el juego. En otras palabras, la interfaz no es trasparente al jugador. Asimismo, podría decirse que el juego no consigue ocultar su naturaleza al jugador, y cuando se juega se perciben las limitaciones del juego a la hora de interactuar con el mundo. A la hora de solventar este problema hay tres juegos distintos que destacar.

1.3.1 Event[0]

En este juego, un tripulante de una nave espacial que, tras la destrucción de su nave y lanzarse en una cápsula de salvamento llega a la *Nautilus*, una nave de recreo abandonada. Allí el jugador debe comunicarse con Kaizen-85, la IA de la nave, con el objetivo de poder volver a la Tierra. La conversación con la IA es uno de los pilares del juego, y la gran variedad de preguntas y respuestas a la hora de interactuar con la IA consiguen difuminar esa barrera entre el juego y el jugador.

1.3.2 AI Dungeon

Este juego es una aventura conversacional propiamente dicha, pero utiliza una IA (en concreto GPT-3 a la fecha de consulta) para generar sus historias. Tiene

varias temáticas predefinidas y permite a los jugadores crear y compartir sus historias. Funciona como una aventura conversacional tradicional, pero otorga una libertad desmedida al jugador a la hora de qué hacer. Sin embargo, al no tener sus historias una estructura estática como tal, sino que son generadas por IA, el resultado es impredecible y no hay un objetivo claro a la hora de jugar.

1.3.3 Façade

Façade es un drama interactivo en el que el jugador es invitado a cenar a casa de un matrimonio amigo, Grace y Trip. En cuanto llega al apartamento al jugador se le presenta una relación al borde del colapso. Debe interactuar en tiempo real (mediante la introducción de texto) con los integrantes de la pareja para tratar de solucionar sus problemas, habiendo múltiples finales como resultado. Este juego fue novedoso en su momento por su uso de la IA y el PLN para poder procesar la conversación del jugador y para construir la trama en tiempo real y reaccionando a las palabras del jugador. El resultado es interesante y, en mi opinión, el juego que mejor logra eliminar la frontera entre juego y jugador de los mencionados, pese a sus significativas carencias.

1.4 Tecnologías utilizadas

A continuación se describen las tecnologías empleadas en este trabajo.

1.4.1 Lenguaje de programación

Se ha optado por el lenguaje de programación Python debido a que ofrece gran capacidad a la hora de procesar textos, ofrece una gran cantidad de módulos, módulos y características nativas para cualquier tipo de tarea y, una característica fundamental, es uno de los lenguajes soportados por la API de OpenAI. Además, el hacedor tiene experiencia previa en su uso. Dentro del lenguaje, se han utilizado las siguientes módulos/bibliotecas adicionales:

1. **Subprocess:** este módulo permite lanzar distintos procesos desde Python. Más aún, permite lanzar estos procesos en segundo plano, permitiendo continuar con la ejecución del programa sin tener que esperar a una finalización del proceso invocado.

2. **API de OpenAI[1]:** nos premitirá interactuar con los servicios ofrecidos por OpenAI (en este caso GPT). Para ello se debe disponer de una API key que ha sido proporcionada por el tutor.

3. **json:** este módulo permite el correcto procesamiento de estructuras de datos con formato JSON, que es el que se utilizará para la comunicación con la máquina Glulxe.

1.4.2 OpenAI, GPT y el PLN

GPT es un acrónimo de *Generative Pre-trained Transformer*. *Generative* se refiere a que es un modelo que genera texto (en este caso). *Pre-trained* indica que el modelo ha sido previamente entrenado mediante grandes cantidades de datos. *Transformer* indica que es un tipo concreto de red neuronal. Explicado de forma muy elemental, estos modelos de lenguaje tratan de predecir, dado un texto, qué palabras vendrán a continuación, usando una distribución de probabilidad concreta cada vez que se añade una palabra. Para ello, el texto que sirve como *input* se divide en *tokens* que, en caso del texto, pueden ser palabras, sílabas, o una combinación de estas. A cada uno de estos *tokens* se le asocia un vector, que es una lista de números, que trata de codificar el significado de cada uno de los *tokens*. Estos vectores son sometidos a diversas operaciones para “refinar” sus valores en relación a los otros *tokens*. Esto es debido a que, por ejemplo, la palabra modelo puede variar su significado e importancia dependiendo de qué palabras la acompañan. Estas operaciones se realizan de forma sucesiva hasta llegar a un resultado final, que será la distribución de probabilidad para la siguiente palabra.La versión de GPT que se utiliza en este proyecto es GPT 3.5 turbo, desarrollado por la empresa OpenAI.

En este caso se utilizará GPT como herramienta para el procesamiento del lenguaje natural, abreviado PLN. El PLN es una disciplina dentro de la computación y la lingüística que trata de dotar a las computadoras la capacidad de procesar datos codificados como lenguaje natural, utilizando para ello una gran diversidad de técnicas.

[1] OpenAI, Documentation, API reference (11 de mayo de 2024). Obtenido de https://platform.openai.com/docs/api-reference

Al inicio del proyecto se comenzó usando GPT 3.5 turbo pero al final del mismo, especialmente en las pruebas finales, se decidió optar por la versión GPT 4 pues, aunque es algo más cara, los tiempos de respuesta son menores y, en opinión del hacedor, sufre menos *alucinaciones*[2].

1.4.3 Glulxe

La máquina Glulx[3] es una máquina virtual similar a la máquina Z[4]. Estas máquinas tienen una función similar a la máquina virtual de Java, esto es, poder escribir programas sin tener que atender a la compatibilidad entre distintos dispositivos hardware. La máquina Glulx permite datos y direcciones de 32 bits, y tiene soporte nativo para la entrada y salida mediante Glk[5], un estándar de interfaz para aventuras conversacionales. En nuestro caso se utilizará Glulxe (acrónimo de Glulx Execute), un intérprete de Glulx.

Para utilizar Glulxe es necesario que, a la hora de compilar el código de Glulxe, se haga junto a una biblioteca que implemente Glk. En este caso se ha optado por utilizar biblioteca RemGlk[6], pues se sirve del formato JSON para el envío y recepción de la comunicación, permitiendo transmitir información sobre el estado y otros parámetros que pueden resultar de interés. Además, RemGlk permite el uso de múltiples ventanas, lo que permite tener distintas sesiones de juego funcionando simultáneamente.

1.5 Descripción de la situación de partida

El punto de partida de este trabajo es “novedoso”. El hacedor no ha encontrado proyectos similares ni documentación al respecto. Hay que notar la existencia de dos TFGs previos[7,8], que hacían uso de la máquina Glulxe como vía

[2] Hace referencia a los casos en que GPT, al no conocer la respuesta que se le pide, la inventa.
[3] Glulx: A 32-Bit Virtual Machine for IF (11 de mayo de 2024). Obtenido de https://eblong.com/zarf/glulx/
[4] Wikipedia: Z-machine (11 de mayo de 2024). Obtenido de https://en.wikipedia.org/wiki/Z-machine
[5] Glk: An Interface Standard for Interactive Fiction (11 de mayo de 2024). Obtenido de https://eblong.com/zarf/glk/index.html
[6] RemGlk: remote-procedure-call implementation of the Glk IF API (11 de mayo de 2024). Obtenido de https://eblong.com/zarf/glk/remglk/docs.html
[7] García Quesada, Manuel Jesús (2017) Bot para Telegram que ejecute aventuras conversacionales. [Trabajo fin de grado, Universidad de Jaén]. https://hdl.handle.net/10953.1/5944

para jugar a aventuras conversacionales, lo cual da pistas sobre el funcionamiento de las tecnologías utilizadas, esto es, la máquina Glulxe y las distintas bibliotecas que existen para comunicarse con ella.

1.5.1 Descripción del entorno actual

A la hora de jugar aventuras conversacionales, la opción más usual suele ser instalar y usar una máquina Z o similar en el equipo (como, por ejemplo, Glulxe). Además, se necesitan juegos cuyo código sea compatible con la máquina ejecutora que se tenga. Otra opción es la existencia de implementaciones de estos juegos para ser jugados vía web, usualmente mediante una biblioteca de la máquina Glulxe que se comunica con un servidor. Esta última opción es similar a la implementada por los TFGs llevados a cabo por dos compañeros hace unos años, que permitía jugar a las aventuras conversacionales mediante un bot en la app Telegram[9].

Para jugar en una aventura conversacional, el usuario debe introducir órdenes en texto mediante el teclado para realizar las acciones deseadas. Estar órdenes suelen requerir de un verbo, que determina el tipo de acción, y un objeto al que el verbo hace referencia o que ajusta el significado del mismo. A la hora de desplazarse por el escenario de juego se deben usar los puntos cardinales, arriba y abajo y, ocasionalmente, dentro o fuera. Para interactuar con el mundo, se deben usar verbos que se adecúen a cada caso (con los personajes se puede hablar, las cosas se pueden coger, etc.).[10]

[8] Medina Íñiguez, Miguel Ramón (2018) Bot para Telegram que ejecute aventuras conversacionales. [Trabajo fin de grado, Universidad de Jaén]. https://hdl.handle.net/10953.1/8436

[9] Plataforma de mensajería y voz sobre IP. Enlaces: https://es.wikipedia.org/wiki/Telegram, https://telegram.org/faq#p-que-es-telegram-que-puedo-hacer-aqui

[10] https://pr-if.org/doc/play-if-card/play-if-card.html

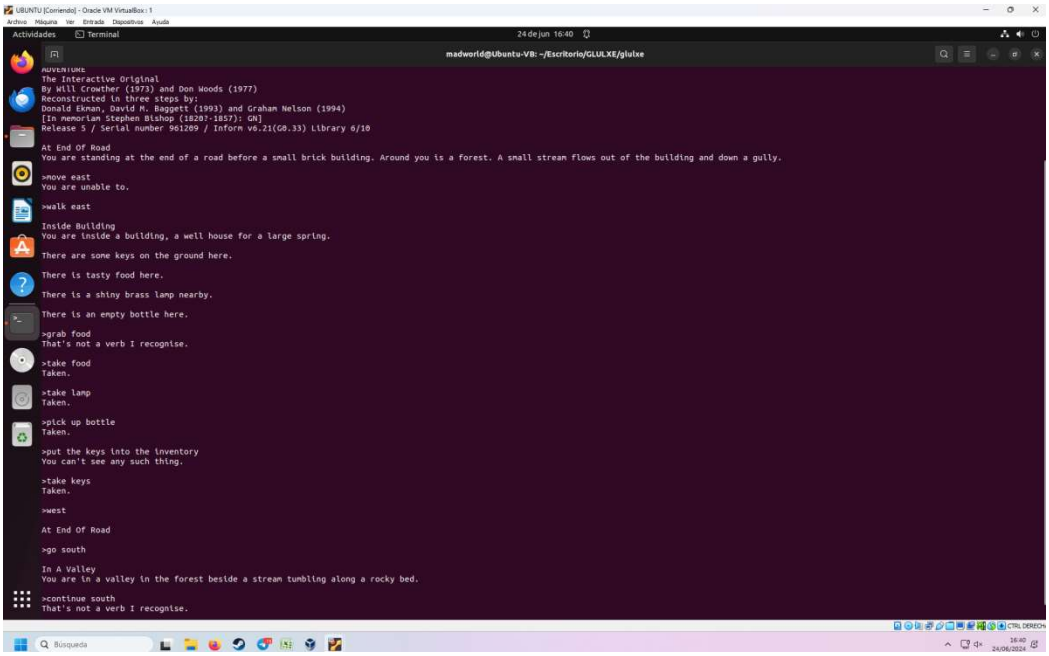

Ilustración 1.1

En la ilustración se puede observar un breve ejemplo. Muestra también que las aventuras conversacionales permiten cierto grado de libertad respecto a qué verbo usar aunque con restricciones.

Además, cada aventura conversacional suele incluir ciertas órdenes generales que permiten ciertas acciones como guardar, cargar partida, deshacer un movimiento o activar un modo *verbose* que no omite descripciones de escenarios ya visitados. Asimismo, existen *palabras mágicas* u órdenes propios de cada aventura conversacional que desempeñan una función especial.

