LOS SIMS LLM

THE SIMS LLM

TRABAJO FIN DE GRADO
CURSO 2024-2025

AUTORES
PABLO ZAPICO GARCÍA
GABRIELA VALENTINA DÍAZ MARÍN

DIRECTORES
JOSÉ IGNACIO REQUENO JARABO
MARÍA ELENA GÓMEZ MARTÍNEZ

GRADO EN INGENIERÍA DE SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

CALIFICACIÓN GABRIELA DIAZ: 8.5 / 10
CALIFICACIÓN PABLO ZAPICO: 8.5 / 10

THE SIMS LLM

THE SIMS LLM

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTORES
PABLO ZAPICO GARCÍA
GABRIELA VALENTINA DÍAZ MARÍN

DIRECTORES
JOSÉ IGNACIO REQUENO JARABO
MARÍA ELENA GÓMEZ MARTÍNEZ

CONVOCATORIA: JUNIO 2025

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

26 DE MAYO DE 2025

DEDICATORIA

Gabriela Díaz Marín

Dedico este trabajo a mis padres, por su apoyo incondicional en todo momento, por confiar en mí incluso cuando yo dudaba, y por enseñarme el valor del esfuerzo y la constancia.

A mis seres queridos, que han estado ahí con palabras de ánimo, paciencia y cariño en cada etapa de esta carrera.

A mis amigos nuevos, los de toda la vida y a los que esta etapa me ha regalado, que han hecho que incluso los momentos más difíciles tuvieran algo de humor o compañía.

En especial, a Pablo, Pedro, Álvaro, Javi y Washington, por haber hecho estos años juntos lo que fueron.

AGRADECIMIENTOS

Queremos agradecer a nuestros tutores María Elena Gómez Martínez y José Ignacio Requeno Jarabo por su implicación, por saber guiarnos sin imponer, por su paciencia y por estar siempre dispuestos a resolver cualquier duda con claridad y cercanía.

Gracias también a nuestros compañeros/as de clase y proyectos, con quienes hemos compartido no solo código, sino frustraciones, cafés y muchas risas que han hecho que este camino sea mucho más llevadero.

Y por supuesto, gracias a todas las personas que han aportado su granito de arena en este proyecto, aunque no lo sepan. Este trabajo es tan nuestro como de ellos.

RESUMEN

Este Trabajo de Fin de Grado presenta el desarrollo de un prototipo de videojuego de simulación espacial que integra modelos de lenguaje de gran tamaño (Large Lenguage Model – LLM :) [1] mejorar la interacción entre el jugador y los personajes no jugadores (Non Playable Characters – NPCs). El objetivo principal es permitir al jugador comunicarse con los tripulantes de la nave utilizando lenguaje natural, facilitando así la gestión de tareas como la investigación, la exploración o la administración de recursos.

El videojuego ha sido desarrollado con el motor Unity [2], y se ha optado por utilizar la interfaz de programación de aplicaciones [3] (Application Programming Interface – API) de Gemini [4] debido a su compatibilidad y facilidad de integración. La incorporación de un LLM permite que los NPCs actúen de forma autónoma y respondan de manera coherente a las instrucciones del jugador, generando una experiencia más inmersiva y dinámica. Este trabajo busca demostrar el potencial de los LLM en el desarrollo de videojuegos con interacción más natural y adaptativa.

Palabras clave

Videojuego, Unity, LLM, IA generativa, tripulantes, simulación, comunicación natural, acciones autónomas, JSON, inventario, interfaz, exploración espacial

ABSTRACT

Language Models (LLM) for Game Control

This Bachelor's Thesis presents the development of a prototype space simulation video game that integrates Large Language Models (LLMs) [1] to enhance interaction between the player and non-playable characters (NPCs). The main objective is to enable the player to communicate with the ship's crew using natural language, thereby facilitating the management of tasks such as research, exploration, and resource administration.

The video game was developed using the Unity engine [2], and the Gemini [4] Application Programming Interface (API) [3] was chosen for its compatibility and ease of integration. The incorporation of an LLM allows NPCs to act autonomously and respond coherently to player instructions, creating a more immersive and dynamic experience. This work aims to demonstrate the potential of LLMs in the development of video games with more natural and adaptive interaction.

Keywords

Video game, Unity, LLM, generative AI, crew members, simulation, natural communication, autonomous actions, JSON, inventory, interface, space exploration

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

En la actualidad, el sector de los videojuegos se enfrenta a un cambio de paradigma: los jugadores demandan experiencias cada vez más personalizadas, inmersivas y reactivas a sus decisiones. Las arquitecturas tradicionales basadas en menús y sistemas predefinidos limitan tanto la naturalidad de la interacción como la profundidad narrativa. Al mismo tiempo, los Modelos de Lenguaje de Gran Tamaño (LLM) han madurado hasta ofrecer una comprensión y generación de texto en lenguaje natural de alta calidad, posibilitando nuevas formas de diálogo, toma de decisiones y generación de contenido dinámico.

Nuestro proyecto aprovecha este avance para integrar un LLM directamente en Unity, con el fin de que los personajes no jugables - NPC de un simulador espacial no solo ejecuten acciones exactas según órdenes de texto, sino que también interpreten el contexto de la nave, sus propios estados (hambre, ánimo, habilidades) y el entorno procedimental de los planetas. De este modo, cada interacción se transforma en una experiencia emergente: los tripulantes pueden argumentar sus decisiones, priorizar tareas según su personalidad y adaptarse a eventos imprevistos generados por la IA.

Esta aproximación aporta varias innovaciones clave:

- **Narrativa Dinámica**: las reacciones de los NPCs se generan en tiempo real, evitando interacciones estáticas.
- **Control Natural**: el jugador se comunica con texto libre, desplazando los esquemas de control basados en botones y menús.
- **Profundidad de Simulación**: al combinar LLM, sistemas de progresión tecnológica y gestión de recursos, logramos un mundo de juego coherente donde cada decisión del jugador tiene consecuencias tangibles.

Al explorar esta convergencia entre IA conversacional y entretenimiento interactivo, sentamos las bases para futuros desarrollos de videojuegos más humanos, adaptativos y capaces de ofrecer experiencias únicas en cada partida.

1.2 Objetivos

El principal objetivo de este Trabajo de Fin de Grado es desarrollar un videojuego de simulación espacial en un motor gráfico de dos dimensiones cuyo control de NPCs y mecánicas clave se realice mediante órdenes en lenguaje natural procesadas por un LLM. Para lograr el correcto desenlace de este objetivo se debe cumplir con una serie de objetivos específicos:

- Obtener un conocimiento sólido de las capacidades y limitaciones de los Modelos de Lenguaje de Gran Tamaño (LLM) aplicados a videojuegos, investigando tanto la literatura académica como la documentación de APIs para seleccionar la opción más adecuada a nuestras necesidades.

- Diseñar y validar un formato uniforme de comunicación entre el motor gráfico y el LLM, definiendo la estructura de las consultas (contexto del juego, parámetros de acción, estado de NPC) y el esquema de respuesta en una estructura específica que permita traducir decisiones del modelo en acciones concretas dentro del motor.

- Desarrollar el núcleo de interacción con el LLM, implementando un módulo en el lenguaje de programación C# capaz de enviar solicitudes asíncronas a la API, gestionar reintentos ante errores de red o formato, y exponer una interfaz interna que alimente la lógica de los tripulantes.

- Implementar la generación automática de tripulantes mediante prefabricados, dotándolos de nombres y personalidades aleatorias extraídos de ficheros de tipo objeto de notación de Java Script [5]

(JavaScript Object Notation - JSON), y garantizar que este proceso pueda ejecutarse tanto al inicio de la partida como en cualquier punto durante la simulación.

- Crear las mecánicas de gestión de recursos y progresión tecnológica de la nave, incluyendo la definición y carga de inventarios (recursos y materiales), el almacenamiento de parámetros de acción, y la aplicación real de sus efectos (hambre, investigación, combustible, fabricación) sobre el estado del juego.

- Desarrollar las interfaces de interacción clave asegurando una experiencia clara, evitando acciones inválidas y facilitando la inmersión.

- Probar y depurar el comportamiento concurrente de múltiples tripulantes, afinando la gestión de la pila de acciones y las condiciones de entrada/salida en los módulos, para garantizar una simulación fluida.

Con este conjunto de objetivos llevados a cabo de forma correcta se podrá llegar a asegurar que el objetivo principal ha sido cumplido.

1.2 Plan de trabajo

Para la planificación y ejecución de este proyecto se ha decidido emplear una metodología ágil basada en Scrum [6], partiendo el trabajo en "Sprints", ciclos de trabajo de cierta duración que permiten iterar rápidamente al principio del proyecto y estructurar la etapa del desarrollo.

En cada sprint se incluirá tareas concretas que realizar. En las primeras etapas entregables claros y en las etapas de desarrollo utilizando historias de usuario (HU).

Estas tareas podrán estar en cinco diferentes estados:

1. TO DO
2. IN PROGRESS

3. ON HOLD
4. UNDER REVIEW
5. DONE

Además, se realizarán reuniones al finalizar cada sprint para analizar qué tareas se completaron y cuales se tienen que seguir trabajando en el siguiente sprint.

Para la gestión del proyecto se ha utilizado la herramienta Jira [7] (Ilustración 1), donde se organizaron todas las tareas y HU (Ilustración 3, Ilustración 4), permitiendo mantener una trazabilidad clara del trabajo realizado en cada sprint, los cuales también se gestionaron mediante dicha herramienta.

El proyecto fue trabajado y almacenado en la plataforma colaborativa de GitHub en forma de repositorio, cuya URL es: https://github.com/TGF-2024-25/sims

1.3.1 Fases del proyecto

Para el desarrollo basándose en Scrum se ha organizado el trabajo en ciclos de duración variable (sprints) y empleando Jira como herramienta de seguimiento (Ilustración 2). Cada fase corresponde a uno o varios sprints y asegura un avance ordenado desde la investigación inicial hasta la entrega final.
El trabajo arrancó en septiembre de 2024 dando lugar a las fases:

1.3.1.1 Investigación y diseño

Esta fase se corresponde con los capítulos 2.2 "Estado del arte" y 2.3 "Tecnologías". En las primeras semanas (Septiembre-Octubre) se realizó una revisión bibliográfica sobre la integración de LLM en videojuegos, las arquitecturas cliente-servidor en el motor gráfico y las APIs. Paralelamente, se exploró la viabilidad técnica de comunicarse con un LLM desde el motor mediante protocolos HTTP/REST y JavaScript Object Notacion - JSON. El resultado de esta fase fue una conclusión de viabilidad que estableció los fundamentos conceptuales y tecnológicos del proyecto.

1.3.1.2 Definición de requisitos inicial

Esta fase y la siguiente se corresponde con los capítulos capítulo 3.1 "Requisitos funcionales" y 3.2 "Progresión tecnológica y recursos". Con los conocimientos adquiridos, se elaboró el backlog inicial en Jira, entre Octubre y Noviembre, desglosando los primeros requisitos funcionales en historias de usuario y tareas. Se validó este conjunto, priorizando funcionalidades críticas como la comunicación LLM con Unity y movimiento básico de tripulantes.

1.3.1.3 Refinamiento de requisitos

Durante varias sesiones de revisión en Jira (Diciembre-Enero) se refinó el backlog tanto funcional como no funcional. Se ajustaron criterios de aceptación y se definieron los límites del alcance (scope) para el MVP.

1.3.1.4 Diseño diagramas

Esta fase se corresponde con el capítulo 4 "Diagramas de diseño". Entre enero y febrero se modelaron los principales artefactos UML [8]:

- Diagrama de clases (Ilustración 5) describiendo los elementos principales como Partida, Nave, Tripulante, Instalaciones y gestor LLM.
- Diagramas de secuencia, uno para la conexión con la API de Gemini y otro para la toma de decisiones del tripulante.
- Diagramas de flujo de eventos y de progresión tecnológica. Estos diagramas sirvieron de guía para el desarrollo y quedaron recogidos en el capítulo 3.

1.3.1.5 Desarrollo de funcionalidades principales

Esta fase se corresponde con los capítulos del 5.1 "Script inicial de comunicación con LLM" al 5.14 "Generación automatizada de NPC". Entre enero y marzo se implementó el pipeline de comunicación con el LLM (envío de contexto, recepción de acciones), el sistema básico de control de tripulantes, el manejador de interacciones y la UI adaptable con sorteo de lienzos y controladores de animación. Al final de esta fase, el juego permitía a los

tripulantes reconocer órdenes, moverse, generar acciones autónomas basándose en información propia y ajena y aplicar sus efectos a la nave, entre otros.

1.3.1.6 Desarrollo de mecánicas avanzadas

Esta fase se corresponde a los capítulos del 5.15 "Mecánicas de investigación y fabricación" al 5.18 "Versiones finales". Entre marzo y abril se añadieron:

- Módulo de gestión de recursos y progresión tecnológica (nivel de nave, fabricación, investigaciones).
- Generación y gestión de múltiples entidades de tripulantes.
- Sistema de navegación, generación de planetas, reclutamiento de tripulantes, exploración y gestión de recompensas.
- Versión final del modelo de comunicación y conversión de datos con el LLM.
- Implementación gráfica de interfaces para gestión de tripulantes, inventario, instalaciones, entre otros.

1.3.1.7 Cierre de desarrollo y pruebas

Esta fase se corresponde al final del capítulo 5 "Implementación" y al capítulo 6 "Pruebas". A finales de abril mediante pruebas de juego se depuraron fallos de escalado de UI y de comportamiento de los NPC, así como fallos derivados de la presencia de múltiples instancias de tripulantes. Se solucionaron ajustando las colisiones de las diferentes instalaciones y su comportamiento al gestionar las diferentes instancias de tripulante durante las acciones. Tras solucionar estos problemas se crearon, ejecutaron y redactaron las pruebas.

1.3.1.8 Finalización de redacción de memoria

Durante mayo se revisaron los capítulos 1 y 2 (introducción, estado del arte, tecnologías), se integraron diagramas y tablas comparativas en el capítulo 3, se

completó la sección de desarrollo, y se redactaron las conclusiones y líneas de trabajo futuro. Finalmente, se revisaron citas, se actualizó el índice automático y se exportó la versión PDF para su entrega.

Ilustración 1: Diagrama de Jira

	Tipo	Clave	Resumen	Estado	Comentarios	Sprint	Persona asignada	Fecha de vencimiento	Etiquetas	Creada	Actualizado
☐	✧	TFGSIMS-1	Escritura de la memoria	TAREAS POR HA...	Añadir comentario			28 abr 2025		9 sept 2024	27 abr 2025
☐	☑	TFGSIMS-94	Estado del arte	UNDER REVIEW	Añadir comentario	TFGSIMS Sprint 13	GD Gabriela Díaz			7 oct 2024	31 mar 2025
☐	☑	TFGSIMS-93	Tecnologías	UNDER REVIEW	Añadir comentario	TFGSIMS Sprint 13	PG PABLO ZAPICO GAR...			7 oct 2024	31 mar 2025
☐	☑	TFGSIMS-97	Diagrama de secuencia y HU en memoria	TAREAS POR HA...	Añadir comentario	TFGSIMS Sprint 13				18 nov 2024	31 mar 2025
☐	✧	TFGSIMS-2	Análisis de Requisitos	TAREAS POR HA...	Añadir comentario					9 sept 2024	25 sept 2024
☐	☑	TFGSIMS-4	Revisión del estado del arte	FINALIZADA	Añadir comentario	TFGSIMS Sprint 9	PG PABLO ZAPICO GAR...			9 sept 2024	15 ene 2025
⠿☐	☑	TFGSIMS-96	Priorizar HU	FINALIZADA	Añadir comentario	TFGSIMS Sprint 8				18 nov 2024	19 dic 2024
☐	☑	TFGSIMS-5	Definición de los requisitos funcionales	FINALIZADA	Añadir comentario	TFGSIMS Sprint 8	GD Gabriela Díaz			9 sept 2024	19 dic 2024
☐	☑	TFGSIMS-3	Exploración de tecnologías	FINALIZADA	Añadir comentario	TFGSIMS Sprint 2				9 sept 2024	7 oct 2024

Ilustración 2: Tareas de planificación de Jira

		TFGSIMS-6	Historias de usuario	TAREAS POR HA...	Añadir comentario					23 sept 2024	23 sept 2024	JOSE IGNACIO REQ...
		TFGSIMS-16	Dar orden a tripulante	FINALIZADA	1 comentario	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	28 ene 2025	PABLO ZAPICO GAR...
		TFGSIMS-25	Mover tripulante	FINALIZADA	2 comentarios	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	28 ene 2025	PABLO ZAPICO GAR...
		TFGSIMS-37	Seleccionar explonadores	FINALIZADA	2 comentarios	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-51	Mostrar interfaz de navegación	FINALIZADA	Añadir comentario	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-26	Mostrar interfaz de tripulante	FINALIZADA	3 comentarios	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
	+	TFGSIMS-23	Interacción de tripulante con instalación	FINALIZADA	3 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-54	Fabricar material	FINALIZADA	3 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-65	Rellenar combustible	FINALIZADA	5 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-59	Completar investigación	FINALIZADA	5 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-22	Evaluar acción a tomar	FINALIZADA	3 comentarios	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-21	Generar tripulante nuevo	FINALIZADA	Añadir comentario	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-62	Comer	FINALIZADA	4 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-36	Iniciar expedición	FINALIZADA	4 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-63	Mostrar interfaz de cocina	FINALIZADA	2 comentarios	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-47	Modificar inventario de nave	FINALIZADA	Añadir comentario	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-53	Mostrar interfaz de almacén	FINALIZADA	2 comentarios	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-50	Viajar a nuevo planeta	FINALIZADA	Añadir comentario	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-39	Terminar expedición	FINALIZADA	2 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-52	Mover recursos	FINALIZADA	3 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
		TFGSIMS-29	Transformar output en acciones de juego	FINALIZADA	2 comentarios	TFGSIMS Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	17 feb 2025	PABLO ZAPICO GAR...
		TFGSIMS-28	Recibir output del LLM	FINALIZADA	1 comentario	TFGSIMS Sprint 1	Gabriela Díaz			1 oct 2024	17 feb 2025	PABLO ZAPICO GAR...

Ilustración 3: Tareas de Jira(1)

	TFGSIM5-37	Pausar investigacion	FINALIZADA	Añadir comentario	TFGSIM5 Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	22 feb 2025	PABLO ZAPICO GAR...
	TFGSIM5-56	Iniciar investigación	FINALIZADA	2 comentarios	TFGSIM5 Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	22 feb 2025	PABLO ZAPICO GAR...
	TFGSIM5-48	Modificar nivel de combustible	FINALIZADA	2 comentarios	TFGSIM5 Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	22 feb 2025	PABLO ZAPICO GAR...
	TFGSIM5-58	Reanudar investigación	FINALIZADA	Añadir comentario	TFGSIM5 Sprint 1	PABLO ZAPICO GAR...			1 oct 2024	22 feb 2025	PABLO ZAPICO GAR...
	TFGSIM5-8	Crear nueva partida	EN CURSO	2 comentarios	TFGSIM5 Sprint 13	Gabriela Diaz			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
	TFGSIM5-69	Mejorar nave	EN CURSO	Añadir comentario	TFGSIM5 Sprint 13	PABLO ZAPICO GAR...			1 oct 2024	31 mar 2025	PABLO ZAPICO GAR...
	TFGSIM5-17	Mostrar interfaz de trabajo	TAREAS POR HA...	3 comentarios					1 oct 2024	9 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-19	Asignar/Quitar tripulante de trabajo	TAREAS POR HA...	3 comentarios					1 oct 2024	9 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-55	Mostrar interfaz de taller	TAREAS POR HA...	2 comentarios					1 oct 2024	9 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-38	Comenzar a tripulante de formar parte en expedición	TAREAS POR HA...	Añadir comentario		Elena Gómez-Martín...			1 oct 2024	5 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-40	Generar evento aleatorio	TAREAS POR HA...	3 comentarios					1 oct 2024	9 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-44	Gestionar evento	TAREAS POR HA...	Añadir comentario					1 oct 2024	18 nov 2024	PABLO ZAPICO GAR...
	TFGSIM5-32	Terminar día	TAREAS POR HA...	1 comentario					1 oct 2024	16 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-61	Mostrar interfaz de laboratorio	TAREAS POR HA...	2 comentarios					1 oct 2024	8 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-66	Eliminar partida	TAREAS POR HA...	1 comentario					1 oct 2024	16 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-67	Guardar partida	TAREAS POR HA...	2 comentarios					1 oct 2024	16 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-68	Cargar partida	TAREAS POR HA...	Añadir comentario					1 oct 2024	18 nov 2024	PABLO ZAPICO GAR...
	TFGSIM5-10	Crear personaje	TAREAS POR HA...	2 comentarios					1 oct 2024	8 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-12	Elegir tripulación inicial	TAREAS POR HA...	3 comentarios					1 oct 2024	9 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-70	Recibir misión	TAREAS POR HA...	2 comentarios					1 oct 2024	8 dic 2024	PABLO ZAPICO GAR...
	TFGSIM5-71	Completar misión	TAREAS POR HA...	Añadir comentario					1 oct 2024	25 nov 2024	PABLO ZAPICO GAR...
	TFGSIM5-72	Fin de juego	TAREAS POR HA...	Añadir comentario					1 oct 2024	25 nov 2024	PABLO ZAPICO GAR...
	TFGSIM5-20	Terminar día temprano	TAREAS POR HA...	Añadir comentario					1 oct 2024	2 dic 2024	PABLO ZAPICO GAR...

Ilustración 4: Tareas de Jira(2)

2 Estado de la cuestión

2.1 Conceptos previos

Para facilitar la lectura de esta sección, a continuación, se definen de manera breve y clara todos los términos y tecnologías que se emplean a lo largo del documento:

- **LLM (Large Language Model)**: Modelo de lenguaje a gran escala que comprende y genera texto en lenguaje natural; aquí se usa para decidir acciones de personajes y generar descripciones de eventos.
- **NPC (Non-Player Character)**: Personaje controlado por el juego o por un LLM, que simula comportamientos autónomos en lugar de respuestas preprogramadas.
- **Unity**: Motor de desarrollo de videojuegos que proporciona herramientas de renderizado 2D/3D, físicas, UI y programación en C#, usado aquí para crear el prototipo de simulador espacial.
- **API (Application Programming Interface)**: Conjunto de reglas y puntos de acceso que permite a Unity comunicarse con el LLM enviando peticiones y recibiendo respuestas.
- **TileMap**: Sistema de videojuegos que permite construir niveles 2D mediante mosaicos (tiles) reutilizables organizados en una cuadrícula, gestionando colisiones y navegación sobre el escenario.
- **Prefab**: Plantilla reutilizable en Unity que agrupa objetos de juego con componentes y configuraciones predefinidas (por ejemplo, el "Tripulante"), para instanciar múltiples copias de forma sencilla.
- **NavMesh**: Sistema de navegación de Unity que genera una malla de zonas transitables y componentes de agente para mover personajes (tripulantes) evitando obstáculos y otro tráfico.

- **Canvas y UI (Imagen, Botón, Layout)**: Conjunto de componentes para crear interfaces gráficas en Unity.
- **JSON**: Formato de texto ligero para intercambio de datos. Se usa para definir parámetros de acciones, listados de nombres/personalidades y catálogo de recursos/materiales.
- **Prompt**: Texto estructurado que se envía al LLM, conteniendo contexto de juego, estado de la nave, parámetros posibles y la instrucción del jugador o del sistema.
- **Parser**: Módulo que procesa la respuesta del LLM, extrae la acción elegida y sus parámetros, valida sus valores y construye el objeto de acción correspondiente.
- **Co-rutinas**: Mecanismo de Unity que permite ejecutar funciones a lo largo de varios ciclos.
- **Acción de juego**: Clase que representa una acción genérica de tripulante. Sus subclases (EatAction, RefillAction, ResearchAction, CraftAction) implementan la lógica específica de cada tarea.
- **Cola de acciones (Queue)**: Estructura de datos donde se encolan las acciones de juego para que el tripulante los ejecute secuencialmente.