1.5.2 Resumen de las deficiencias y carencias identificadas

Un problema que presentan las aventuras conversacionales es la necesidad de conocer las órdenes y su estructura a la hora de poder jugarlas. A no ser que se tenga una guía o similar, no es difícil llegar a momentos en los cuales no saber qué hacer o, incluso, atascarse y tener que buscar ayuda online. Asimismo, la necesidad de usar órdenes en lugar de poder utilizar el lenguaje natural limita la inmersión del jugador en juego.

1.6 Requisitos iniciales

A partir de la especificación del problema a resolver, se han establecido los siguientes requisitos generales:

1. El programa debe permitir jugar al usuario a cualquier aventura conversacional de las disponibles.
2. El programa debe traducir el lenguaje natural del usuario a órdenes de aventuras conversacionales.
3. El programa debe comunicar el resultado del proceso anterior al usuario.
4. El programa debe ser robusto frente a posibles errores, tanto de la máquina Glulxe como por parte de la entrada del usuario.

Como resultado del cumplimiento de estos requisitos se hará entrega de:

- Una memoria (esta) que contendrá información sobre el proceso de desarrollo, metodologías empleadas en el mismo, tecnologías utilizadas y un manual de usuario para su uso e instalación.
- El código del software desarrollado que se almacenará en un repositorio de GitHub.

1.7 Alcance

El trabajo consiste en la implementación de un programa que se sitúe entre el usuario y la máquina Glulxe y que, en conjunto con la capacidad que ofrece GPT en el procesamiento del lenguaje natural, permita al usuario utilizar este último para jugar a aventuras conversacionales, así como poder utilizar otras funciones que ofrece la máquina Glulxe.

Por la propia naturaleza del trabajo, queda fuera del alcance el mantenimiento posterior del software generado, posibles modificaciones y/o mejoras del software o la elaboración de un catálogo elaborado de juegos.

1.8 Hipótesis y restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo

de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

Una limitación menor es el repertorio de disponible de aventuras conversacionales. Muchas de ellas se encuentran en formatos compatibles con la máquina Z original y su uso implicaría compilar dichos códigos al formato de código de la máquina Glulxe, tarea que queda fuera del ámbito de este trabajo. Además, no se han encontrado aventuras conversacionales en idiomas distintos al inglés.

1.9 Estudio de alternativas y viabilidad

Una alternativa a la hora de usar GPT como herramienta para el procesamiento del lenguaje natural sería implementar directamente el software que llevase a cabo esa tarea. Se decidió descartar esta opción por lo complejo que sería, el limitado tiempo disponible y la falta de formación del hacedor en el campo del procesamiento del lenguaje natural.

Por otro lado, se valoró utilizar el de lenguaje de programación javascript, pues era otro lenguaje para el cual, tanto la máquina Glulxe como la API de GPT, ofrecían soporte. Dado que no se tenía intención de alojar el software como un servicio web, el hacedor no tenía demasiada experiencia con este lenguaje y, más adelante, la necesidad de trabajar con subprocesos del sistema, llevaron a continuar con el lenguaje de programación Python.

Respecto a la máquina Glulx, esta requiere, a la hora de ser compilada, de una biblioteca Glk que permita la comunicación con la máquina. Una opción considerada fue la biblioteca CheapGlk[11]. Esta es una implementación simple y minimalista de Glk. Si bien emplea las entradas y salidas estándar y su formato es muy sencillo, se descartó en favor de la biblioteca RemGlk por ser esta última más completa en cuanto a la información transmitida y a que permite múltiples sesiones de juego simultáneas, algo que CheapGlk no permite.

Se planteó la posibilidad de que el programa utilizase programación concurrente para cada una de sus funciones. Sin embargo, debido a que es perfectamente funcional en su versión monohilo, se descartó esta opción pues o

[11] Glk: An Interface Standard for Interactive Fiction (11 de mayo de 2024). Obtenido de https://eblong.com/zarf/glk/index.html

aporta una mejora en lo que al rendimiento e implicaría un aumento considerable en la lógica del programa.

1.10 Descripción de la solución propuesta

La solución consiste en un programa que se situará entre el usuario, la máquina Glulxe y GPT que permitirá al usuario jugar a aventuras conversacionales mediante el lenguaje natural.

1. El programa tomará la entrada por teclado del usuario. Esta será una orden para la máquina Glulxe escrita en lenguaje natural.
2. Enviará la entrada a GPT con el prompt adecuado para que este traduzca el lenguaje natural de la petición del usuario en órdenes de juego que sean comprensibles por la aventura conversacional.
3. Construirá un mensaje que pueda ser comprendido y decodificado por la máquina Glulxe.
4. Una vez enviado y recibida la respuesta de la máquina, procesará ese mensaje y se lo mostrará al usuario.

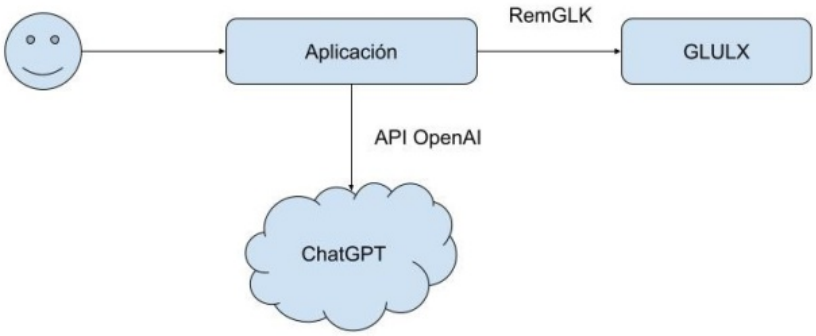

Ilustración 1.2

Respecto a la estructura, se ha optado por una programación monohilo, funcional y sin clases. La “sencillez” del problema permite desechar una orientación a objetos en su estructura, si bien se hace uso de clases ya existentes del lenguaje de programación y de las herramientas empleadas.

1.11 Metodología de desarrollo de software

Respecto a la metodología utilizada se ha optado por una metodología clásica incremental. Esta metodología es una modificación de la metodología en cascada (o metodología de la cascada), a veces llamada ciclo de vida clásico, la cual implica un enfoque sistemático y secuencial. Se cuenta además con una estructura de etapas bien definida.

Ilustración 1.3

A diferencia de la metodología en cascada, en la metodología incremental se entregarán sucesivas versiones inacabadas del software añadiendo nuevas funcionalidades y características en cada entrega. En cada una de estas entregas se pueden señalar modificaciones, correcciones u optar por un enfoque distinto.

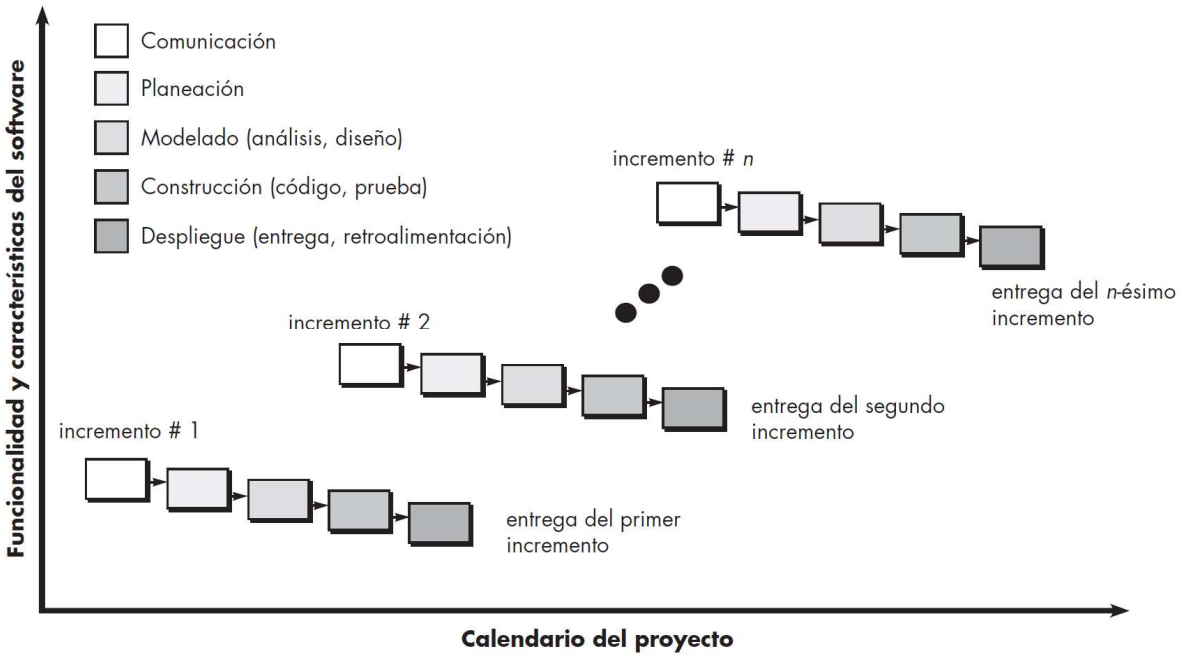

Ilustración 1.4

Se ha optado por esta metodología por varias razones:

- El ser un proyecto desarrollado por un único individuo.
- El tener un tamaño y alcances bien delimitados.
- El tener unos requisitos bien definidos y sin previsión de modificación.
- La naturaleza del problema a resolver permite descomponer el problema en un conjunto de subproblemas casi independientes entre sí, lo que comulga razonablemente bien con una metodología incremental.

1.12 Análisis de riesgos

La mayor parte de los riesgos asociados al proyecto están relacionados con el uso de OpenAI y GPT. Una modificación de las políticas de uso o monetización podrían implicar que su uso fuese imposible. Asimismo, un cambio de versión en GPT podría provocar la necesidad de ajustar y/o reescribir parte del código del proyecto. Además, existía la posibilidad de que GPT no fuese capaz de entender la función que se le pide y, por tanto, incapaz de proporcionar las órdenes buscadas.

Por otra parte, la comunicación con la máquina Glulxe está supeditada a qué biblioteca Glk se utilice. Era necesario disponer de una biblioteca que emplease la entrada y salida estándar. En primera instancia se ha partido de RemGlk que emplea un formato JSON para la trasmisión de información. En caso de que la trasmisión de información a la máquina mediante RemGlk tuviese deficiencias insalvables, se disponía de otra biblioteca, CheapGlk, que no emplea JSON (sigue empleando la entrada y salida estándar).

1.13 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, existen restricciones de tipo económico y de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. Debido a las restricciones económicas se ha optado por soluciones de código abierto y gratuitas o mediante licencias proporcionadas por la universidad (caso de *Visual Paradigm*). La única excepción es la API de OpenAI dado que su tarifa es asumible a día de la realización de este TFG. En cuanto al esfuerzo, se dispone de tan un solo efectivo (la persona autora del trabajo).

Se ha utilizado **COCOMO básico** para estimar el esfuerzo del desarrollo del software. Se ha optado por un **tipo orgánico** pues es un proyecto pequeño a nivel de líneas de código y no hay gran innovación técnica. Las ecuaciones necesarias serán:

$$E = a \cdot KLDC^{b}$$

$$TD = c \cdot E^{d}$$

Siendo E la estimación del esfuerzo medida en *hombres-mes*, KLDC el número de líneas de código (expresado en miles de líneas), TD el tiempo de desarrollo medido en meses y a, b, c, d coeficientes del método.

Respecto a los cálculos, se ha estimado que el tamaño aproximado del proyecto será de unas 400 líneas de código. Con esto tenemos que:

$E = 2.4 \cdot 0.4^{1.05} = 0.9170$ hombres-mes ≈ 1 hombre-mes

$TD = 2.5 \cdot 1^{0.38} = 2.5$ meses

Como resultado se obtiene que el tiempo de desarrollo necesario será de 2.5 meses además de la realización de la memoria y del proceso de investigación previo al desarrollo del software.

1.14 Planificación temporal

Para la planificación del trabajo se ha decidido descomponer el trabajo a realizar en distintas tareas.