Esta lista cubre todos los conceptos y componentes esenciales que aparecen en las secciones siguientes. Si surge algún término adicional, su definición se añadirá al pie de página o mediante aclaración puntual.

2.2 Estado del arte

Los modelos de lenguaje de gran tamaño (LLM, por sus siglas en inglés) han revolucionado la interacción hombre-máquina gracias a su capacidad para procesar y generar texto de manera coherente y contextual. Basados en redes neuronales

profundas y entrenados en grandes cantidades de datos textuales, los LLM son capaces de comprender e interpretar el lenguaje natural con un nivel de precisión sin precedentes. Ejemplos como GPT de OpenAI [9], Bard de Google [10] y LLaMA de Meta [11] han demostrado aplicaciones que van desde asistentes virtuales hasta generación creativa de contenido.

En el ámbito de los videojuegos, los LLM permiten nuevas formas de interacción, donde los diálogos y las decisiones de los personajes controlados por IA se generan dinámicamente, ampliando las posibilidades de narrativa y jugabilidad. Sin embargo, su integración en videojuegos aún está en etapas tempranas, y presenta desafíos técnicos relacionados con el rendimiento, la coherencia contextual y los requisitos computacionales.

Históricamente, la inteligencia artificial en videojuegos [12] ha dependido de técnicas como scripts predefinidos, máquinas de estados finitos [13] y árboles de decisiones [14]. Estas técnicas, aunque efectivas en su momento, limitaban las posibilidades de interacción entre el jugador y los personajes no jugadores (NPCs). Un hito clave en la evolución de la IA fue la franquicia de *The Sims* [15] de Electronic Arts, donde los NPCs gestionaban necesidades y deseos mediante árboles de comportamiento. Este sistema les permitía tomar decisiones dinámicas dentro de un marco de prioridades, creando una experiencia más autónoma e inmersiva. Sin embargo, estas decisiones seguían siendo limitadas a un conjunto de reglas predeterminadas, sin la capacidad de generar diálogos o respuestas contextuales únicas, como lo hacen los LLM.

En los últimos años, el uso de LLM en videojuegos ha generado experimentos interesantes. Por ejemplo, *AI Dungeon* [16] utiliza LLM para crear una narrativa completamente no lineal, donde las respuestas del sistema se adaptan a las entradas del jugador, ofreciendo una experiencia personalizada. Otro ejemplo es la experimentación con aplicaciones adyacentes en *Minecraft* [17], donde los LLM han

permitido dotar de mayor autonomía a NPCs y facilitar interacciones más naturales y complejas.

Estas aplicaciones demuestran la capacidad de los LLM para generar interacciones y narrativas emergentes, pero también ponen de manifiesto desafíos como:

- **Rendimiento computacional:** Requieren recursos significativos para procesar texto en tiempo real.
- **Coherencia narrativa:** Es necesario garantizar que las respuestas del LLM mantengan consistencia en contextos prolongados.
- **Diseño de experiencias equilibradas:** Integrar LLM de manera que complemente, y no reemplace, las mecánicas de juego tradicionales.

Nuestro proyecto toma elementos clave de videojuegos como *The Sims*, *AI Dungeon* y *Minecraft*. De *The Sims*, adoptamos el enfoque indirecto del jugador, que gestiona el entorno mientras los NPCs actúan de manera autónoma. De *AI Dungeon*, integramos la capacidad de los LLM para generar textos dinámicos y personalizados. Finalmente, de *Minecraft*, adoptamos la idea de autonomía extendida, donde los NPCs interactúan con el entorno y adaptan su comportamiento según una serie de datos internos y externos.

En este sentido, el proyecto busca combinar estas características (Tabla 1) en un entorno de simulación espacial, donde el jugador gestione una nave mientras interactúa con una tripulación controlada por LLM, que toma decisiones complejas de manera autónoma y dinámica.

Tabla 1: Tabla de comparación de Juegos

Juego	Uso de LLM	Ventajas	Desventajas	Ideas aplicables
AI Dungeon	Generación de narrativa en tiempo real	Experiencias personalizadas; Libertad narrativa	Respuestas incoherentes en contextos prolongados; Alto costo computacional	Integración de textos no lineales y adaptativos
Minecraft	Experimentación con LLM en NPCs	Autonomía y adaptabilidad de personajes	Implementación limitada; Enfoque experimental	Incorporar NPCs que reaccionen dinámicamente al entorno
The Sims	Árboles de comportamiento en NPCs	Gestión autónoma de necesidades y dinámicas sociales complejas	Limitación a reglas predefinidas; Interacciones predecibles	Gestión indirecta del jugador y personajes con prioridades dinámicas
ChatGPT Plugins	Uso de LLM para experiencias interactivas fuera de juegos	Interacciones altamente personalizables	Dependencia de contexto bien estructurado	Adaptación del sistema de interacción natural para tripulantes

A pesar de los desafíos técnicos y de diseño, la integración de LLM en videojuegos representa un avance significativo en la creación de experiencias más inmersivas y

dinámicas. Nuestro proyecto busca capitalizar estas capacidades, aplicándolas en un contexto de simulación espacial donde la interacción entre jugador y tripulación controlada por IA genere un entorno único y adaptable.

Como punto de partida y referencia en la investigación, se han revisado los Trabajos de Fin de Grado desarrollados en el curso 2023-2024 en el Departamento de Informática de la UCM [18], lo que ha permitido identificar enfoques y métodos afines para la integración de IA y sistemas interactivos en Unity.

2.3 Tecnologías

2.3.1 Modelos de lenguaje y APIs

Para nuestro proyecto requerimos de una manera de acceder a un LLM para el comportamiento de los NPCs. Para esto evaluamos diferentes API (Application Programming Interface), que son conjuntos de reglas que permiten que diferentes programas se comuniquen entre sí, para conectarnos a diferentes modelos del lenguaje. Finalmente llegamos a la conclusión de que la mejor opción era la API de Gemini. La principal razón fue su fácil integración con Unity, que es el motor que estamos utilizando, y en unas pruebas que realizamos conseguimos una buena conexión y respuestas válidas.

Inicialmente estuvimos considerando la API de OpenAI con GPT-3.5 Turbo [19], debido a que tiene mucha potencia y un precio asequible. Sin embargo, la descartamos porque el uso era más costoso que el de Gemini y porque la conexión con la API era más compleja y no logramos realizarla desde Unity.

Otra alternativa fue utilizar Hugging Face [20], que ofrece una amplia variedad de modelos de lenguaje abiertos. Su conexión con Unity fue rápida debido a la abundante documentación existente. Sin embargo, decidimos descartarla porque en nuestras pruebas las respuestas del LLM eran incoherentes con las peticiones (*prompts* -

entrada de texto que se proporciona a un modelo de IA para guiar la generación de una respuesta específica) enviadas, repetían información y no eran capaces de responder en el formato JSON que necesitábamos.

Tabla 2: Tabla de comparación de LLMs

Api de LLM	Ventajas	Desventajas	Decisión
Gemini	Fácil integración con Unity, conexión rápida y respuestas acordes al formato requerido	Menor popularidad y comunidad comparada con OpenAi	Seleccionada por facilidad de uso con Unity y por ser gratuita hasta cierto nivel de uso.
OpenAI	Modelos avanzados y gran precisión. Respuestas acordes al formato	Integración compleja con Unity, coste más elevado para la cantidad de prompts que requerimos	Descartada por complejidad y por su coste de uso
Hugging Face	Amplia variedad de modelos, comunidad activa	Respuestas poco fiables y repetitivas y no acordes al formato requerido	Descartada por mal rendimiento en pruebas

2.3.2 Motor de videojuegos

Para desarrollar un videojuego lo más cómodo y conveniente es apoyarse en un motor de videojuegos. Un motor de videojuego (*Game Engine)* es una herramienta que proporciona librerías y motores de vídeo y audio para facilitar el desarrollo de videojuegos. Nosotros decidimos utilizar Unity. Unity es una herramienta con la que ya teníamos algo de familiaridad, es rápida de aprender y tiene una fácil conexión con LLMs. Además, su entorno es muy adecuado para juegos en 2D, que es el formato de nuestro proyecto.

También consideramos utilizar Unreal Engine [21], un motor de alto rendimiento gráfico especialmente potente para proyectos 3D. Sin embargo, sus capacidades eran excesivas para un videojuego 2D, y su curva de aprendizaje más elevada habría ralentizado el desarrollo.

Por último, evaluamos Game Maker [22], un motor caracterizado por su simpleza y curva de aprendizaje muy accesible. A pesar de su facilidad, su enfoque en programación mediante *drag and drop* limita nuestras posibilidades y no encontramos forma de integrar un LLM en este motor.

Tabla 3: Tabla de comparación de motores de videojuegos

Motor de juego	Ventajas	Desventajas	Decisión
Unity	Familiaridad, buena integración con LLMs, ideal para 2D, gratuito para proyectos no comerciales	Gráficos menos avanzados que Unreal.	Seleccionado debido a facilidad y compatibilidad con 2D

Unreal Engine	Potente motor gráfico, excelente para juegos 3D, gratuito para proyectos no comerciales	Excesivo para juego 2D, alto consumo de recursos	Descartado por ser innecesario para 2D
Game Maker	Simple y muy buena curva de aprendizaje , totalmente gratuito	Límite en las posibilidades debido al tipo de programación y incompatibilidad con la conexión al LLM	Descartado debido a no poder hacer uso del LLM elemento crucial en nuestro proyecto

3 Sistema de Progresión y Mecánicas de Juego

3.1 Requisitos funcionales

Esta sección detalla los requisitos funcionales que guían la interacción del jugador con el sistema de juego.

Etiqueta: TFGSIMS-72

Nombre: Fin del juego

Descripción: Como jugador quiero que el juego tenga un final al que se pueda llegar en todas las partidas.

Criterios aceptación:

1. Antes de completar la última misión de la lista de misiones del juego se dará una advertencia al usuario de que esto terminará la partida.
2. Al completar la última misión se notificará al jugador del final del juego.
3. Cuando finalice el juego se mostrará una pantalla de final con opción de volver a la pantalla principal.
4. Al volver a la pantalla principal se archivarán los datos de la partida en un historial de partidas y ya no será accesible para jugar.

Etiqueta: TFGSIMS-71

Nombre: Completar misión

Descripción: Como jugador quiero que al conseguir el/los objetivos marcados por una misión se complete.

Criterios aceptación:

1. Al conseguir el/los objetivos marcado por una misión ésta será completada y se dará una/varias nuevas al jugador.

Etiqueta: TFGSIMS-70

Nombre: Recibir misión

Descripción: Como jugador quiero poder recibir misiones a lo largo de la partida que marquen objetivos dentro del juego.

Criterios aceptación:

1. Al iniciar partida y al completar una misión el jugador recibirá una misión en forma de notificación que consistirá en uno o varios objetivos dentro del juego.
2. El jugador contará con al menos una misión en todo momento de la partida que le indique el camino a seguir para completar el juego.

Etiqueta: TFGSIMS-69

Nombre: Mejorar nave

Descripción: Como jugador, quiero poder mejorar la nave, viajar a mejores planetas y avanzar en la exploración del espacio.

Criterios aceptación:

1. Cuando el jugador quiere mejorar la nave si tiene los materiales necesarios se mejora la nave.
2. Cuando el jugador quiere mejorar la nave si no tiene los materiales necesarios se informará que faltan materiales.

Etiqueta: TFGSIMS-68

Nombre: Cargar partida

Descripción: Como jugador, quiero poder cargar una partida guardada previamente para continuar desde el punto exacto donde la dejé, manteniendo el progreso.

Criterios aceptación:

1. Cuando el usuario carga la partida si tiene una partida guardada se cargarán los datos desde la base de datos y se abrirá la partida.
2. Cuando el usuario carga la partida si no tiene una partida guardada se mostrará un error de que no se puede cargar partida.

Etiqueta: TFGSIMS-67

Nombre: Guardar partida

Descripción: Como jugador, quiero poder guardar el progreso de la partida para poder continuar en otro momento desde el mismo punto en el que dejé, sin perder el avance alcanzado.

Criterios aceptación:

1. Cuando el usuario guarde la partida se mandarán todos los datos a almacenamiento en base de datos y se saldrá de la partida.
2. Los datos a guardar serán: estado actual de cada tripulante, inventario de la nave, inventario de cada instalación, recursos desbloqueados, nivel de la nave, planeta actual, estado de las misiones.

Etiqueta: TFGSIMS-66

Nombre: Eliminar partida

Descripción: Como jugador, quiero poder eliminar una partida para empezar de nuevo o eliminar datos guardados.

Criterios aceptación:

1. Cuando el jugador elimine la partida todos los datos de la partida se borrarán, siendo: estado actual de cada tripulante, inventario de la nave, inventario de cada instalación, recursos desbloqueados, nivel de la nave, planeta actual, estado de las misiones, nombre del jugador y nombre de la partida.

Etiqueta: TFGSIMS-65

Nombre: Rellenar combustible

Descripción: Como jugador, quiero que los tripulantes puedan rellenar el combustible de la nave para asegurar que la nave pueda continuar explorando planetas.

Criterios aceptación:

1. Cuando un tripulante quiera rellenar el combustible si tiene en su inventario y el tanque no está lleno llenara una parte proporcional al combustible usado.
2. Cuando un tripulante quiera rellenar el combustible si tiene en su inventario y el tanque está lleno no podrá hacerlo y se notificará al usuario.
3. Cuando un tripulante quiera rellenar el combustible si no tiene en su inventario se notificará del error al usuario.

Etiqueta: TFGSIMS-63

Nombre: Mostrar interfaz de cocina

Descripción: Como jugador, quiero poder acceder y visualizar la interfaz de cocina para ver la cantidad de comida que queda disponible.

Criterios aceptación:

1. Cuando el jugador interactúe con la instalación de cocina se le mostrara una interfaz que muestra un número con la cantidad de comida que queda disponible.

Etiqueta: TFGSIMS-62

Nombre: Comer

Descripción: Como jugador, quiero que los tripulantes puedan comer para satisfacer su necesidad de comida, lo que les permitirá mantener su bienestar.

Criterios aceptación:

1. Cuando un tripulante va a comer si la cocina tiene comida el tripulante aumentará su nivel de satisfacción de hambre y se reducirá la cantidad de comida en la cocina.
2. Cuando un tripulante va a comer si la cocina no tiene comida se notificará que no queda comida.

Etiqueta: TFGSIMS-61

Nombre: Mostrar interfaz de laboratorio

Descripción: Como jugador, quiero poder acceder y visualizar la interfaz de laboratorio para ver el avance en la investigación actual y todas las investigaciones anteriores.

Criterios aceptación:

1. Cuando el jugador interactúe con la instalación de laboratorio se le mostrara una interfaz con el avance de la investigación actual y su nombre así como las investigaciones ya completadas con su correspondiente recurso desbloqueado y sus nombres.

Etiqueta: TFGSIMS-59

Nombre: Completar investigación

Descripción: Como jugador, quiero que los tripulantes puedan completar una investigación para que se obtengan los avances correspondientes.

Criterios aceptación:

1. Cuando el tiempo necesario para completar la investigación (tiempo que es especifico de cada una) se cumpla se desbloquearán los recursos correspondientes y la posibilidad de empezar las siguiente investigación.

Etiqueta: TFGSIMS-58

Nombre: Reanudar investigación

Descripción: Como jugador, quiero que los tripulantes puedan reanudar una investigación pausada para que la nave continúe avanzando en las investigaciones una vez que los tripulantes hayan completado otras tareas prioritarias.

Criterios aceptación:

1. Cuando un tripulante interactúa con la instalación de investigación si hay una pausada esta se reanudará.

Etiqueta: TFGSIMS-57

Nombre: Pausar investigación

Descripción: Como jugador, quiero que los tripulantes puedan pausar una investigación para que puedan gestionar su tiempo y realizar otras tareas, sin que se pierda el progreso de la investigación.

Criterios aceptación:

1. Cuando un tripulante se salga de la instalación de investigación si había una investigación en progreso se pausará la investigación.
2. Cuando un tripulante se salga de la instalación de investigación si no había una investigación en progreso no pasará nada.

Etiqueta: TFGSIMS-56

Nombre: Iniciar investigación

Descripción: Como jugador, quiero que los tripulantes realicen investigaciones para descubrir recursos para poder mejorar la nave.

Criterios aceptación:

1. Cuando un tripulante quiera iniciar una investigación si no hay ninguna investigación activa iniciará la siguiente investigación.
2. Cuando un tripulante quiera iniciar una investigación si hay una investigación activa se notificará que solo se puede realizar una investigación a la vez.

Etiqueta: TFGSIMS-55

Nombre: Mostrar interfaz de taller

Descripción: Como jugador, quiero poder acceder y visualizar la interfaz de taller para ver los materiales disponibles a fabricar.

Criterios aceptación:

1. Cuando el jugador interactúe con la instalación de taller se le mostrara una interfaz con los posibles materiales a fabricar y los recursos necesarios para fabricarlos, sus costes.

Etiqueta: TFGSIMS-54

Nombre: Fabricar material

Descripción: Como jugador, quiero que los tripulantes puedan fabricar materiales utilizando los recursos disponibles en la nave, para poder crear mejoras de la nave.

Criterios aceptación:

1. Cuando un tripulante intente fabricar un material si se tienen los recursos necesarios se añadirá a su inventario y se quitarán de su inventario los utilizados.
2. Cuando un tripulante intente fabricar un material si no se tienen los recursos necesarios se notificará del error.

Etiqueta: TFGSIMS-53

Nombre: Mostrar interfaz de almacén

Descripción: Como jugador, quiero poder acceder y visualizar la interfaz de almacén para ver los recursos disponibles en la nave.

Criterios aceptación:

1. Cuando el jugador interactúe con la instalación de almacén se le mostrara una interfaz con las cantidades de cada objeto que hay en la nave y su nombre.

Etiqueta: TFGSIMS-52

Nombre: Mover recursos

Descripción: Como jugador, quiero que los tripulantes puedan mover recursos entre la nave y las instalaciones para mantener el buen funcionamiento de la nave.

Criterios aceptación:

1. Cuando un tripulante necesite mover recursos a una instalación si hay recursos suficientes en la nave los sacará de la nave y los meterá en la instalación.
2. Cuando un tripulante necesite mover recursos a una instalación si no hay recursos suficientes en la nave se notificará que faltan recursos para hacer dicha acción.

Etiqueta: TFGSIMS-51

Nombre: Mostrar interfaz de navegación

Descripción: Como jugador, quiero que se muestre la interfaz de navegación de la nave para poder seleccionar destinos.

Criterios aceptación:

1. Cuando el jugador interactúe con la instalación de navegación se abrirá una interfaz donde se mostrarán los planetas a los que puede viajar así como el coste en combustible de viajar a estos.

Etiqueta: TFGSIMS-50

Nombre: Viajar a nuevo planeta

Descripción: Como jugador, quiero poder viajar a un nuevo planeta para obtener nuevos recursos.

Criterios aceptación:

1. Cuando el jugador intente viajar a un nuevo planeta si un planeta ha sido seleccionado y tienes combustible suficiente, la nave irá al planeta y se reducirá el combustible adecuado.

2. Cuando el jugador intente viajar a un nuevo planeta si un planeta ha sido seleccionado pero no tienes combustible suficiente se notificará al usuario que falta combustible y se cancelará el viaje.

3. Cuando el jugador intente viajar a un nuevo planeta si un planeta no ha sido seleccionado se notificará al usuario que debe escoger planeta destino.

Etiqueta: TFGSIMS-48

Nombre: Modificar nivel de combustible

Descripción: Como jugador, quiero que los tripulantes puedan modificar el nivel de combustible de la nave para asegurar que se tenga suficiente combustible para desplazarse, optimizando el uso de recursos disponibles.

Criterios aceptación:

1. El nivel de combustible será visible en la interfaz base de la nave.

2. Cuando el tripulante intente rellenar el tanque si el tanque no está lleno se aumentará el nivel de combustible.

3. Cuando el tripulante intente rellenar el tanque si el tanque está lleno se notificará que no se puede rellenar un tanque completo.

4. Cuando se efectúa un viaje a un planeta se reducirá el nivel de combustible dependiendo del planeta al que se viaje.

Etiqueta: TFGSIMS-47

Nombre: Modificar inventario de nave

Descripción: Como jugador, quiero que los tripulantes puedan modificar el inventario de la nave para gestionar los recursos disponibles.

Criterios aceptación:

1. Cuando un tripulante quiera sacar un objeto de un inventario si hay suficientes se sumarán al inventario del tripulante y se quitarán de la nave.

2. Cuando con una recompensa de evento se obtienen materiales se meterán los recursos a la nave.

Etiqueta: TFGSIMS-44

Nombre: Gestionar evento

Descripción: Como sistema, debo gestionar un evento aleatorio para determinar si el jugador lo supera o no.

Criterios aceptación:

1. Cuando el jugador mande su respuesta al evento aleatorio si el LLM devuelve que el evento ha sido superado se notificará al usuario y se aplicarán los beneficios de superarlo.

2. Cuando el jugador mande su respuesta al evento aleatorio si el LLM devuelve que el evento no ha sido superado se notificará al usuario y se aplicarán los efectos por no superarlo.

Etiqueta: TFGSIMS-40

Nombre: Generar evento aleatorio

Descripción: Como sistema, debo generar eventos aleatorios para introducir desafíos manteniendo la experiencia del jugador dinámica y emocionante.

Criterios aceptación:

1. Periódicamente hay una probabilidad de generar un evento aleatorio.
2. Cuando se genera un evento se manda un mensaje al LLM para obtener la descripción del evento que se mostrará al usuario junto a un campo para responder al evento.

Etiqueta: TFGSIMS-39

Nombre: Terminar expedición

Descripción: Como jugador, quiero poder terminar una expedición para que tripulantes vuelvan a sus puestos y continuar gestionando la nave con los recursos adquiridos.

Criterios aceptación:

1. Cuando intentes terminar una expedición se otorgarán las recompensas de esta dependiendo del tiempo transcurrido.
2. Cuando termines la expedición habrá una posibilidad de que algunos de tus tripulantes no puedan volver que aumentará con la duración de la exploración.

Etiqueta: TFGSIMS-38

Nombre: Convencer a tripulante de formar parte en expedición

Descripción: Como jugador, quiero poder convencer a un tripulante de formar parte en una expedición cuando inicialmente se niega, de manera que pueda completar el equipo necesario para la misión y maximizar las posibilidades de éxito.

Criterios aceptación:

1. Cuando un tripulante se niegue si el mensaje enviado al tripulante este lo considera convincente (decisión del LLM) el tripulante accederá y se unirá a la expedición.
2. Cuando un tripulante se niegue si el mensaje enviado al tripulante este no lo considera convincente (decisión del LLM) la expedición se iniciará sin el tripulante.

Etiqueta: TFGSIMS-37

Nombre: Seleccionar exploradores

Descripción: Como jugador, quiero poder seleccionar exploradores para una expedición de manera que pueda asignar a los tripulantes más adecuados según las habilidades requeridas y las condiciones de la misión.

Criterios aceptación:

1. EL jugador si tiene tripulantes en su partida podrá seleccionar exploradores que dejarán de hacer sus tareas mientras están asignados como exploradores y hasta el final de la expedición.

Etiqueta: TFGSIMS-36

Nombre: Iniciar expedición

Descripción: Como jugador, quiero poder iniciar una expedición para enviar a mi tripulación a explorar planetas y recolectar recursos.

Criterios aceptación:

1. Cuando el jugador inicie una expedición si la nave se encuentra en un planeta que aún no ha sido explorado y algún tripulante ha sido seleccionado y ninguno de ellos se opone a ir se iniciará la expedición.

2. Cuando el jugador inicie una expedición si la nave se encuentra en un planeta que aún no ha sido explorado y algún tripulante ha sido seleccionado, se consultará con el LLM de cada tripulante seleccionado si éste quiere asistir a la expedición o no, si alguno se opone se iniciará una conversación con ese tripulante.

3. Cuando el jugador inicie una expedición si la nave se encuentra en un planeta que aún no ha sido explorado si ningún tripulante ha sido seleccionado se mandará un mensaje al usuario que debe seleccionar algún tripulante.

4. Cuando el jugador inicie una expedición si la nave se encuentra en un planeta que ya ha sido explorado se notificará al usuario que dicho planeta ya ha sido explorado.