Tarea	Tiempo estimado
Investigación	2 semanas
Comunicación del programa con Glulxe	2 semanas
Comunicación del programa con GPT	1 semanas
Combinar ambas secciones de comunicación	2 semanas
Primer borrador de la memoria	1 semana
Refinado del procesado de la salida de Glulxe, refinado del prompt y refinado de la entrada del usuario.	2 semanas
Interfaz de usuario para guardar, cargar, ayuda y salir.	1 semana
Implementación de la revisión de la traducción por parte de GPT.	1 semana
Finalización de la memoria.	1 semana

Tabla 1.1

Esta estimación da un total de 12 semanas de trabajo, cada una con 5 días laborales resultan en 60 días laborales completos, lo que equivale a 3 meses de trabajo a media jornada.

1.15 Presupuesto

Para la estimación del coste se han tenido en cuenta costes directos, esto es, equipos, licencias de programas, subscripciones a servicios, etc., así como costes indirectos. Se tienen los siguientes gastos:

- Equipo informático: Ordenador portátil MSI GS63 Stealth 8RD-062XES[12]. Es el ordenador del hacedor pero se podría usar cualquier equipo que cumpliese con los requisitos de los programas utilizados.
- IDE: PyCharm[13]. Su versión *Community* es gratuita. Su versión *Professional* es de pago. Pueden utilizarse otros IDEs o editores de texto teniendo en cuenta que variará la configuración descrita más adelante. En este caso consideraremos que el IDE no presenta un coste.
- OpenAI y GPT: Las tasas y precios estipulados se calculan en función de los tokens empleados[14].
- Salario: Como resultado de la estimación del esfuerzo y la planificación temporal se obtiene un volumen de trabajo de 5 horas diarias durante 60 días laborables, lo que equivaldría a 3 meses de trabajo.

Con estos supuestos se tiene:

Concepto	Importe unitario	Amortización	Total / mes	Total
Ordenador	999,00[15] €	5 años	16,65 €	49,95 €
Visual Paradigm Standard	349,00 €	3 años	9,69 €	29,07 €
	Importe fraccionario			

[12] Enlace: https://www.pccomponentes.com/msi-gs63-stealth-8rd-062xes-intel-core-i7-8750h-16gb-512ssd-gtx1050ti-156
[13] Enlace: https://www.jetbrains.com/pycharm/download/
[14] Enlace: https://openai.com/api/pricing/
[15] Aunque el precio referenciado anteriormente es el precio actual del producto, debido a que se encuentra descatalogado se ha optado por usar el precio por el que se compró en su momento.

Salario	1202,09 € / mes[16]	3.606,27 €
OpenAI API (GPT)	0,50 € / 1 Millón de tokens[17]	0,50 €
Total (Suma de todos los totales)		3.685,79 €

Tabla 1.2

En base a los cálculos realizados se estima un coste total de 3.685,79 €.

1.16 Normas y referencias

A continuación, se presentan las normas, reglamentos y referencias de cualquier tipo que han sido de aplicación en la elaboración del trabajo y/o en la puesta en práctica del mismo.

- Se han utilizado las normas APA para la bibliografía y referencias.

1.16.1 Mecanismos de control de calidad

El aseguramiento de la calidad en la redacción del trabajo implica la verificación de la completitud (falta de omisiones), la integridad de la documentación, así como una redacción clara, concisa y entendible por todos los participantes e interesados en dicho trabajo.

Al estar el trabajo desarrollado por una sola persona y verificado por el tutor, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, se han utilizado mecanismos básicos para la verificación de la integridad y completitud, que incluyen un control de versiones a nivel de documento y un control sencillo de la trazabilidad de especificaciones y requisitos.

[16] Según el *Boletín Oficial del Estado. Convenio colectivo estatal de empresas de consultoría, tecnologías de la información y estudios de mercado y de la opinión pública*. El sueldo de un programador es 28.850,23 € al año por lo que en este caso se traduce en 1.202,09 € al mes en un trabajo a media jornada.

[17] Depende de la versión de GPT empleada. Se ha supuesto que un millón de tokens son suficientes para este proyecto y se ha optado por usar el coste de GPT 3.5-turbo pues es el que se ha usado durante la mayor parte del proyecto.

2 DISEÑO INICIAL

El diseño inicial consta de una pieza de software que debe situarse entre el usuario y la máquina Glulxe. Este software debe tomar la entrada del usuario, convertirla de lenguaje natural a órdenes de la aventura conversacional y trasladar dicha órdenes a la máquina Glulxe. Por último, mostrar la respuesta de la máquina al usuario.

2.1 Especificaciones del sistema

A continuación, se hace una exposición detallada de los requisitos del programa.

2.1.1 Requisitos funcionales

1. **RF1:** El programa debe ser capaz de traducir el lenguaje natural del usuario a órdenes del juego.
2. **RF2:** El programa debe ser capaz de gestionar los casos en los que la aventura conversacional no comprenda la orden proporcionada por el mismo programa.
3. **RF3:** En caso de error de la máquina Glulxe el programa debe ser capaz de informar correctamente al usuario y finalizar correctamente.
4. **RF4:** El programa debe ser capaz de reconocer y procesar correctamente ciertas órdenes del usuario que no requieran un procesamiento pues no son lenguaje natural. Ejemplo: rutinas para salir de la partida, guardar la partida, etc.

2.1.2 Requisitos no funcionales

1. **RNF1:** A la hora de traducir el lenguaje natural del usuario a órdenes de juego, el programa debe proporcionar una respuesta en un tiempo aceptable, especialmente a la hora de gestionar los casos en que la orden proporcionado por el programa no sea entendido por el juego. Se propone un máximo de tres iteraciones a la hora de reformular la traducción antes

de abandonar y comunicar al jugador que no se le ha entendido y que debe probar de nuevo.

2. **RNF2:** El programa debe proveer un formato de salida del texto del juego que sea comprensible para el usuario.

2.2 Análisis y diseño del sistema

Como se ha mencionado anteriormente, la arquitectura general el sistema consta de tres partes:

- La máquina Glulxe que es el software encargado de ejecutar los juegos y es con quien se comunicará el programa principal.
- GPT que será el encargado de traducir el lenguaje natural del usuario a órdenes de juego. Para la comunicación con este se empleará la API de OpenAI
- El programa principal que será quien comunique al usuario con la máquina utilizando la traducción de GPT.

Respecto a la estructura interna del programa, la secuencia de ejecución es la siguiente:

1. En primera instancia, se presenta al usuario el menú para elegir qué juego desea jugar.

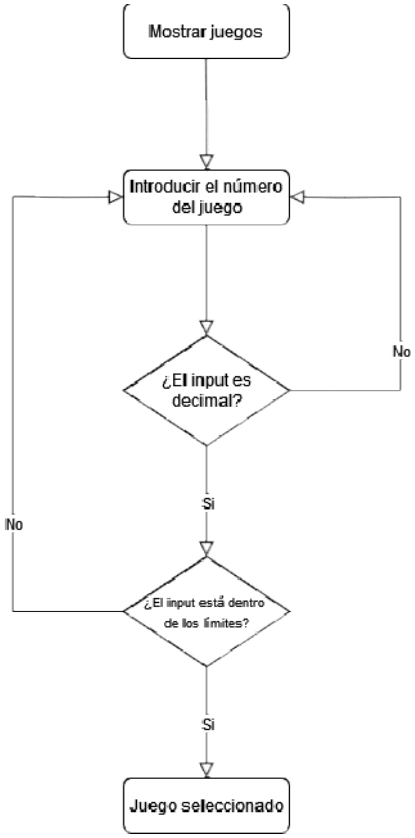

Ilustración 2.1

2. Una vez elegido el juego, se cargan en el programa la lista con los archivos de guardado existentes, la lista de palabras reservadas y la lista con las frases que responderá la máquina Glulxe en caso de no entender la traducción de GPT. Asimismo, se conforma el *prompt* con la descripción de la tarea que debe hacer GPT, así como con la lista de verbos más comunes.
3. Tras esto comienza el bucle principal del programa.
   3.1. En primera instancia, se recoge y se procesa la salida de la máquina Glulxe.
   3.2. Lo siguiente es pedir la orden al usuario. Esta se comparará con la lista de palabras reservadas para decidir si debe ser procesada o no por GPT.
   3.3. El resultado de la decisión se proporciona a la máquina Glulxe, volviendo así al comienzo del bucle. Hay que señalar que la repetición de la traducción de GPT sería parte de este bucle principal excepto que, en este caso, no se pide la orden al usuario mientras se repite dicha traducción. Asimismo, existen las rutinas para guardar/cargar partida y para salir del juego. Estas rutinas se invocan desde el bucle principal.

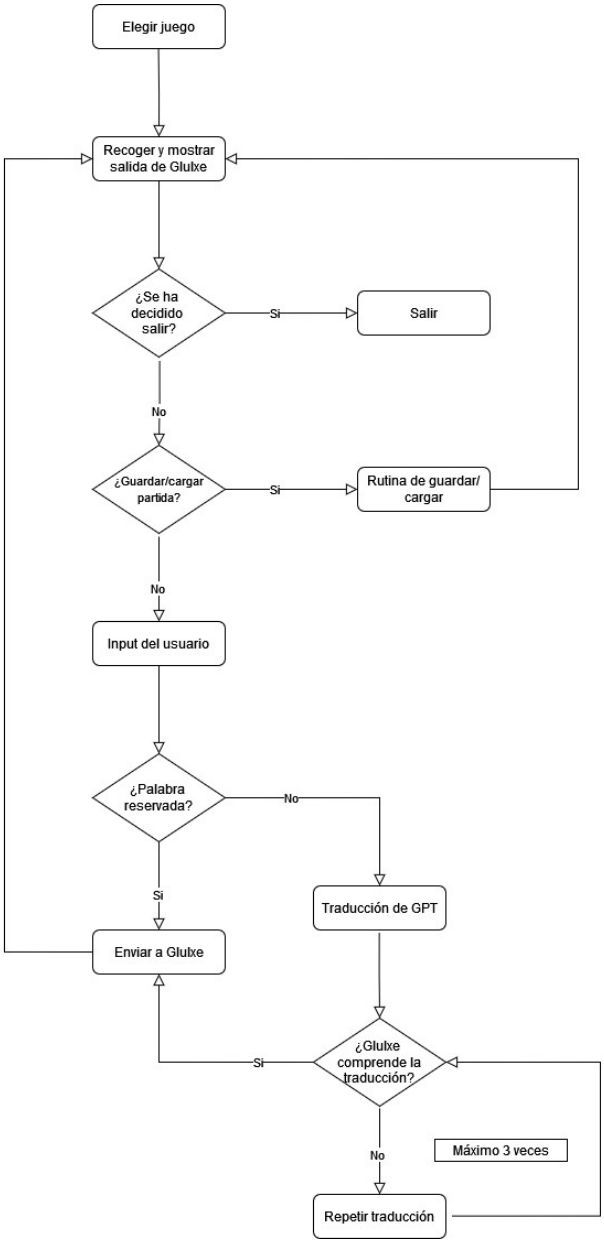

Ilustración 2.2

4. Para cargar partida, se presenta al usuario un menú con cada uno de los archivos de guardado disponibles. El usuario debe seleccionar cual de ellos desea cargar.

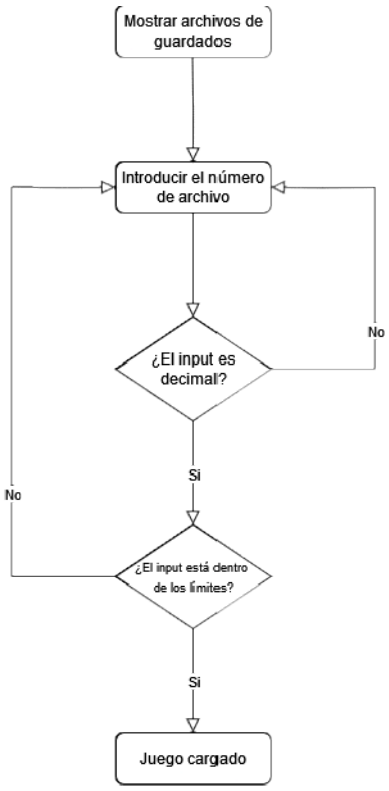

Ilustración 2.3

5. Para guardar partida, el usuario debe escribir el nombre del archivo de guardado con un formato adecuado. En caso de que el archivo ya exista, debe elegir si debe sobreescribir o no dicho archivo.