5. Cuando el jugador inicie una expedición si la nave no se encuentra en un planeta se notificará al usuario que debe de estar en un planeta para iniciar una expedición.Principio del formulario

---

Etiqueta: TFGSIMS-34

Nombre: Explorar planeta

Descripción: Como jugador, quiero poder explorar un planeta para descubrir nuevos recursos y oportunidades para la nave.

Criterios aceptación:

1. Cuando un explorador llega a un nuevo planeta si tiene suficiente equipamiento comenzará la exploración automáticamente.

2. Si durante la exploración se encuentra un recurso, este se añadirá al inventario del explorador.

3. Si el explorador encuentra un obstáculo, se notificará al usuario con las opciones disponibles para resolverlo.

Etiqueta: TFGSIMS-32

Nombre: Regresar a la nave

Descripción: Como jugador, quiero que los exploradores puedan regresar a la nave una vez que hayan completado sus tareas en un planeta.

Criterios aceptación:

1. Cuando los exploradores terminan la exploración de un planeta, pueden regresar a la nave automáticamente si hay combustible suficiente.
2. Si no hay combustible suficiente, se notificará al usuario para tomar las medidas necesarias.
3. Al regresar, los recursos recolectados se trasladarán automáticamente al inventario de la nave.

Etiqueta: TFGSIMS-29

Nombre: Reasignar tripulantes

Descripción: Como jugador, quiero poder reasignar tripulantes a diferentes tareas dentro de la nave según las necesidades actuales.

Criterios aceptación:

1. Cuando el usuario selecciona un tripulante, podrá asignarlo a una nueva tarea disponible en la nave.

2. Si la tarea requiere habilidades específicas, se verificará si el tripulante cumple con ellas antes de reasignarlo.

Etiqueta: TFGSIMS-28

Nombre: Reparar nave

Descripción: Como jugador, quiero que los tripulantes puedan reparar la nave cuando sufra daños para garantizar su funcionalidad continua.

Criterios aceptación:

1. Cuando un tripulante inicia reparaciones, se consumirán los materiales necesarios del inventario de la nave.
2. Si no hay materiales suficientes, se notificará al usuario para que los recolecte o fabrique.
3. Al finalizar la reparación, se restaurará el estado funcional de la nave.

Etiqueta: TFGSIMS-27

Nombre: Supervisar estado de la nave

Descripción: Como jugador, quiero poder supervisar el estado general de la nave para tomar decisiones informadas sobre su gestión.

Criterios aceptación:

1. La interfaz principal mostrará el estado actual de los sistemas de la nave, como combustible, integridad estructural y recursos disponibles.
2. Si un sistema está en estado crítico, se notificará al usuario con recomendaciones de acción.

Etiqueta: TFGSIMS-26

Nombre: Gestionar energía de la nave

Descripción: Como jugador, quiero poder gestionar el uso de la energía de la nave para optimizar su funcionamiento.

Criterios aceptación:

1. Cuando la energía disponible disminuya por debajo de un umbral crítico, se notificará al usuario.
2. El usuario podrá redistribuir la energía entre los sistemas de la nave según sus prioridades.

Etiqueta: TFGSIMS-25

Nombre: Almacenar recursos

Descripción: Como jugador, quiero que los recursos recolectados se almacenen correctamente en la nave para utilizarlos más adelante.

Criterios aceptación:

1. Cuando un tripulante entrega recursos en el almacén, estos se registrarán en el inventario de la nave.
2. Si el almacén está lleno, se notificará al usuario para que amplíe su capacidad o utilice los recursos existentes.

Etiqueta: TFGSIMS-23

Nombre: Distribuir recursos

Descripción: Como jugador, quiero poder distribuir recursos almacenados en la nave a las distintas instalaciones según las necesidades.

Criterios aceptación:

1. El usuario podrá seleccionar los recursos a distribuir y la instalación de destino.
2. Si la instalación no tiene capacidad para recibir los recursos, se notificará al usuario.

Etiqueta: TFGSIMS-22

Nombre: Supervisar tripulación

Descripción: Como jugador, quiero poder supervisar el estado de la tripulación para asegurar su bienestar y productividad.

Criterios aceptación:

1. La interfaz de tripulación mostrará el estado de salud, ánimo y habilidades de cada tripulante.
2. Si algún tripulante está en estado crítico, se notificará al usuario para que tome medidas.

Etiqueta: TFGSIMS-21

Nombre: Entrenar tripulación

Descripción: Como jugador, quiero poder entrenar a la tripulación para mejorar sus habilidades y desempeño en tareas específicas.

Criterios aceptación:

1. Cuando el usuario asigna un tripulante a una sesión de entrenamiento, este mejorará en la habilidad seleccionada.

2. Si no hay recursos suficientes para entrenar, se notificará al usuario.

Etiqueta: TFGSIMS-20

Nombre: Planificar misiones

Descripción: Como jugador, quiero poder planificar misiones para que mi tripulación cumpla objetivos específicos.

Criterios aceptación:

1. El usuario podrá seleccionar los objetivos de la misión y los tripulantes asignados.
2. Si los objetivos no son alcanzables con los recursos disponibles, se notificará al usuario para ajustar la planificación.

Etiqueta: TFGSIMS-19

Nombre: Evaluar riesgos de misión

Descripción: Como jugador, quiero poder evaluar los riesgos asociados a cada misión para tomar decisiones informadas.

Criterios aceptación:

1. Antes de iniciar una misión, se mostrará un informe detallado de riesgos basado en las condiciones actuales.
2. Si los riesgos son demasiado altos, se recomendará al usuario posponer o cancelar la misión.

Etiqueta: TFGSIMS-17

Nombre: Analizar entorno

Descripción: Como jugador, quiero que la nave pueda analizar el entorno para identificar recursos y peligros potenciales.

Criterios aceptación:

1. Cuando se active el análisis del entorno, se generará un informe detallado con los hallazgos.
2. Si el análisis requiere energía adicional, se notificará al usuario.

Etiqueta: TFGSIMS-16

Nombre: Mejorar sistemas de navegación

Descripción: Como jugador, quiero poder mejorar los sistemas de navegación de la nave para alcanzar destinos más lejanos y con mayor precisión.

Criterios aceptación:

1. Cuando se mejora el sistema de navegación, se ampliará el rango de exploración disponible.
2. Si no hay suficientes recursos para la mejora, se notificará al usuario.

Etiqueta: TFGSIMS-12

Nombre: Mantener comunicación con tripulación

Descripción: Como jugador, quiero poder comunicarme con la tripulación en todo momento para dar órdenes y recibir informes.

Criterios aceptación:

1. La interfaz de comunicación permitirá enviar y recibir mensajes entre el usuario y los tripulantes.

2. Si la comunicación falla, se notificará al usuario con opciones para restablecerla.

Etiqueta: TFGSIMS-10

Nombre: Gestionar inventario

Descripción: Como jugador, quiero poder gestionar el inventario de la nave para mantener un control sobre los recursos disponibles.

Criterios aceptación:

1. El usuario podrá añadir, retirar o reordenar elementos en el inventario según sea necesario.

2. Si el inventario alcanza su capacidad máxima, se notificará al usuario para que tome medidas.

Etiqueta: TFGSIMS-8

Nombre: Implementar medidas de seguridad

Descripción: Como jugador, quiero que la nave implemente medidas de seguridad para proteger a la tripulación y los recursos.

Criterios aceptación:

1. Cuando se activa un protocolo de seguridad, se restringirá el acceso a áreas sensibles de la nave.

2. Si se detecta una amenaza, se notificará al usuario con las acciones recomendadas.

3.2 Progresión Tecnológica y Recursos

3.2.1 Mejora de la nave e Investigaciones

El progreso en el juego está marcado por el nivel de la nave, iniciando en nivel 1 y con el máximo siendo 5. La mejora de la nave requerirá de una lista de recursos y materiales obtenibles sólo en el nivel actual.

Recursos son los elementos básicos recolectables directamente en los planetas durante las expediciones, mientras que materiales son los productos de estos recursos una vez procesados o combinados.

Cada nivel consta de una serie de recursos disponibles en los planetas explorables, así como investigaciones propias de cada nivel que tras investigar una cierta cantidad de tiempo desbloquean nuevos materiales que son clave para alcanzar el siguiente nivel.

- **Nave Nivel 1**
  - Requiere: 5 de Carbonite y 5 de Timberite.
  - **Desbloqueos:**
    - 12 horas ingame: **Carbonite** (Carbón + Piedra).
    - 14 horas ingame: **Timberite** (Carbonite + Madera).
- **Nave Nivel 2**
  - Requiere: 5 de Ferronite y 5 de Timbersteel.
  - **Desbloqueos:**
    - 14 horas ingame: **Ferronite** (Carbonite + Hierro).
    - 16 horas ingame: **Timbersteel** (Timberite + Hierro).
- **Nave Nivel 3**
  - Requiere: 5 de Cable y 5 de Circuito.
  - **Desbloqueos:**
    - 16 horas ingame: **Cable** (Ferronite + Cobre).
    - 18 horas ingame: **Circuit** (Timbersteel + Cobre).
- **Nave Nivel 4**

- Requiere: 3 de Super Computadora.
- **Desbloqueos:**
  - 18 horas ingame: **Power Cell** (Cable + Oro).
  - 20 horas ingame: **Display** (Circuit + Oro).
  - 24 horas ingame: **Super Computer** (Cable + Circuit + Power Cell + Display).

3.2.2 Recursos Disponibles por Nivel de Nave

Los recursos disponibles en los planetas explorados dependen del nivel de la nave. A medida que se mejora la nave, se accede a nuevos recursos, manteniendo siempre disponibles los de niveles anteriores.

- **Nivel 1**: Carbón, Piedra, Madera.
- **Nivel 2**: Hierro.
- **Nivel 3**: Cobre.
- **Nivel 4**: Oro.

4 Diagramas de diseño

4.1 Diagrama de clases

En esta sección se presenta el diagrama de clases del proyecto. Un diagrama de clases es una representación estática del sistema que muestra las clases que lo componen, sus atributos, métodos y las relaciones entre ellas.

4.1.1 Contexto

El diagrama de clases refleja los componentes principales del videojuego:

- La **Nave** y sus **Instalaciones** (cocina, laboratorio, motor, navegación, etc.)

- Los **Tripulantes**, que son instancias controladas por IA mediante un LLM.
- El **Jugador**, que interactúa indirectamente gestionando los recursos y asignando tareas.
- Los **Materiales** y **Recursos**, que forman parte de la mecánica de progresión.

Cada clase contiene atributos y métodos relevantes que permiten modelar el comportamiento autónomo de los NPCs y la evolución de la nave.

4.1.2 Diagrama de clases

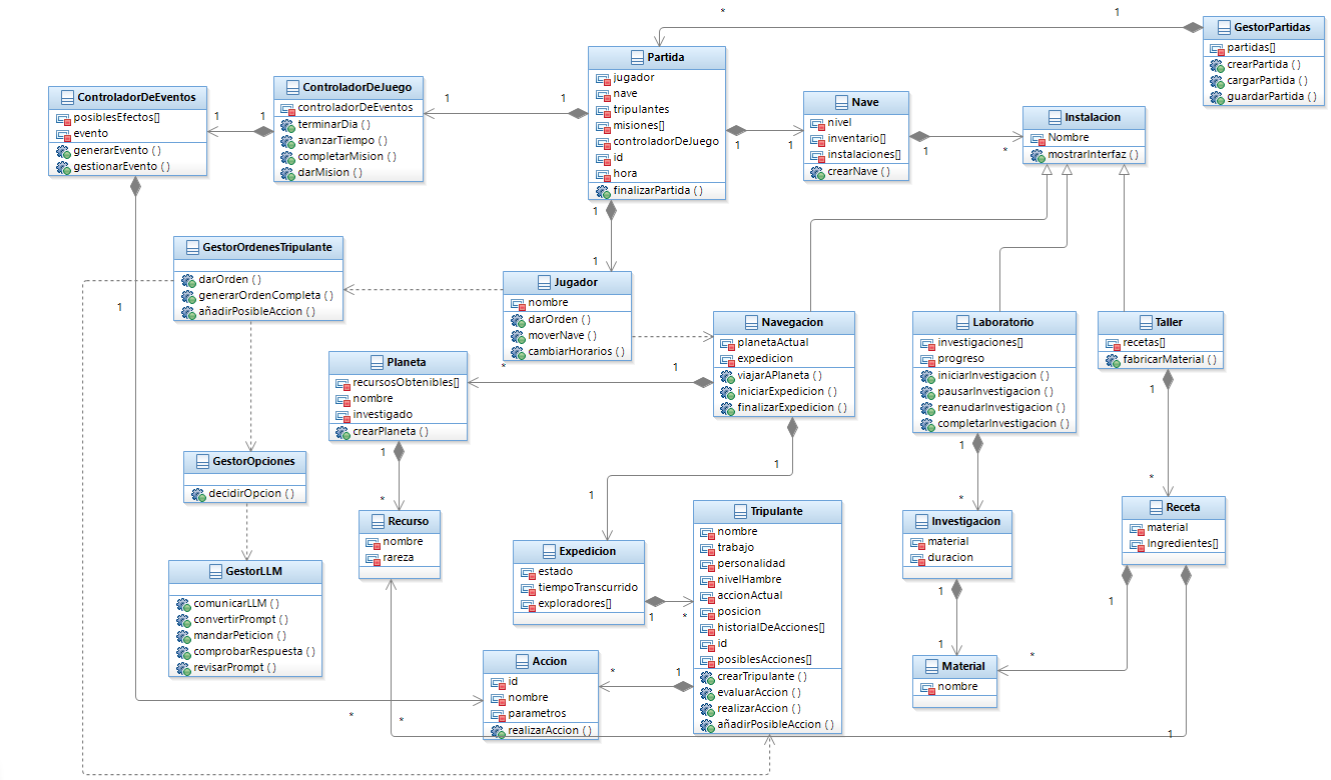

Ilustración 5: Diagrama de Clases

4.1.3 Descripción de las clases principales

Partida

- **Responsabilidad**: Representa el estado global del juego, agrupando la información de la nave, los tripulantes, el jugador y el controlador de eventos.

Nave

- **Responsabilidad**: Modela la nave espacial que el jugador gestiona. Incluye su nivel de progreso, el inventario de recursos y las instalaciones internas como cocina, taller, laboratorio o navegación.

Tripulante

- **Responsabilidad**: Representa a los miembros de la tripulación. Cada tripulante tiene características como hambre, ánimo, personalidad y un conjunto de acciones posibles que realiza de manera autónoma, en coordinación con el LLM.

Jugador

- **Responsabilidad**: Representa al jugador humano que interactúa de manera indirecta, dando órdenes y moviendo la nave.

Instalación (y subclases: **Laboratorio**, **Taller**, **Navegación**)

- **Responsabilidad**: Definen los distintos módulos de la nave. Cada instalación ofrece funcionalidades específicas: investigaciones científicas, fabricación de materiales, gestión de expediciones, entre otros.

Gestor LLM

- **Responsabilidad**: Facilita la comunicación con el modelo de lenguaje utilizado para simular las decisiones y comportamientos de los NPCs.

4.2 Diagramas de secuencia

Los diagramas de secuencia modelan el flujo de mensajes y la interacción entre objetos o componentes del sistema. En el proyecto los hemos utilizado para representar algunas operaciones críticas como todas las relacionadas con la conexión con el LLM y con la toma de decisiones por parte de los tripulantes.

4.2.1 Diagrama de conexión con el LLM

En este diagrama se representa el flujo principal utilizado en el proyecto para conectarse a la API de Gemini.

Pasos principales:

1. Convertir el prompt recibido al formato requerido por la API
2. Mandar y recibir respuesta de la API
3. Comprobar si la respuesta de la API es correcta y si no repetir proceso un número determinado de veces.

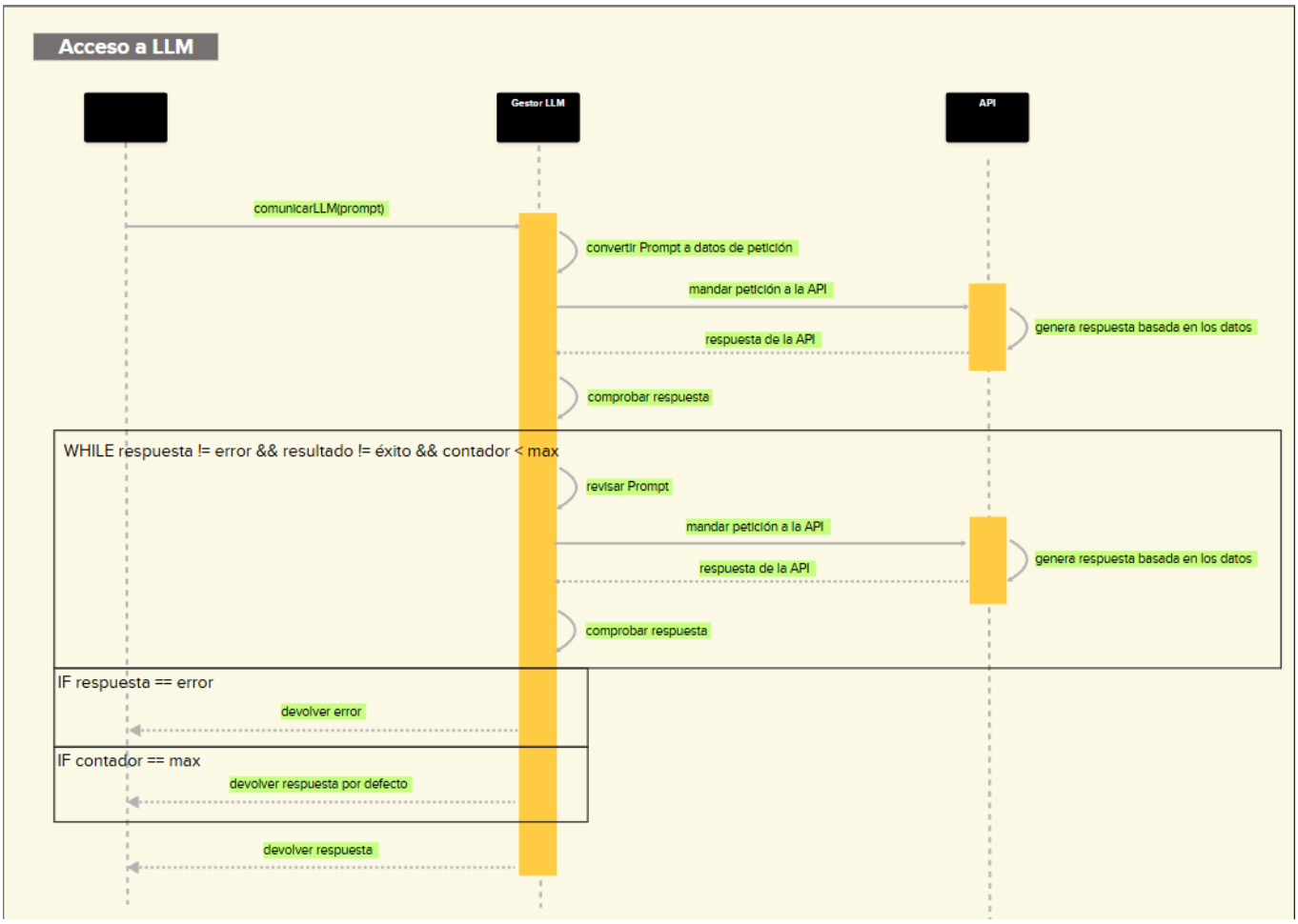

Ilustración 6: Diagrama de conexión LLM

4.2.2 Diagrama de toma de decisiones de tripulante

En este diagrama se representa la toma de decisiones de los tripulantes mediante opciones implementada en el proyecto.

Pasos principales:

1. Genera el contexto de la nave y el tripulante.
2. Crea el prompt con ese contexto y las opciones posibles.

3. Manda el prompt al LLM y devuelve la respuesta.

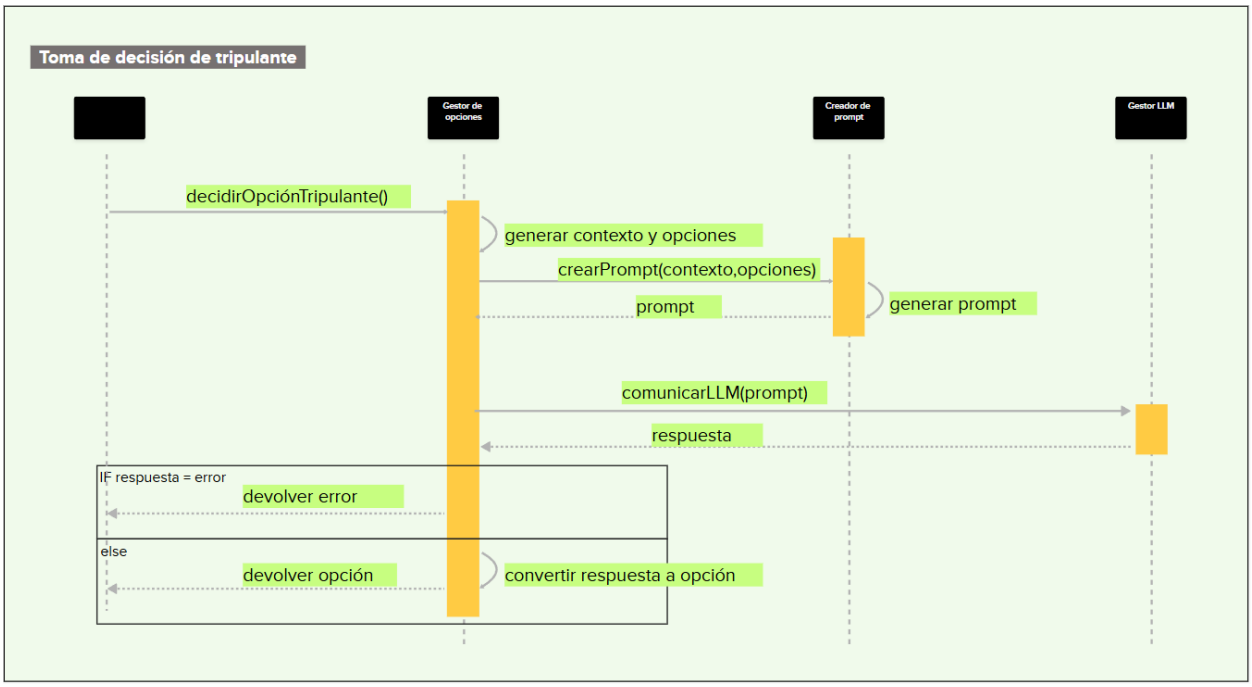

Ilustración 7: Diagrama de toma de decisiones del tripulante

5 Implementación

5.1 Script inicial de comunicación con LLM

La primera integración con el LLM se hizo con un guion de código (script) básico utilizando el sistema de solicitudes HTTP de Unity para comunicarse con la API de Gemini. El objetivo de esta fase inicial no era conseguir una interacción compleja, sino comprobar que la comunicación con el LLM era correcta y daba respuestas con el formato que se le pedía y dentro de las posibles respuestas.

En este primer script, el mensaje era muy simple y se limitaba a indicar unas posibles acciones y se le pedía que escogiera una. No tenía contexto, reglas ni una estructura trabajada. La respuesta consistía en una sola palabra correspondiente a la acción seleccionada.

Funcionó correctamente desde el inicio y permitió validar la conexión entre el juego y el modelo. La simplicidad fue deliberada ya que el objetivo era solo centrarse en la conexión técnica antes de centrarse en el prompt complejo y el manejo de las respuestas.

5.2 Estética, sprites y TileMap

Para representar visualmente los entornos del juego se utilizó el sistema mapa de mosaicos (TileMap) de Unity, que permite diseñar niveles 2D a partir de mosaicos reutilizables. Se trabajó sobre una cuadrícula con celdas de 1 unidad para facilitar la alineación con los elementos de navegación y detección de clics por parte del jugador.

La estética general del juego se inspiró en un estilo pixel-art retrofuturista en un formato de 32x32 pixeles, con referencias visuales similares a títulos como *Starbound* [23] o *RimWorld* [24]. Se priorizó la claridad visual de los elementos interactivos (instalaciones e interfaces) y la coherencia entre elementos de la nave.