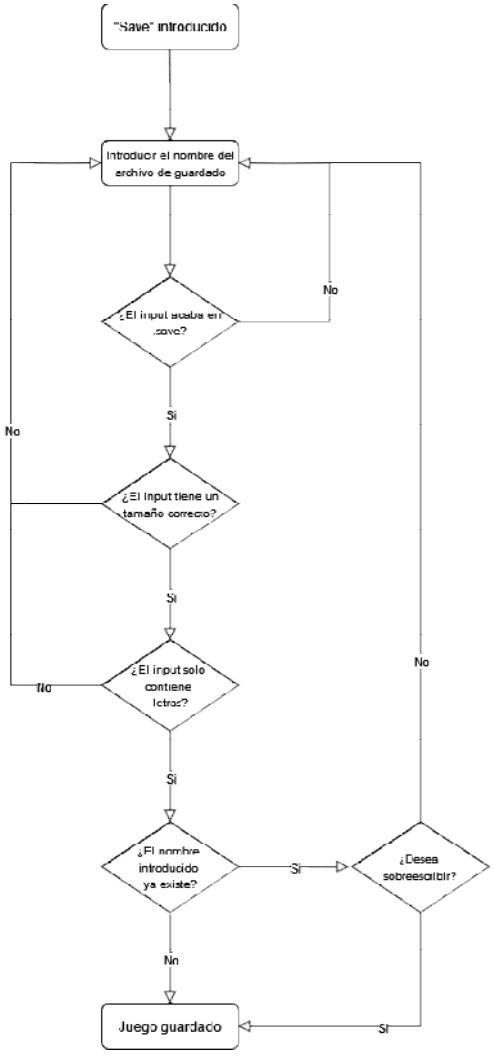

Ilustración 2.4

6. Para salir, el usuario simplemente debe escribir la orden para ello y confirmar.

2.2.1 Casos de uso

En virtud de lo descrito en el apartado anterior se han establecido los siguientes casos de uso:

ID	Actor	Caso de uso
1	Usuario	Seleccionar un juego
2	Usuario	Jugar al juego
3	Usuario	Guardar partida
4	Usuario	Cargar partida
5	Usuario	Ayuda
6	Usuario	Salir

Tabla 2.1

2.2.1.1 Diagrama de los casos de uso:

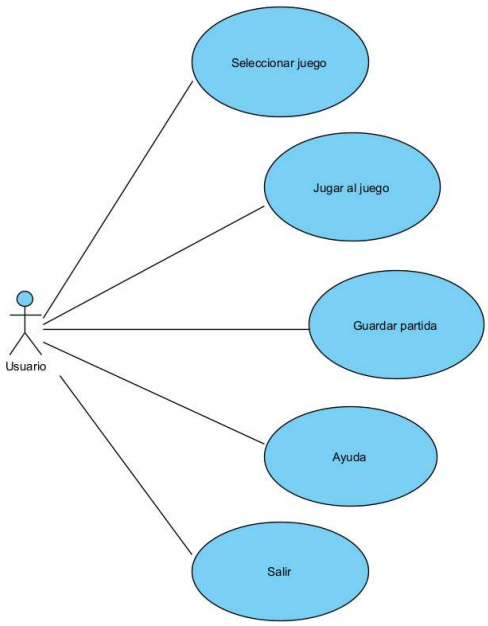

Ilustración 2.5

2.2.1.2 Especificaciones de los casos de uso

Caso de uso	Seleccionar un juego
ID	1
Actor primario	Usuario
Condiciones previas	El usuario ha lanzado el programa
Operaciones básicas	

Escenario principal	1. Elegir uno de los juegos ofrecidos 2. Indicar al programa cual se ha elegido

Tabla 2.2

Caso de uso	Jugar a un juego
ID	2
Actor primario	Usuario
Condiciones previas	El usuario ha seleccionado uno de los juegos
Operaciones básicas	
Escenario principal	1. Leer la información del escenario proporcionada por el juego 2. Escribir la acción que se desea hacer 3. Repetir desde el paso 1
Escenarios alternativos	2a, La máquina no ha entendido la traducción. 1. Reformular la escritura de la acción

Tabla 2.3

Caso de uso	Guardar partida
ID	3
Actor primario	Usuario
Condiciones previas	El usuario está jugando a uno de los juegos
Operaciones básicas	
Escenario principal	1. Escribir la orden de guardar partida 2. Escribir el nombre del archivo de guardado 3. Confirmar
Escenarios alternativos	2a, El nombre de archivo ya existe 1. Sobrescribir el archivo. 2. Cambiar el nombre elegido del archivo

Tabla 2.4

Caso de uso	Ayuda
ID	4
Actor primario	Usuario
Condiciones previas	El usuario está jugando a uno de los juegos
Operaciones básicas	
Escenario principal	1. Escribir la orden de ayuda. 2. Navegar el menú de ayuda 3. Seleccionar la sección de ayuda a consultar 4. Consultar la ayuda

Tabla 2.5

Caso de uso	Salir
ID	5
Actor primario	Usuario
Condiciones previas	El usuario está jugando a uno de los juegos
Operaciones básicas	
Escenario principal	1. Escribir la orden de salir 2. Confirmar que se quiere salir
Escenarios alternativos	2a, Se ha cambiado de opinión 1. Rechazar que se quiere salir

Tabla 2.6

Caso de uso	Cargar partida
ID	6
Actor primario	Usuario
Condiciones previas	El usuario está jugando a uno de los juegos
Operaciones básicas	
Escenario principal	1. Escribir la orden de guardar partida 2. Escribir el nombre del archivo de guardado 3. Confirmar
Escenarios alternativos	2a, El nombre de archivo ya existe

	1. Sobrescribir el archivo. 2. Cambiar el nombre elegido del archivo

Tabla 2.7

2.2.2 Diagramas de secuencia

A continuación se proporcionan los diagramas de secuencia para cada caso de uso:

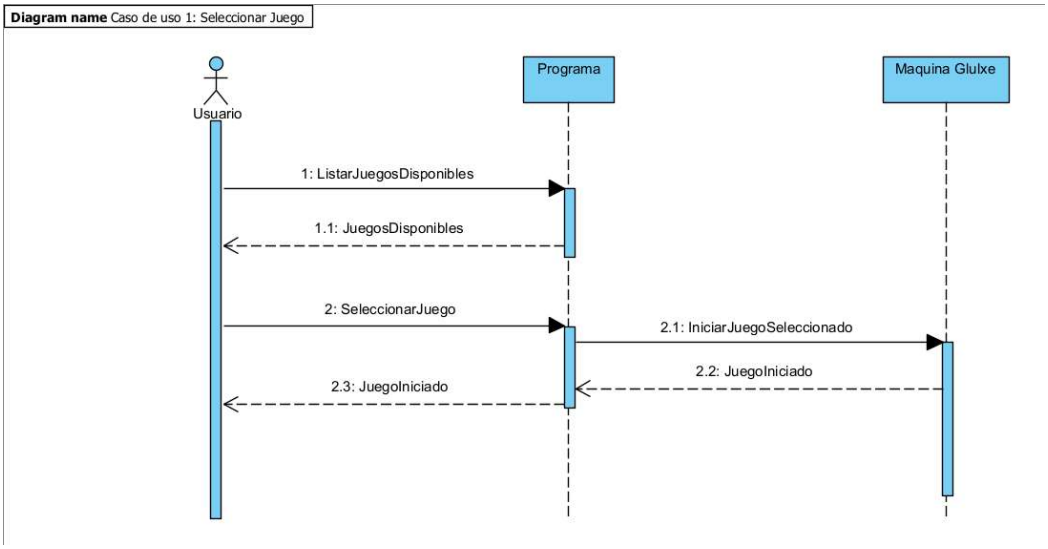

Ilustración 2.6

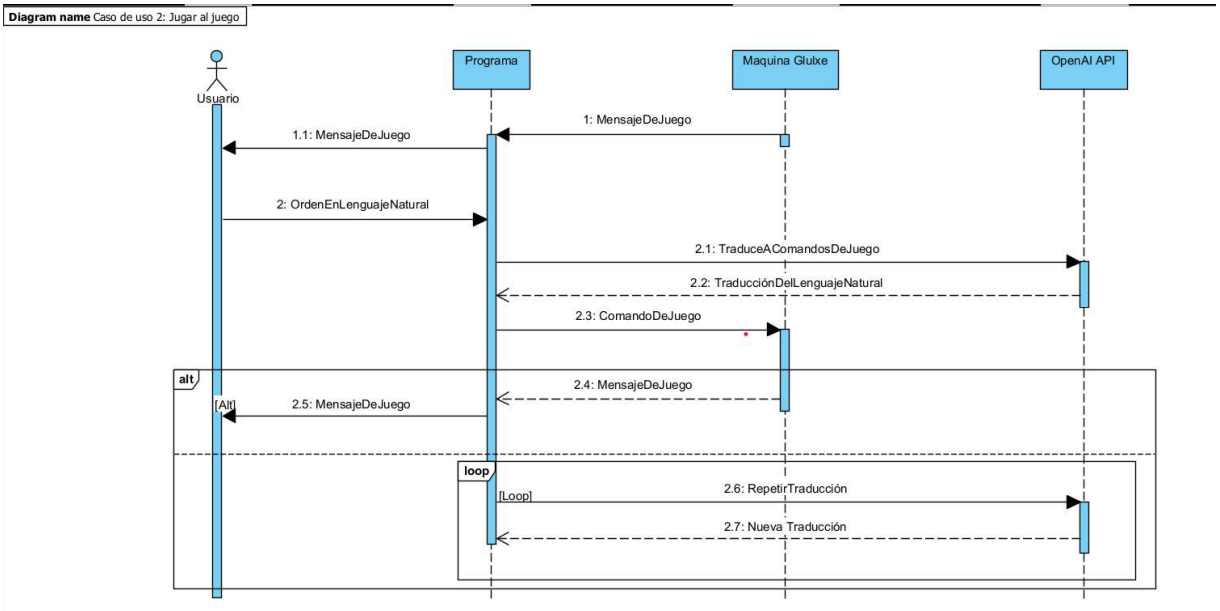

Ilustración 2.7

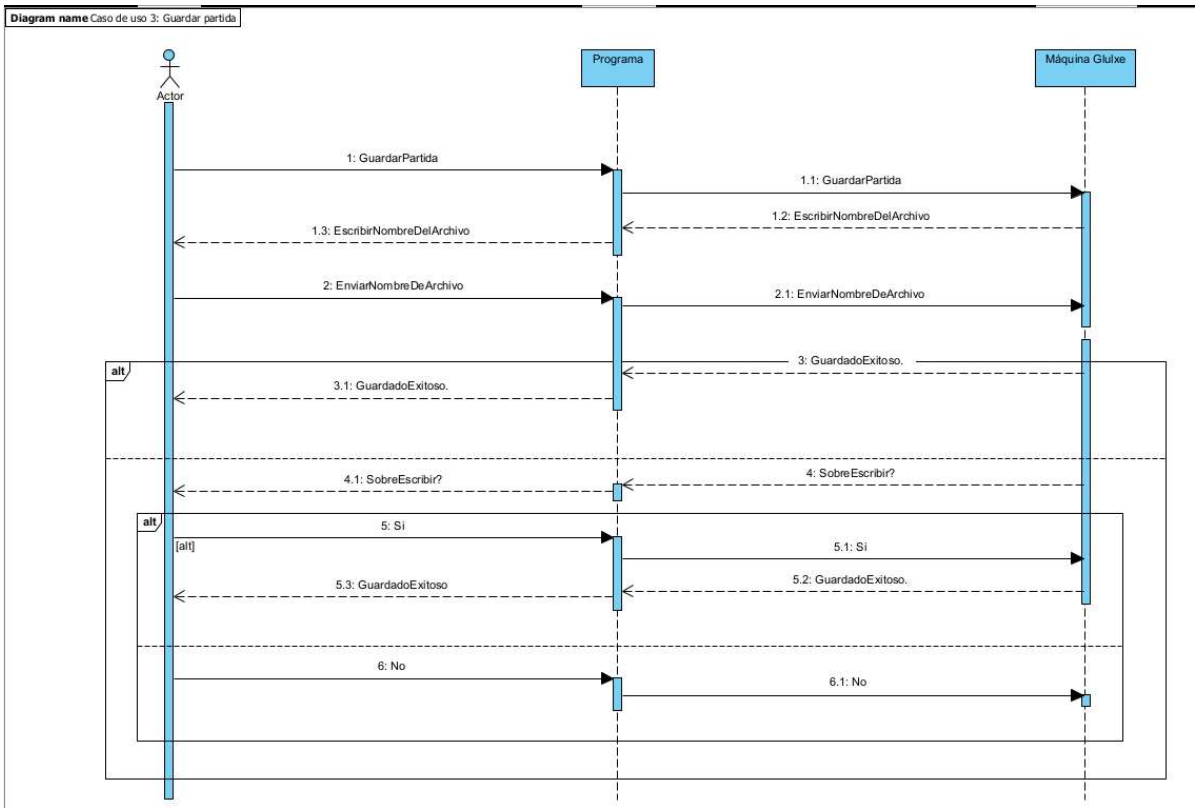

Ilustración 2.8

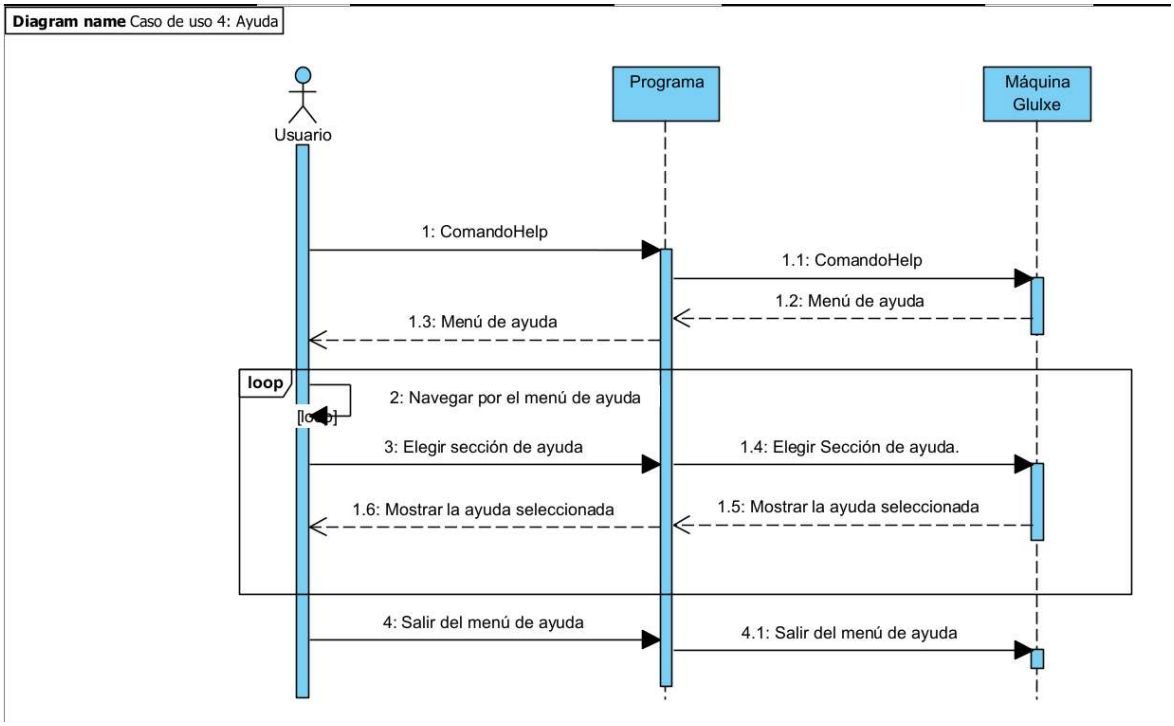

Ilustración 2.9

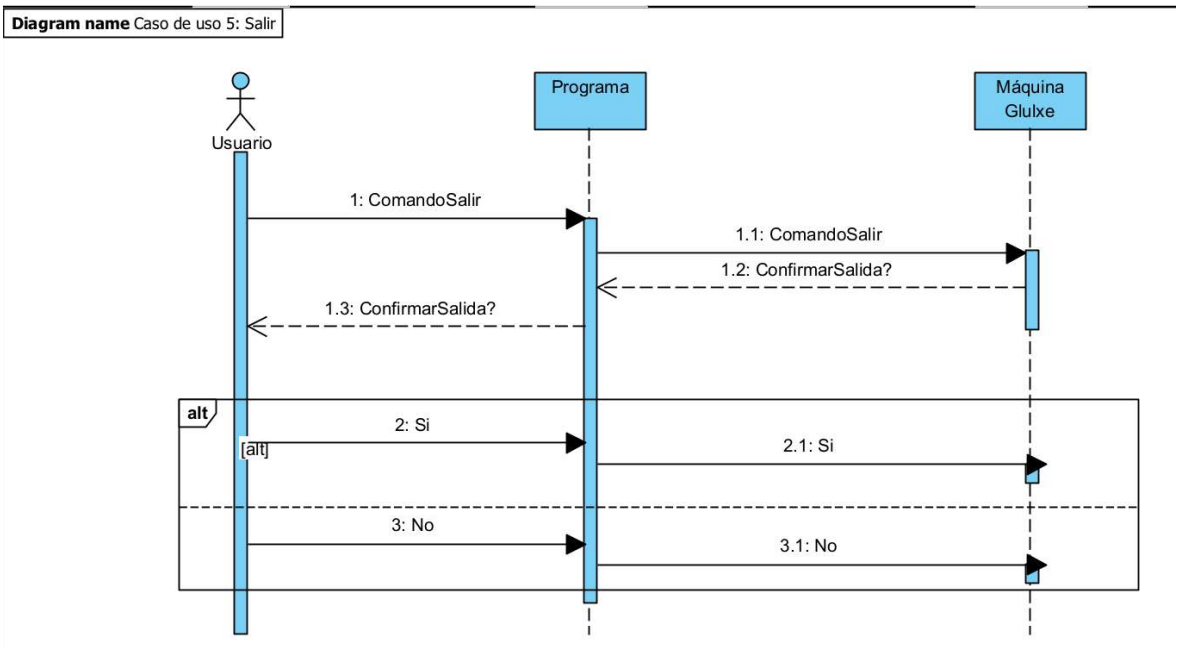

Ilustración 2.10

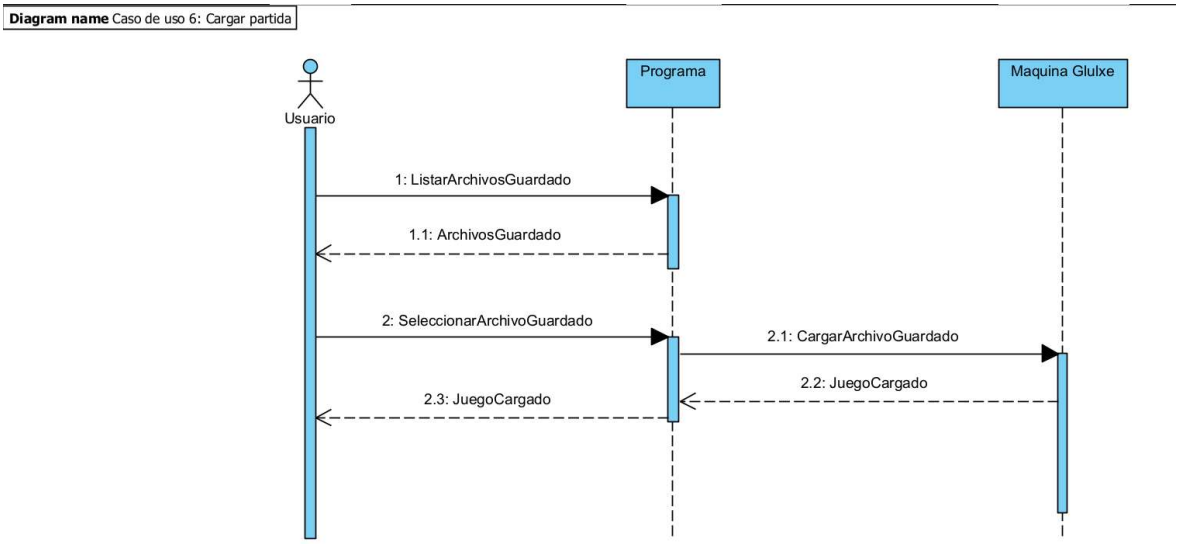

Ilustración 2.11

3 DESARROLLO

3.1 Investigación

El primer paso fue la investigación sobre las aventuras conversacionales y la máquina Glulxe pues no solo el hacedor no estaba nada familiarizado con ellas, sino que era de vital importancia tener familiaridad con las órdenes y estructura de juego de las aventuras conversacionales. Este proceso requirió:

- Jugar a varias aventuras conversacionales distintas con el objetivo de familiarizarse con ellas.
- Comprobar las diferentes estructuras que presentan las órdenes y realizar una clasificación de los mismos en base a esta estructura.
- Conocer qué órdenes son comunes en la mayoría de aventuras conversacionales.

Tras este proceso y ya definida la estructura del problema, hubo que investigar qué tecnologías ofrecidas por el lenguaje de investigación permitirían desarrollar el software del proyecto, así como qué posibilidades ofrecía la API de OpenAI y qué capacidades proporcionaba GPT para resolver el problema, además del funcionamiento de ambas.

3.2 Primera Iteración

Durante primera iteración se resolvió el problema de comunicación entre la máquina Glulxe y el programa. Para ello se investigó sobre cómo lanzar un proceso mediante un programa en Python de forma que dicho programa no tuviese que esperar a que el proceso lanzado finalizase para procesar el resultado (es decir, que esa invocación fuese no bloqueante). La opción recomendada por la propia documentación oficial de Python fue el módulo Subprocess.

Una vez que solucionado la cuestión de lanzar la máquina a través de Subprocess se trabajó en la comunicación del programa con la máquina. Esta es una de las razones por las que se usó una biblioteca Glk que utilizase la entrada y salida estándar, pues el propio módulo Subprocess permite capturar ambas. Sin

embargo, hay que ser cuidadoso al manejar la entrada y salida estándar pues los errores no son sencillos de depurar y tienden a bloquear el programa.

3.3 Segunda Iteración

En esta iteración se resolvió el problema de la interacción con GPT a través de la API de OpenAI. Esta parte fue bastante sencilla y no causó excesivo problema, al menos en su forma básica. Hay que notar que se descubrió que, al menos a través de la API, GPT no guarda memoria de las interacciones pasadas, lo que implica que se tuvo que definir un *prompt* inicial que se enviaría todas las veces que se interactuase con el Chat, de forma que se comportase de la forma deseada. Hay que añadir, sin embargo, que existe la posibilidad de enviar mensajes al Chat de forma que contengan, aparte del ya mencionado *prompt*, un historial de la conversación, de forma que se pueda pedir al Chat que actúe según el contexto, cosa que será útil cuando se pida al Chat que reformule su traducción del lenguaje natural a órdenes de la máquina Glulxe.

3.4 Tercera Iteración

Tras las dos iteraciones anteriores se llevó a cabo la combinación de ambas en un mismo programa, dando lugar a la primera versión funcional del mismo. En este punto se pudo empezar a pulir cuestiones como el procesado de la respuesta de la máquina Glulxe para su presentación al usuario. Asímismo, se decidió que ciertas palabras no fuesen procesadas por el Chat pues son rutinas que no forman parte del proceso de juego en sí. Por consiguiente, tales acciones como guardar partida, salir del juego, entrar en el menú de ayuda y las correspondientes subrutinas asociadas no son procesadas por el Chat y son directamente tratadas por la máquina Glulxe de la misma forma que si se interactuase con ella directamente.

3.5 Cuarta Iteración

En este momento del desarrollo se comenzó a trabajar en la memoria del proyecto. Además, se implementaron las rutinas que permitían evitar el procesamiento de ciertas ordenes por parte de GPT y su correcto tratamiento, tanto para la captación de la salida de la máquina Glulxe como para la introducción de los

mensajes para la misma, pues hay diferencias notables en las estructura del JSON de estos mensajes, tanto para la entrada como para la salida.

3.6 Quinta Iteración

Durante esta iteración se añadió la interfaz para el usuario, con menús para elegir juegos y guardar/cargar partidas. Además, se reorganizó la estructura de los directorios y archivos necesarios para el funcionamiento del programa.