Las imágenes (sprites) fueron seleccionadas y adaptadas desde bibliotecas de recursos libres y posteriormente organizadas y modificadas en un conjunto de mosaicos (Tileset), cambiando los existentes y añadiendo nuevos recursos, luego integrados dentro del Editor de Unity. Esto facilitó la creación modular de habitaciones e instalaciones. Se emplearon capas de mapa de mosaicos separadas para suelos, paredes y decoración, con sus respectivos elementos de colisión para delimitar el espacio caminable del tripulante.

5.3 Creación de la nave

La nave representa el entorno principal de interacción del jugador con los tripulantes y de los tripulantes con el sistema de juego. Su diseño fue planteado como un conjunto de habitaciones conectadas, cada una correspondiente a una instalación (laboratorio, cocina, navegación, etc.), con funciones diferenciables y específicas (). Para definir el espacio caminable se utilizó el sistema de colisión del mapa de mosaicos (Tilemap Collider) además de los límites de la herramienta de navegación (NavMesh) sobre la capa de suelos, bloqueando las paredes mediante colisiones para restringir el movimiento de los tripulantes a las zonas deseadas.

Se estableció una zona central de aparición para los tripulantes al comenzar la partida. Las habitaciones cuentan con zonas de paso libres y amplias que permiten el flujo entre módulos de varios tripulantes. Se integró un sistema de capas de organización en los objetos para asegurar la correcta superposición de elementos y la correcta interacción y reconocimiento de las entidades.

También se trabajó la ambientación mediante efectos visuales del fondo para simular el movimiento de la nave, mejorando la inmersión.

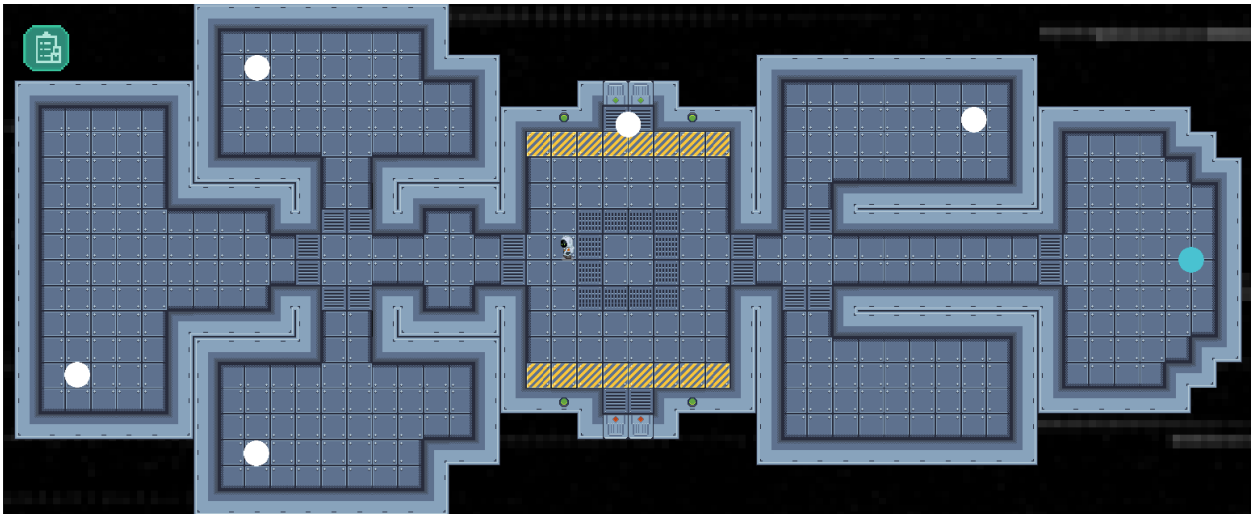

Ilustración 8: Diseño de la nave

5.4 Creación del tripulante

El tripulante es el NPC principal del juego, representando a uno de los miembros de la tripulación controlados por el LLM. Su imagen se compone de dos animaciones básicas: parado y caminar. Las animaciones se controlan mediante un controlador de animaciones (Animator Controller), gobernado por parámetros de dirección y estado gestionados por su script de comportamiento.

El objeto Tripulante es una entidad con componente de física 2D, un componente de cuerpo rígido 2D para su movimiento y un elemento de colisión 2D para las interacciones y un guion (script) que define su comportamiento. El desplazamiento se realiza por interpolación hacia posiciones discretas alineadas con la cuadrícula del mapa de mosaicos , permitiendo una navegación natural en la cuadrícula.

Además, se le asigna una personalidad inicial aleatoria que condiciona sus respuestas y acciones sugeridas por el LLM, y un nombre único aleatorio. Estos datos se definen en una clase Tripulante que contiene también variables como hambre y tareas asignadas.

Cada instancia de tripulante cuenta con una referencia directa al módulo de comunicación con el LLM permitiendo consultas constantes y concurrentes respecto a

su comportamiento, siendo capaz de guardar la información recibida en estructuras internas como la lista de órdenes pendientes, que consiste en una cola de acciones registradas como conversión de respuestas del LLM a acciones de juego.

Para gestionar la prioridad de colisiones de los clics del jugador sobre el tripulante o las instalaciones se implementó un sistema de gestión de capas de entidades para dar preferencia a los tripulantes por encima de las instalaciones y a su vez por encima del resto de elementos, activando solo las funcionalidades del más prioritario.

Esta estructura de tripulante fue exportada y convertida en un elemento prefabricado para facilitar la creación repetida y mediante petición de múltiples instancias de tripulante posteriormente.

5.5 Creación de instalaciones

Durante las primeras fases del desarrollo se implementaron las instalaciones de la nave: la cocina, el motor, el laboratorio y el taller. Estas fueron creadas inicialmente como objetos simples situados en el escenario para marcar una posición donde mover a los tripulantes para que realizaran las acciones relacionadas (por ejemplo, comer, recargar combustible o investigar). No obstante, se programó desde el inicio con la arquitectura orientada a objetos y herencia mostrada previamente en el diagrama de clase. Esto permitió establecer una infraestructura que haría más fácil la incorporación más adelante de la funcionalidad.

5.6 Automatización de movimiento

Para gestionar el desplazamiento de los tripulantes por la nave necesitábamos que fueran capaces tanto de esquivar entre ellos como de poder moverse sin chocar con el escenario. Al principio nos planteamos implementar nosotros mismo un algoritmo A*. Pero después de investigar los paquetes de Unity encontramos NavMesh. A través de este paquete de navegación definimos toda la superficie de la nave como zona caminable y le añadimos las propiedades para poder navegar por la cuadrícula correctamente.

Cada instalación en el juego contiene un punto asociado que es el destino de la navegación. El movimiento a una instalación empieza cuando un tripulante inicia la acción asociada a esa instalación. En ese momento se asigna el punto objetivo al agente de navegación hasta que entra en la instalación y el disparador al entrar del objeto para el movimiento del agente.

5.7 Sistema de acciones

En la primera versión del sistema de acciones se estableció la arquitectura básica mediante una clase abstracta acción de juego (GameAction), de la que heredan las acciones de comer, rellenar, investigar y fabricar (EatAction, RefillAction, ResearchAction y CraftAction). Estas clases tenían una referencia a la instalación a la cual debía moverse el NPC. Su funcionalidad aún era solo mover el tripulante a la instalación donde aún no estaba implementada la lógica.

Las acciones se ejecutaban sólo por orden del jugador. El mensaje era mandado al LLM que respondía con un mensaje en formato JSON con la acción, pero aún sin los parámetros detallados sobre cómo ejecutar la acción. La acción después se añadía a la cola de acciones del tripulante y se establecía como acción Actual.

Se detectó que el LLM tendía a negarse a realizar acciones si se le daba la opción, lo que señaló la necesidad de refinar la peticiones en versiones posteriores. Esta versión sirvió para probar la estructura general de acciones sólo por órdenes antes de implementar el comportamiento autónomo.

5.8 Primera comunicación con el LLM y conversión de respuestas

Uno de los hitos clave del desarrollo del proyecto fue establecer la primera comunicación funcional entre el videojuego y el LLM, que permite dotar de autonomía a los tripulantes mediante decisiones simuladas. Para este propósito como ya se había probado anteriormente con el primer script de prueba de comunicación se utilizó la API de Gemini de Google.

Para la generación de la petición (prompt) para las consultas primero se recogen datos del estado de diferentes elementos del juego, entre ellos el contexto de la nave, con información de cada instalación, como la cantidad de comida disponible o el nivel de combustible en el motor, y también información propia del tripulante sobre el cual se hace la consulta, como su nombre, personalidad, estadísticas actuales y el texto de la orden dada por el jugador.

Después estos datos se les da un formato de:

1. Declaración de intenciones y petición: Se establece la razón de la consulta y el contexto de esta, explicando la existencia de un tripulante en una nave que recibe órdenes cuya respuesta es lo que se va a simular.
2. Identidad del LLM: Se da contexto de la identidad del tripulante a simular por el LLM para tomar decisiones más acertadas respecto a su carácter, cuyo efecto se describe también.
3. Información de contexto: Toda la información recolectada anteriormente que podría ser relevante.
4. Explicación de sistema de juego: Se describen todas las opciones disponibles de respuesta, sus requerimientos y sus efectos dentro del juego.
5. Formato de respuesta deseado: Especificación del formato exacto que debe usar el LLM para responder a este prompt con tal de hacer posible el proceso de captación de la respuesta.

Esta petición (prompt) creada se pasa al módulo de comunicación con el LLM y espera una respuesta asíncrona.

Al recibir la respuesta del modelo se procede al "Parse" o recogimiento de la acción escogida por el LLM, se aplican pruebas de validación de datos que de fallar reiniciaría el proceso de comunicación hasta recibir una respuesta válida o fallar un número específico y limitado de veces.

Una vez validada se transforma la acción en datos de juego, para lo cual se crea un objeto acción del tipo específico que contiene los datos necesarios para realizarse, así como el tripulante y la instalación a los que referencian, luego se introduce en la cola

de acciones del tripulante que realizará la próxima vez que haga la comprobación de acciones en la cola.

5.9 Sistema de parámetros de acción

Con el objetivo de permitir que las decisiones generadas por el LLM sean ejecutables en el contexto del juego, fue necesario definir un sistema que almacenase los posibles parámetros asociados a cada acción que un tripulante puede realizar. Para ello se emplearon archivos en formato JSON que contienen, por cada tipo de acción, una lista de posibles valores válidos para los distintos parámetros implicados. Un ejemplo de parte de esta estructura es el siguiente:

```
"refill": {
   "percentage": ["25", "50", "75", "100"]
 }
```

Que hace referencia a la acción de rellenar combustible en el juego y uno de los parámetros es el porcentaje de capacidad al que debería rellenar el combustible el tripulante, listando una lista de 4 valores posibles que el LLM debe valorar y escoger el más cercano a la orden dada por el jugador.

Estos archivos JSON son cargados en la fase de inicialización del juego y sus datos se asignan internamente a las instalaciones correspondientes, como la cocina, el laboratorio, el taller o el motor, en función de la acción que allí se puede realizar. Así, cada instalación conoce de antemano qué parámetros son necesarios para ejecutar una acción en su dominio y cuáles son sus valores válidos.

Durante la generación del prompt para el LLM, esta información es incluida dentro de la sección de explicación del sistema de juego, donde se describe cada acción posible, qué parámetros requiere, qué valores puede tomar cada uno y las restricciones lógicas que deben respetarse. Por ejemplo, para una acción como "investigar", se explica que requiere un parámetro "horas" con un valor entre 1 y 8, representando las horas que el tripulante dedicará a investigar en el laboratorio.

Una vez obtenida la respuesta del LLM, el sistema de captación extrae no solo el nombre de la acción a realizar, sino también los valores de cada uno de sus parámetros. Estos valores son validados contra los permitidos previamente cargados desde el archivo JSON correspondiente. Si todos los parámetros son correctos, se construye un objeto de tipo acción con esta información, el cual se añade a la cola de acciones del tripulante correspondiente.