3.7 Sexta Iteración

Tras la programación de la interfaz de usuario, se refinó el *prompt* proporcionado a GPT para una respuesta más precisa y con un formato más adecuado. Se añadió la función para repetir la consulta a GPT en caso de que la máquina Glulxe no comprendiese la respuesta. Asimismo, se finalizó la memoria del proyecto y se realizaron pruebas con los usuarios.

3.8 Pruebas finales

Las pruebas definitivas realizadas con el sistema han resultado satisfactorias. El sistema es capaz de procesar una gran variedad de situaciones a órdenes conocidas por la máquina Glulxe, proporcionando un mayor grado de flexibilidad a la hora de jugar a de aventuras conversacionales. El sistema también permite guardar y cargar partidas, y permite utilizar órdenes especiales como *verbose* o *undo*.

A continuación se muestran una serie de capturas con un ejemplo ilustrando el funcionamiento final del sistema.

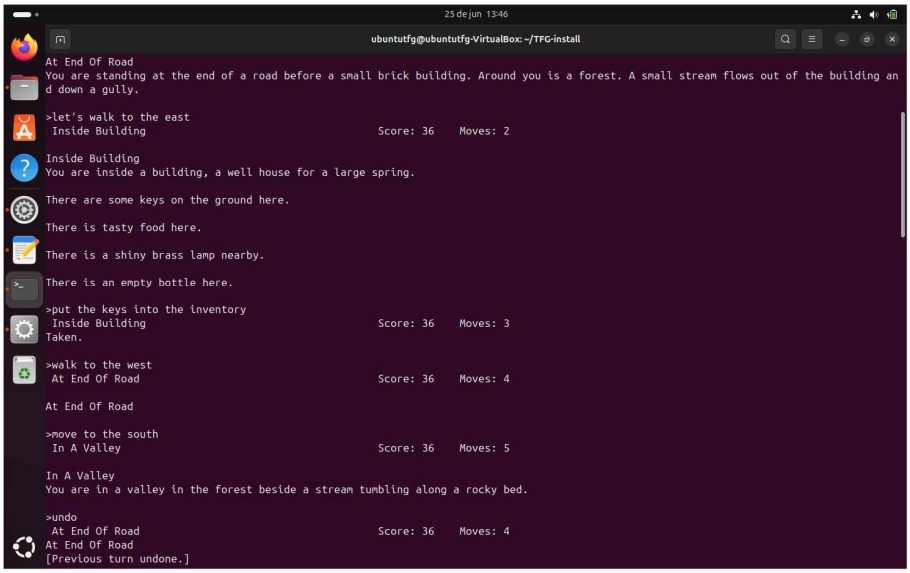

Ilustración 3.1

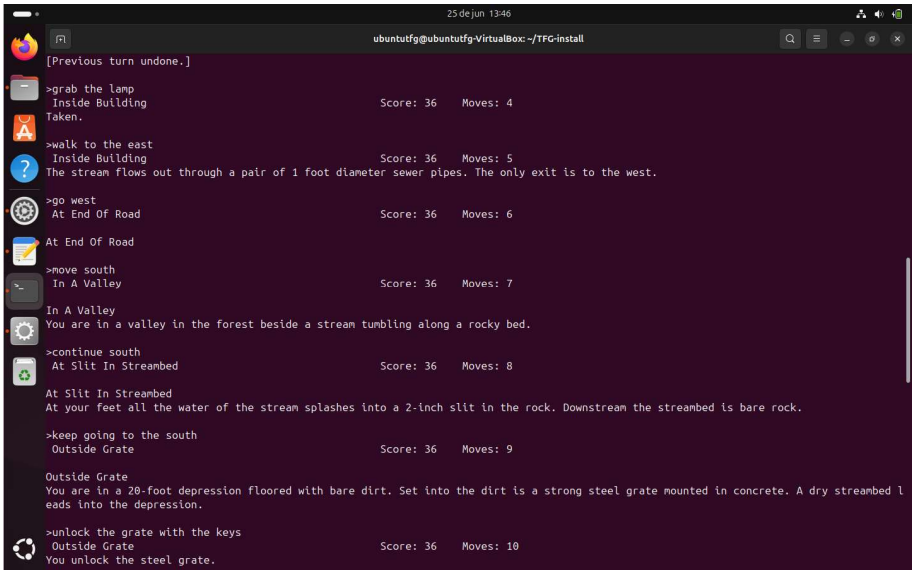

Ilustración 3.2

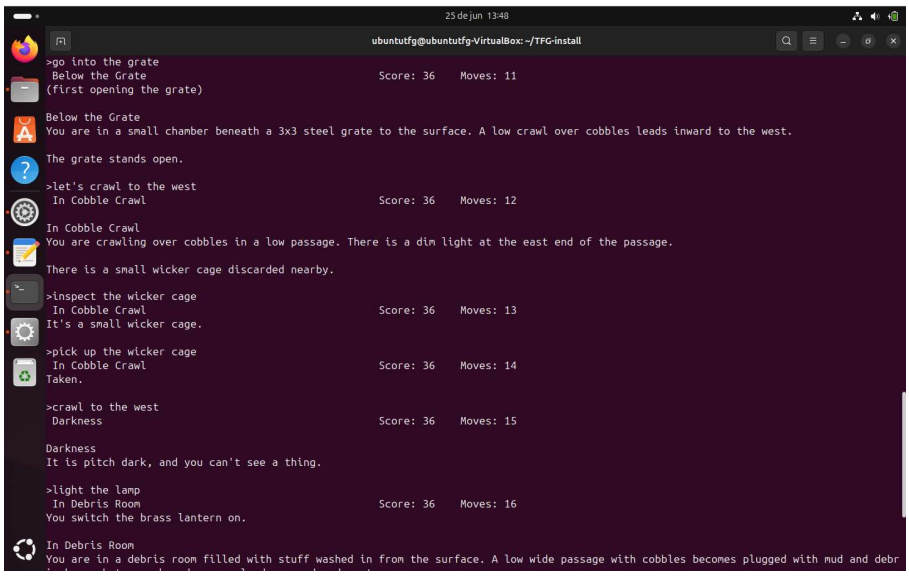

Ilustración 3.3

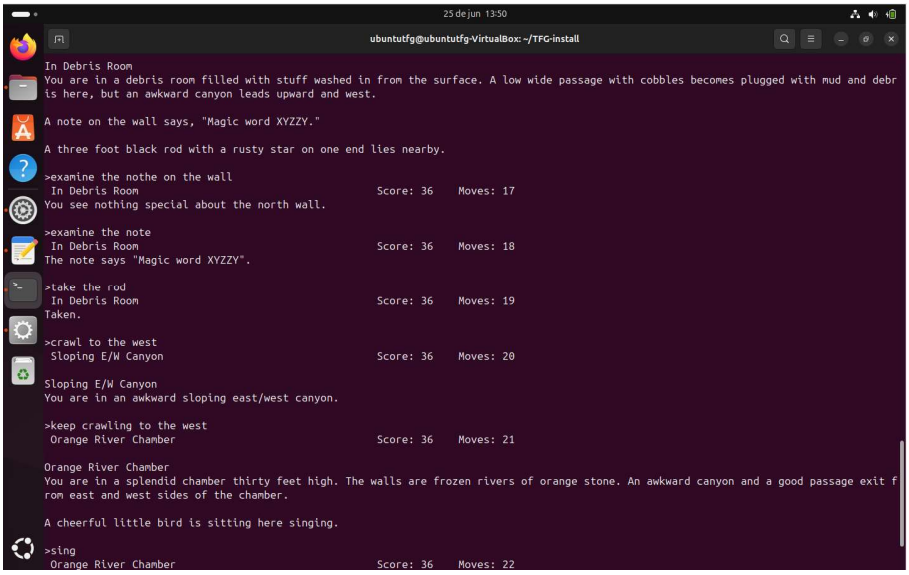

Ilustración 3.4

3.9 Pruebas con usuarios

Para las pruebas con usuarios, se hizo a los mencionados jugar a dos aventuras conversacionales mediante el programa desarrollado, así como ciertas preguntas de control. Antes de comenzar, se les hizo dos preguntas:

1. ¿Sabe lo que es una aventura conversacional?
2. ¿Ha jugado o sabe cómo se juega una aventura conversacional?

Tras esto hizo jugar a los usuarios sin proporcionarles ningún conocimiento sobre las órdenes que usan este tipo de juegos. Tras la sesión de juego se les explicó qué había sucedido mientras ellos jugaban y se les preguntó qué les había parecido.

Desde el punto de vista del hacedor del trabajo, las pruebas con distintos usuarios también constatan esta tesis, pues todos ellos fueron capaces de avanzar en ellas sin conocimiento a priori de los verbos conocidos por la aventura en cuestión. Más aún, los usuarios que conocían el concepto pero no tenían un conocimiento claro sobre cómo jugar, también pudieron jugar y avanzar, lo que recalca la versatilidad del sistema frente al procesamiento de la entrada del usuario tanto de una naturaleza más cercana al lenguaje natural, como de una naturaleza más similar a las órdenes de la máquina Glulxe. Cabe señalar también que los usuarios encontraron el concepto de este trabajo muy interesante.

3.10 Validación de los requisitos

En vista de los resultados de las pruebas finales y con los usuarios, el sistema es capaz de traducir el lenguaje natural del usuario a órdenes que comprenda la máquina Glulxe **(RF1)**.

En caso de que la respuesta de la máquina no sea comprendida, el programa es capaz de manejar correctamente la situación pidiendo a GPT que reformule la traducción o, en caso de haber agotado el número de reformulaciones posibles, mostrar la respuestas de la máquina al usuario para que vuelva a escribir lo que desea hacer **(RF2)**.

El programa está diseñado de forma robusta, de forma que, ante un error de la máquina, se informa al usuario que, efectivamente ha ocurrido un error, y mostrando información sobre el mismo si la hubiera **(RF3)**.

Es posible introducir órdenes especiales y las llamadas *palabras mágicas* de forma que GPT no tenga que interpretar estas órdenes como lenguaje natural **(RF4)**.

Los tiempos de respuesta son razonablemente aceptables, no tomando más de 5 segundos aun en el peor de los casos, esto es, que se agote el número de peticiones de reformulación de la traducción a GPT y aún así fracase **(RNF1)**.

El formato de salida de las respuestas de la máquina Glulxe es claro para el usuario, muy parecido de hecho al que so obtendría si se usase CheapGlk **(RNF2)**.

Por último, añadir el cumplimiento del requisito inicial **1** pues se permite al usuario elegir qué aventura conversacional quiere jugar mediante el menú de inicio.

4 CONCLUSIONES Y TRABAJOS FUTUROS

La opinión del hacedor es que el resultado es satisfactorio y a la vez sorprendente. GPT es capaz de hacer una traducción del lenguaje natural a órdenes con una gran precisión sin un entrenamiento previo. Además, los tiempos de latencia son razonablemente tolerables. Hay que señalar que esta implementación, si bien engloba la mayoría de estos juegos, se restringe a juegos de aventuras conversacionales cuyo modelo sigue al de *Colossal Cave Adventure*. Algunas aventuras conversacionales tienen sistemas de juego muy particulares y requerirían su propia implementación de este trabajo.

Ahora bien, el resultado también es limitado. Esto se debe a que los juegos, al funcionar como laberintos y puzles están significativamente delimitados en las posibilidades de acción que ofrecen al jugador. Esto es claro si se compara este proyecto con *Façade*. El diseño de este último es idóneo para que el jugador emplee el lenguaje natural y permite esta inmersión. El resultado es variable y está sujeto a las acciones del jugador. En la mayoría de aventuras conversacionales el resultado no es variable sino que se puede completar o no el juego con mayor o menor perfección. Las posibilidades de interacción son limitadas y el uso de NPCs es escaso y con unas opciones de interacción también limitadas.

Sin embargo, las posibilidades que ofrecen el PLN y modelos como GPT para los videojuegos son interesantes, en concreto, los videojuegos enfocados al rol y a la interacción con otros NPCs del mundo. Videojuegos como Gothic, Fallout, The Elder Scrolls, Baldur's Gate, The Witcher, S.T.A.L.K.E.R. por citar algunos ejemplos, podrían incorporar estos sistemas a sus sistemas de diálogo, emancipándose de los menús de opciones predefinidas y ofreciendo al jugador una interfaz mucho más inmersiva y natural.

Como mejoras futuras, se podría incorporar el jugar mediante síntesis de voz en lugar de usar texto o añadir música inferida del contexto a cada escenario de la aventura conversacional de forma automática.

Otra mejora consistiría en añadir este trabajo al realizado por compañeros anteriormente en el que se podía jugar a través de un bot de telegram, de forma que los usuarios podrían jugar a través de dicha aplicación sin conocer necesariamente las órdenes del juego.

Una última mejora podría ser conseguir que GPT fuese consciente del contexto del escenario y del jugador del escenario en la aventura conversacional, pudiendo aumentar aún más la flexibilidad del sistema frente a la entrada del usuario.

Como reflexión final, ha sido satisfactorio y entretenido trabajar en este TFG. Creo que para muchos estudiantes, entre los que me incluyo, es la primera vez participamos en un proyecto informático como tal. A diferencia de las prácticas y ejercicios presentes en las diversas asignaturas, es un trabajo a mayor escala en el que, además, la documentación y la rigurosidad del proceso gozan de gran importancia, a diferencia de la mayoría de los ejercicios previamente mencionados. Asimismo, el TFG me ha permitido explorar tecnologías novedosas, y explorar tareas y conceptos que apenas he tratado en el grado, como son el manejo de APIs en general e invocación de procesos y posterior comunicación con ellos.

5 APÉNDICES

5.1 Guía original del Trabajo Fin de Título[18]

(Cód.: 23/24-3988) Juegos de aventuras conversacionales mejorados con sistemas de diálogo

- Tutor del TFG: ARTURO MONTEJO RAEZ
- Modalidad: Proyecto de Ingeniería | Tipo: TFG Específico
- Número máximo de estudiantes: 1 (1 asignados)
- Idioma: Castellano
- Asignado al alumno con DNI:26510180N

[18] Enlace: http://eps-anterior.ujaen.es/TFGtemporal/mostrarTFG.php?id=3988

Las aventuras conversacionales son un género de juegos por ordenador con una larga historia. El proyecto propone la mejora de la experiencia de juego mediante la aplicación de sistemas de diálogo avanzados que posibiliten mayor libertad de expresión en el proceso de comunicación jugador-juego. Para ello se estudiará la viabilidad de sistemas basados en intenciones dentro de la estrategia conversacional.

Conocimientos Previos

Se recomienda, aunque no es obligatorio, haber cursado la asignatura de procesamiento del lenguaje natural.

Objetivos del TFG

- Conocer cómo funcionan las aventuras conversacionales y las plataformas/tecnologías existentes.
- Diseñar un sistema de diálogo que flexibilice la comunicación entre el jugador y la plataforma.
- Integrar el sistema de diálogo en una arquitectura para aventuras conversacionales popular.
- Evaluar, mediante un grupo usuarios de prueba, la validez de la solución.

Metodología a Desarrollar

1. Estudio de las plataformas conversacionales actuales.
2. Estudio de los sistemas de diálogo actuales.
3. Selección de tecnologías para la ejecución de aventuras conversacionales.
4. Selección de tecnologías para el sistema de diálogo.
5. Desarrollo del sistema de diálogo.
6. Integración de tecnologías en el sistema final.
7. Pruebas y validación del sistema con algunos títulos populares

Documentos y Formatos de Entrega

- Repositorio en GitHub con el código del proyecto.
- Memoria del proyecto.
- Manual de instalación y despliegue.

5.2 Documentación de entrada

Como documentación previa para la realización de este TFG, se proporcionaron dos TFGs anteriores:

- Medina Íñiguez, Miguel Ramón (2018) *Bot para Telegram que ejecute aventuras conversacionales.* [Trabajo fin de grado, Universidad de Jaén]. https://hdl.handle.net/10953.1/8436
- García Quesada, Manuel Jesús (2017) *Bot para Telegram que ejecute aventuras conversacionales.* [Trabajo fin de grado, Universidad de Jaén]. https://hdl.handle.net/10953.1/5944

5.3 Instalación y configuración del sistema

A continuación, se va a proporcionar una guía sobre la instalación y puesta a punto de un sistema para usar y modificar el programa y la máquina Glulxe. Se va a optar por un sistema GNU/Linux y la distribución escogida será Ubuntu. Esta elección se debe a que es el sistema que ha usado el hacedor a la hora de desarrollar el software. Se va a partir de un Ubuntu recién instalado en su última versión, la 24.04. No se va a tratar la instalación del IDE pues se considera ajeno a la finalidad de este trabajo. El código fuente del proyecto se encuentra disponible aquí.

5.3.1 Comprobaciones iniciales.

El primer paso será comprobar que se tenga la última versión de Python instalada. En caso contrario, el proceso de instalación es sencillo y bien documentado.

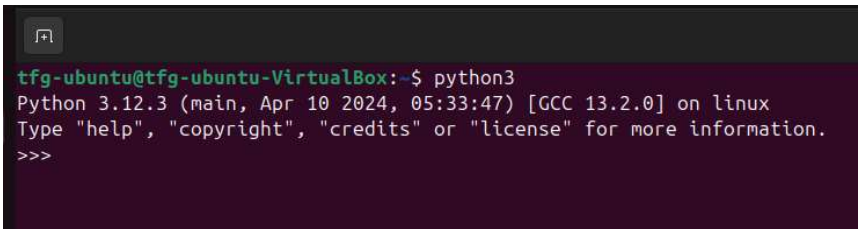

Ilustración 5.1

Asimismo hay que comprobar que se tenga instalado algún compilador para el lenguaje C, idealmente el compilador GCC. Si no estuviese instalado, la orden para ello en Ubuntu es: `sudo apt install gcc`.

De la misma forma hay que instalar la utilidad Make. La orden para ello en Ubuntu es: sudo `apt install make`.

5.3.2 Instalación de la máquina Glulxe

Tras estas comprobaciones el siguiente paso será la instalación de la máquina Glulxe. Para ello hay que descargar tanto el código fuente de la máquina Glulxe[19] como la biblioteca RemGlk[20]. Se nos descargarán sendos comprimidos con el software correspondiente que habrán de descomprimirse.

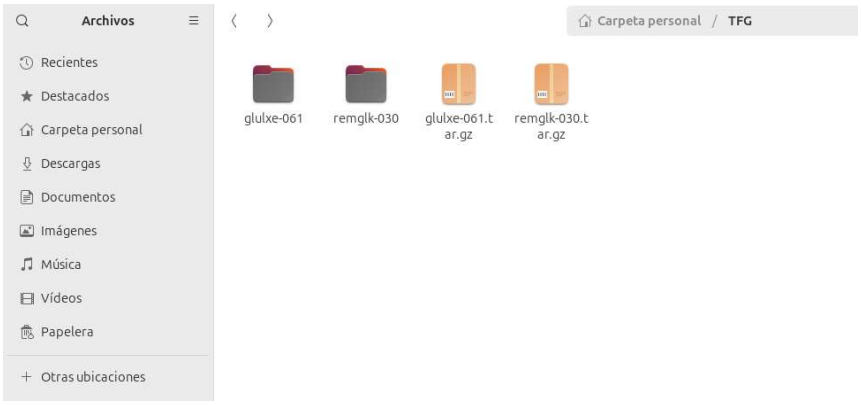

Ilustración 5.2

Una vez hecho esto, dentro de la carpeta `glulxe-061/glulxe` habrá una gran cantidad de archivos, siendo el `Makefile` el que nos interesa. Este archivo contiene las instrucciones[21] necesarias para el compilado del código de la máquina Glulxe, así como la "vinculación" con la biblioteca RemGlk.

En la primera sección se encuentran las rutas para la biblioteca a utilizar. Lo único que hay que hacer es comentar todas las instrucciones salvo las de la biblioteca RemGlk. Hay que asegurarse que las rutas son adecuadas, en caso contrario aparecerá un error de compilación.

[19] Enlace: https://eblong.com/zarf/glulx/index.html
[20] Enlace: https://eblong.com/zarf/glk/index.html
[21] Están explicadas de forma detallada dentro del propio Makefile pero están en inglés.

```
# Unix Makefile for Glulxe.

# To use this, you must set three variables. GLKINCLUDEDIR must be the
# directory containing glk.h, glkstart.h, and the Make.library file.
# GLKLIBDIR must be the directory containing the library.a file.
# And GLKMAKEFILE must be the name of the Make.library file. Two
# sets of values appear below; uncomment one of them and change the
# directories appropriately.

#GLKINCLUDEDIR = ../cheapglk
#GLKLIBDIR = ../cheapglk
#GLKMAKEFILE = Make.cheapglk

#GLKINCLUDEDIR = ../glkterm
#GLKLIBDIR = ../glkterm
#GLKMAKEFILE = Make.glkterm

#GLKINCLUDEDIR = ../xglk
#GLKLIBDIR = ../xglk
#GLKMAKEFILE = Make.xglk

GLKINCLUDEDIR = ../remglk
GLKLIBDIR = ../remglk
GLKMAKEFILE = Make.remglk

#GLKINCLUDEDIR = ../gtkglk/src
#GLKLIBDIR = ../gtkglk
#GLKMAKEFILE = ../Make.gtkglk
```

Ilustración 5.3

Para ello hay que colocar la carpeta de la biblioteca RemGlk al mismo nivel que la carpeta `glulxe`.

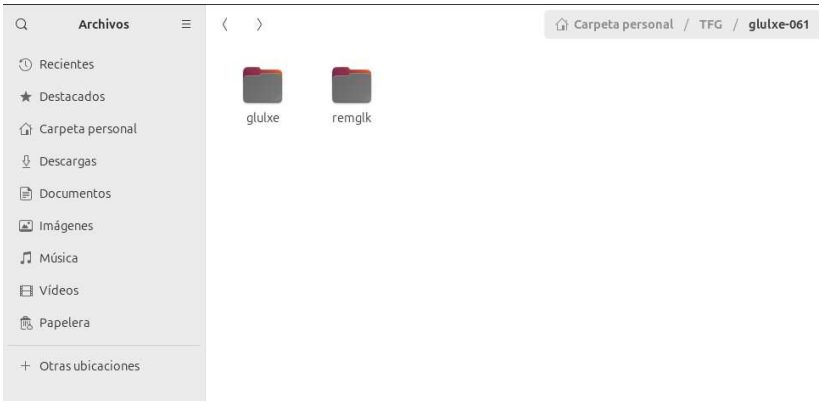

Ilustración 5.4

En la siguiente sección hay que escoger el compilador e indicar es sistema operativo para la compilación. Al estar usando un sistema GNU/Linux, también se pide que se indique un generador de números aleatorios.

```
# Also set an appropriate OS config in OPTIONS, below.
#   -DOS_MAC for MacOS
#   -DOS_WINDOWS for Windows
#   -DOS_UNIX for Unix/Linux
# For OS_UNIX, you probably also want to set a random number generator
# option. These are unfortunately not very standardized across Unixes.
# We recommend -DUNIX_RAND_GETRANDOM on Linux and -DUNIX_RAND_ARC4
# on NetBSD.
# (MacOS always uses ARC4, in case you were wondering.)

# Pick a C compiler.
#CC = cc
CC = gcc

OPTIONS = -g -Wall -Wmissing-prototypes -Wno-unused -DOS_UNIX -DUNIX_RAND_GETRANDOM
```

Ilustración 5.5

Una vez hecho esto, hay seguir un procedimiento similar para la biblioteca RemGlk. En este caso bastará con seleccionar el compilador adecuado:

```
# Unix Makefile for RemGlk library

# This generates two files. One, of course, is libremglk.a -- the library
# itself. The other is Make.remglk; this is a snippet of Makefile code
# which locates the remglk library and associated libraries.
#
# When you install remglk, you must put libremglk.a in the lib directory,
# and glk.h, glkstart.h, and Make.remglk in the include directory.

# Pick a C compiler.
#CC = cc
CC = gcc
```

Ilustración 5.6

Ahora se debe abrir una terminal en el directorio `remglk` y escribir la orden `make` obteniendo el siguiente resultado.

```
tfg-ubuntu@tfg-ubuntu-VirtualBox:~/TFG/glulxe-061/remglk$ make
gcc -g -Wall -Wno-unused    -c -o main.o main.c
gcc -g -Wall -Wno-unused    -c -o rgevent.o rgevent.c
gcc -g -Wall -Wno-unused    -c -o rgfref.o rgfref.c
gcc -g -Wall -Wno-unused    -c -o rggestal.o rggestal.c
gcc -g -Wall -Wno-unused    -c -o rgdata.o rgdata.c
gcc -g -Wall -Wno-unused    -c -o rgmisc.o rgmisc.c
gcc -g -Wall -Wno-unused    -c -o rgauto.o rgauto.c
gcc -g -Wall -Wno-unused    -c -o rgstream.o rgstream.c
gcc -g -Wall -Wno-unused    -c -o rgstyle.o rgstyle.c
gcc -g -Wall -Wno-unused    -c -o rgwin_blank.o rgwin_blank.c
gcc -g -Wall -Wno-unused    -c -o rgwin_buf.o rgwin_buf.c
gcc -g -Wall -Wno-unused    -c -o rgwin_grid.o rgwin_grid.c
gcc -g -Wall -Wno-unused    -c -o rgwin_pair.o rgwin_pair.c
gcc -g -Wall -Wno-unused    -c -o rgwin_graph.o rgwin_graph.c
gcc -g -Wall -Wno-unused    -c -o rgwindow.o rgwindow.c
gcc -g -Wall -Wno-unused    -c -o rgschan.o rgschan.c
gcc -g -Wall -Wno-unused    -c -o rgblorb.o rgblorb.c
gcc -g -Wall -Wno-unused    -c -o cgunicod.o cgunicod.c
gcc -g -Wall -Wno-unused    -c -o cgdate.o cgdate.c
gcc -g -Wall -Wno-unused    -c -o gi_dispa.o gi_dispa.c
gcc -g -Wall -Wno-unused    -c -o gi_debug.o gi_debug.c
gcc -g -Wall -Wno-unused    -c -o gi_blorb.o gi_blorb.c
ar r libremglk.a main.o rgevent.o rgfref.o rggestal.o rgdata.o rgmisc.o rgauto.o rgstream.o rgstyle.o rgwin_blank.o rgwin_buf.o rgwin_grid.o rgwin_pair.o rgwin_graph.o rgwindow.o rgsch
an.o rgblorb.o cgunicod.o cgdate.o gi_dispa.o gi_debug.o gi_blorb.o
ar: creando libremglk.a
ranlib libremglk.a
echo LINKLIBS =    > Make.remglk
echo GLKLIB = -lremglk >> Make.remglk
tfg-ubuntu@tfg-ubuntu-VirtualBox:~/TFG/glulxe-061/remglk$ gcc
```

Ilustración 5.7

Repetir el mismo procedimiento en el directorio `glulxe`.

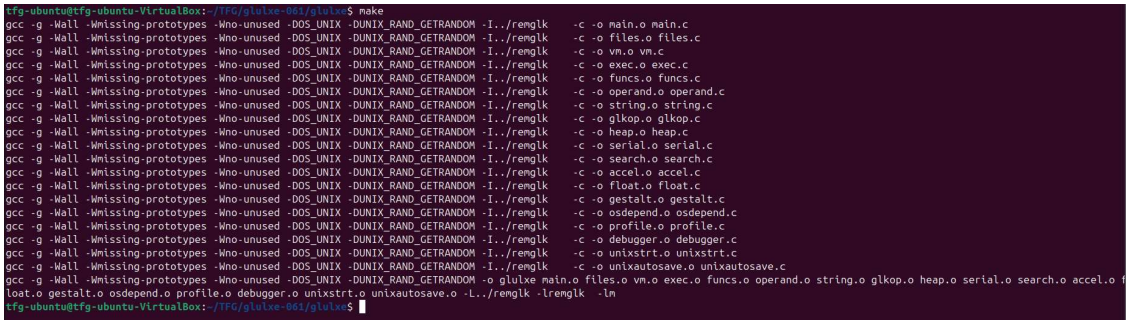

Ilustración 5.8

Si no ha habido errores y se desea probar el funcionamiento de la máquina directamente, la orden a utilizar desde la terminal en el directorio glulxe es `./glulxe -fm nombre_del_juego`.

5.3.3 Instalación de los módulos necesarios de Python

En primera instancia, es de utilidad comprobar qué módulos de los necesarios se tienen instalados. Para ello, basta con hacer un *import* de los módulos que se van a utilizar dentro de la consola de Python. En la mayoría de los casos, el resultado será el siguiente:

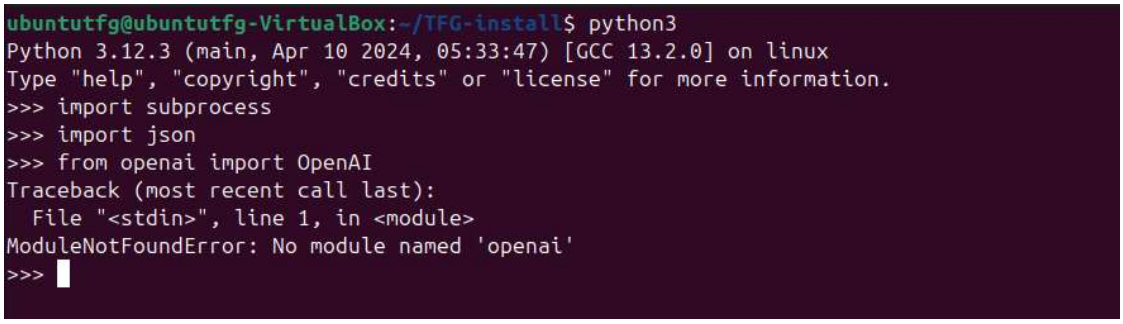

Ilustración 5.9

Para instalar cualquiera de los módulos bastaría utilizar la orden: `sudo apt install Python3-[nombre del módulo]`. En este caso, la orden sería: `apt install Python3-openai`.

5.3.4 Organización de los directorios de la instalación

Una vez instalados todos los componentes, si no se ha hecho durante el proceso de instalación, es necesario disponer los directorios de la siguiente forma:

Ilustración 5.10

De tal modo que la estructura será la siguiente:

```
.
├── config
│   ├── api_key
│   ├── comandos
│   ├── no_entendido
│   └── palabras_reservadas
├── glulxe
│   ├── código de la máquina
├── glulxe-games
│   ├── Juegos disponibles
├── logs
│   ├── guardado_glulxe.json
│   ├── log_chatgpt
│   ├── log_glulxe
│   └── log_usuario
├── main.py
├── remglk
│   ├── codigo de la librería
└── archivos de guardado
```

Ilustración 5.11

5.4 Manuales de usuario

Órdenes para realizar diversas acciones:

- Cargar partida: `restore`
- Guardar partida: `save`

- Ayuda del juego[22]: `help`
- Deshacer movimiento: `undo`
- Modo *verbose*[23]: `verbose`
- Salir: `quit`
- Repetir la orden anterior: `again`
- Esperar: `wait`
- Para introducir palabras mágicas y/o palabras clave de una aventura en concreto, debe ser escrita precedida del símbolo '#' sin espacios.

Respecto a cómo jugar, para desplazarse por el escenario se puede ir en dirección este, oeste, norte, sur, sureste, suroeste, noreste y noroeste. Se recomienda ser creativo pero a la vez lógico con lo que se desea hacer. En ocasiones, una respuesta de que una acción no es posible es resultado de no emplear la acción correcta, no de que realmente sea imposible.

6 DEFINICIONES Y ABREVIATURAS

- **Máquina Z:** máquina virtual de 16 bits desarrollada por Joel Berez y Marc Blank para Infocom para ejecutar sus juegos de aventuras conversacionales.
- **Glulx:** máquina virtual de 32 bits para ejecutar juegos de aventuras conversacionales. Similar a la máquina Z.
- **Glulxe:** acrónimo de "Glulx Execute". Es un intérprete para Glulx.
- **Glk:** estándar de interfaz para juegos de aventuras conversacionales.
- **RemGlk:** biblioteca en C que implementa la interfaz la API de Glk como objetos JSON.
- **GlkOte:** equivalente a RemGlk implementado en el lenguaje JavaScript para ser empleado por navegadores web.

[22] No siempre estará disponible. Depende del juego.
[23] Se repetirá la descripción de cada escenario, incluso aunque se haya visitado con anterioridad.

- **CheapGlk:** implementación minimalista de Glk que utiliza exclusivamente la entrada y salidas estándar.
- **Subprocess:** módulo del lenguaje Python que permite invocar otros procesos en un script de este lenguaje.
- **JSON:** acrónimo de JavaScript Object Notation, es un formato de estructura de datos. Es ampliamente utilizado en la web y, como consecuencia, en otros ámbitos en los que se requiera una estructura común para la transmisión de datos.
- **PLN:** acrónimo de Procesamiento del lenguaje natural.
- **API:** acrónimo de *Application Programming Interface* es una sección de código que permite la comunicación entre piezas de software.

7 BIBLIOGRAFÍA

Pressman, R. (2010). *Ingeniería del Software.* McGraw-Hill.

Mateas, M., Stern, A. (2004). Natural Language Understanding in Façade: Surface-Text Processing. In: Göbel, S., *et al.* Technologies for Interactive Digital Storytelling and Entertainment. TIDSE 2004. Lecture Notes in Computer Science, vol 3105. Springer, Berlin, Heidelberg.

Canal AI and Games. (22 de abril de 2022). *The Story of Facade: The AI-Powered Interactive Drama | AI and Games #49* [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=POv1cOX8xUM

Canal Pazos64. (3 de febrero de 2017). *Cuidao Ahí… Event[0]* [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=0v8-BxDVJ8Q

Canal 3Blue1Brown. (1 de abril de 2024). *But what is a GPT? Visual intro to transformers | Chapter 5, Deep Learning* [Archivo de Vídeo]. Youtube. https://www.youtube.com/watch?v=wjZofJX0v4M

Wikipedia: Façade (24 de abril de 2024). Obtenido de https://en.wikipedia.org/wiki/Fa%C3%A7ade_(video_game)

Wikipedia: AI Dungeon (24 de abril de 2024). Obtenido de https://en.wikipedia.org/wiki/AI_Dungeon#

Wikipedia: Event[0] (24 de abril de 2024). Obtenido de https://en.wikipedia.org/wiki/Event_0#

Steam: Event[0] (24 de abril de 2024). Obtenido de https://store.steampowered.com/app/470260/Event0/

RemGlk: remote-procedure-call implementation of the Glk IF API (11 de mayo de 2024). Obtenido de https://eblong.com/zarf/glk/remglk/docs.html

Glk: An Interface Standard for Interactive Fiction (11 de mayo de 2024). Obtenido de https://eblong.com/zarf/glk/index.html

Glulx: A 32-Bit Virtual Machine for IF (11 de mayo de 2024). Obtenido de https://eblong.com/zarf/glulx/

GlkOte: a Javascript library for IF interfaces (11 de mayo de 2024). Obtenido de https://eblong.com/zarf/glk/glkote.html

Python 3.12.3 documentation, The Python Standard Library, Concurrent Execution, subprocess (11 de mayo de 2024). Obtenido de https://docs.python.org/3/library/subprocess.html

OpenAI, Documentation, API reference (11 de mayo de 2024). Obtenido de https://platform.openai.com/docs/api-reference

Wikipedia: Z-machine (11 de mayo de 2024). Obtenido de https://en.wikipedia.org/wiki/Z-machine

Wikipedia: Natural language processing (23 de junio de 2024). Obtenido de https://en.wikipedia.org/wiki/Natural_language_processing

Pressman, Roger S. (2010). *Ingeniería del software. Un enfoque práctico.* McGrawHill

García Quesada, Manuel Jesús (2017) *Bot para Telegram que ejecute aventuras conversacionales.* [Trabajo fin de grado, Universidad de Jaén]. https://hdl.handle.net/10953.1/5944

Medina Íñiguez, Miguel Ramón (2018) *Bot para Telegram que ejecute aventuras conversacionales.* [Trabajo fin de grado, Universidad de Jaén]. https://hdl.handle.net/10953.1/8436