Este mecanismo permite que la respuesta textual del modelo de lenguaje se traduzca de forma estructurada en una acción ejecutable dentro del motor del juego, garantizando coherencia con las reglas internas y evitando errores de ejecución por decisiones mal formateadas o imposibles de realizar.

5.10 Acciones libres

Una de las características principales del proyecto es que los tripulantes pueden realizar acciones de forma autónoma, sin intervención del jugador. Estás acciones son escogidas en tiempo real por el LLM del banco de acciones previamente mencionado.

Cada actualización de juego el tripulante mira si tiene alguna acción en su cola de acciones, si está vacía empieza el proceso de creación y realización de acción autónoma y hace que no se pueden crear más hasta que este proceso termine. El proceso consiste en dado el contexto del tripulante y de cada instalación de la nave decide que acción realizar. En el prompt se le da unas indicaciones sobre prioridades que debe de seguir. Debe priorizar realizar la acción asociada a su trabajo sin dejar acciones críticas como el comer o rellenar combustible. Una vez escogida la acción se vuelve a hacer una llamada para que escoja los parámetros de esta (por ejemplo, en la acción de comer sería cuántas raciones comer). Después se crea la acción y pasa a ejecutarse.

En esta fase se logró que el comportamiento de los tripulantes fuese totalmente autónomo si el jugador lo deseaba, pero sus decisiones aún no eran las óptimas, pues priorizaban demasiado el comer repetidas veces más que el hacer las acciones de su trabajo.

5.11 Gestión de inventario

Una parte esencial de la funcionalidad del juego es el sistema de inventario, encargado de almacenar y gestionar tanto los recursos (elementos encontrados naturalmente mediante exploración sin fabricación de por medio) como los materiales (elementos obtenidos mediante fabricación de los tripulantes). Para estructurar este sistema de forma clara y eficiente, se estableció un modelo basado en un archivo JSON que define desde el inicio todos los elementos posibles que pueden formar parte del inventario, primero los recursos separados por nivel de rareza del 1 al 4 indicando el nivel de la nave necesario para encontrar estos recursos en un planeta, y luego los materiales.

Durante la fase de carga del juego, estos datos se utilizan para inicializar la estructura del inventario de la nave. Cada elemento se representa como una entrada con un contador asociado, comenzando todos en cero. A medida que los tripulantes fabrican o consumen elementos, los valores de este inventario se actualizan dinámicamente.

Además de la estructura lógica, se desarrolló una interfaz gráfica de inventario (Ilustración 9). Esta se presenta al jugador como una cuadrícula visual que muestra tanto recursos como materiales de manera indistinta. Cada elemento ocupa una celda representada por un dibujo específico asociado a su tipo, acompañado de un número que indica la cantidad actual disponible.

Para optimizar la legibilidad y evitar ruido visual innecesario, la interfaz solo muestra aquellos elementos cuya cantidad es igual o superior a 1 y en tiempo real, al modificarse el inventario se actualiza automáticamente la visualización.

Este sistema no solo facilita al jugador un control directo sobre los recursos disponibles, sino que también resulta fundamental para validar acciones de los tripulantes que dependen de ciertos materiales. La conexión entre el inventario visual, la lógica interna y el sistema de acciones garantiza coherencia entre las decisiones del LLM y el estado de juego.

Ilustración 9: Diseño de inventario

5.12 Efectos de acciones sobre el sistema de juego

Una vez completada la lógica de comunicación con el LLM y la integración de la respuesta como acciones realizables por los tripulantes, fue necesario desarrollar el sistema que aplicara efectos reales sobre el estado interno del juego como consecuencia directa de dichas acciones. Hasta entonces, las acciones ejecutadas por los tripulantes eran meramente simbólicas, registradas a modo de confirmación por consola, sin producir alteraciones funcionales dentro del juego, enfocadas en el desarrollo.

La implementación se centra en asociar efectos específicos a cada acción dependiendo de los parámetros dados por el LLM al crearse. Siendo estas:

- Cocina(acción:eat):
  Al finalizar la acción de comer, se calcula el efecto sobre el atributo de hambre del tripulante en función de la cantidad de raciones seleccionada y el total

restaurado por cada ración de comida. Simultáneamente, se reduce del inventario general el número correspondiente de raciones disponibles.

- Motor(acción:refill):
  Se traduce directamente en un incremento de la cantidad actual de combustible en el motor. El porcentaje de recarga deseado se obtiene desde el parámetro de acción y se aplica estableciendo el nivel actual de combustible acorde al parámetro de la acción a la vez que se modifica el inventario.
- Laboratorio(acción:research):
  Se implementó el proceso interno asociado al área de investigación descrito en otros apartados. Aplicando los cambios en el tiempo de investigación actual, las recetas de materiales disponibles y la lista de investigaciones.
- Taller(acción:craft):
  Reduce automáticamente del inventario las cantidades necesarias de los materiales y/o recursos definidos por la receta seleccionada y. La validación previa garantiza que existan los materiales suficientes. Luego se añade al inventario el nuevo material resultante en la cantidad especificada en los parámetros de la acción.

Con todo esto las acciones de los tripulantes tienen una repercusión directa y coherente dentro del juego, cerrando así el bucle entre decisión (por parte del LLM), ejecución (por parte del tripulante) y efecto (sobre el estado interno de la nave o el inventario). Esta lógica de consecuencias transforma las decisiones en eventos funcionales, ampliando significativamente la sensación de autonomía de los personajes dentro del entorno.

5.13 Representación visual de tripulantes

Para facilitar el seguimiento del estado individual de cada tripulante y permitir la interacción directa con ellos, se diseñó e implementó una interfaz específica de tripulantes, accesible desde la vista principal del juego (Ilustración 10). Esta interfaz

ofrece al jugador una representación centralizada de cada miembro de la tripulación, así como permitir la emisión de órdenes de lenguaje natural.

Al hacer clic sobre un tripulante la interfaz muestra los siguientes elementos clave:

- Nombre del tripulante, extraído de su perfil interno.
- Sprite representativo, utilizado también en el mapa general.
- Personalidad asignada, que influye directamente en la interpretación de instrucciones por parte del LLM.
- Nivel de hambre actual, reflejado en valor numérico.
- Acción actual en curso

A la izquierda de la interfaz se encuentra un botón para abrir el panel de comunicación (Ilustración 11), una sección fundamental que permite al jugador enviar órdenes en lenguaje natural al tripulante seleccionado. Esta subinterfaz incluye:

- Un cuadro de texto donde se redacta la instrucción.
- Un botón de envío, que lanza el proceso completo de generación del prompt, envío al LLM, captación de respuesta y asignación automática de la acción correspondiente al tripulante y se actualiza su estado.

Ilustración 10: Diseño interfaz de tripulante

Ilustración 11: Interfaz de comunicación

5.14 Generación automatizada de NPC

Se automatizó la generación automática de tripulantes para utilizarse como punto de partida para poder poblar la nave por instancias de tripulante a demanda del sistema.

Este proceso se basa en un prefabricado preconfigurado de tripulante, que contiene toda la lógica, referencias y componentes visuales necesarios para instanciar correctamente un nuevo personaje en el entorno del juego. Al activarse la función de generación, este prefabricado se instancia en la posición central de aparición de la nave.

Durante la creación, se asignan valores aleatorios a los campos clave de cada tripulante, tomados de un archivo JSON externo cargado al inicio de la partida. Concretamente:

- Nombre elegido de una lista de más de 100 opciones distintas.
- Personalidad se selecciona aleatoriamente de entre cinco predefinidos:
    - Leal: Siempre sigue las órdenes sin cuestionarlas.
    - Cauto: Duda si la tarea parece arriesgada.
    - Rebelde: Ignora órdenes que considera innecesarias.
      Pragmático: Sólo sigue órdenes lógicas o útiles.
    - Curioso: Puede explorar alternativas antes de actuar.

Estos atributos determinan no solo el aspecto narrativo del personaje, sino también su comportamiento posterior en la interacción con el LLM, ya que la personalidad influye directamente en cómo interpreta y decide ante las instrucciones recibidas.

Inicialmente, se genera un único tripulante al inicio de la partida, como parte del estado base de la nave. Sin embargo, el sistema de generación fue diseñado desde el inicio para ser reutilizable en cualquier punto del juego. Mediante una simple función, se puede instanciar un nuevo tripulante con todos sus parámetros configurados

dinámicamente, lo que abre la puerta a mecánicas como incorporación de nuevos miembros de la tripulación, eventos especiales o reemplazos a futuro.

Este sistema modular y escalable facilita futuras expansiones del juego sin necesidad de grandes modificaciones estructurales.

Durante la implementación del sistema de acciones por parte de los tripulantes surgieron múltiples problemas derivados de cómo se gestionaba internamente las distintas instancias de estos personajes. Cuando se introdujo más de un tripulante en el juego, aunque cada uno tenía su propia pila de acciones y lógica independiente, en la práctica se producían bloqueos y comportamientos erráticos cuando dos o más tripulantes intentaban realizar acciones al mismo tiempo. El error no venía de colisiones físicas, sino de cómo cada instalación gestionaba internamente la entrada, el uso y la salida de los tripulantes. Al haber múltiples instancias activas, las condiciones que controlaban si una instalación estaba ocupada, si una acción podía comenzar o si una tarea se había completado satisfactoriamente, no estaban correctamente aisladas y se pisaban entre sí. Esto causaba que los tripulantes quedaran atascados intentando hacer una acción que nunca terminaba o que no podían iniciar. Para solucionarlo, se rehízo toda la lógica de interacción con las instalaciones, con una validación clara y controlada en cada fase del proceso. Además, se reforzó la gestión interna de la pila de acciones para asegurar que cada tripulante pudiera actuar sin afectar ni depender del estado de los demás.

5.15 Mecánicas de investigación y fabricación

En la primera versión de investigación y fabricación se desarrollaron sin la interacción entre estos debido a que queríamos centrarnos en terminar el funcionamiento principal de todas las instalaciones.

La investigación se implementó usando un JSON como fuente de datos para cargar todas las investigaciones y sus desbloqueos. Al entrar en la instalación se inicia una subrutina que actualiza el estado de la investigación actual cíclicamente e inicia una nueva si esta termina. Al terminar de implementar esta instalación nos dimos de cuenta

de un error que ocurría en todas las instalaciones si el tripulante repetía la misma acción dos veces seguidas. Debido a como estaba previamente implementado solo comprobaba e iniciaba el estado del tripulante al entrar en la instalación así que se tuvo que cambiar a comprobar en cada ciclo si estaba dentro, no estaba realizando ninguna acción y la acción que tenía que realizar después era la relacionada con la instalación. Este cambio se realizó en todas las instalaciones.

En el caso de la fabricación esta se implementa usando una clase auxiliar llamada recetas que guardaba los recursos necesarios para fabricar cada recurso y se cargaba al iniciar la partida también de un JSON. En esta versión aún no se desbloqueaban las recetas al investigar y teníamos todas desbloqueadas de primeras para poder realizar pruebas.

5.16 Mecánicas de navegación, reclutamiento y exploración

La navegación es el sistema que hace que puedas obtener nuevos recursos y seguir avanzando en el progreso del juego. Este se implementa mediante una generación pseudoaleatoria de planetas. Al generarse un nuevo planeta se escoge qué recursos se van a poder obtener en este planeta solo pudiendo conseguir los recursos del nivel correspondiente de nave. Luego aleatoriamente se decide la cantidad máxima de cada recurso que se podrá obtener. El planeta también se crea con un nombre aleatorio de una lista y con una imagen aleatoria que pasa a ser el fondo de la nave para reflejar en qué planeta te encuentras. Se implementó el viajar entre planetas utilizando combustible de la nave.

El sistema de reclutamiento se implementó como una interfaz secundaria dentro del módulo de navegación, diseñada para seleccionar qué tripulantes participarán en una expedición al planeta actual. Al activarse, muestra una lista con todos los tripulantes disponibles y permite al jugador seleccionar hasta un máximo de tres. La selección se realiza de forma sencilla y visual, añadiendo a los elegidos a una lista interna de reclutados. Esta lista se transfiere al sistema de navegación, donde se utiliza para planificar la expedición. Cuando esta se inicia, se añade automáticamente la acción

de explorar a la cola de acciones de cada tripulante reclutado. Aunque actualmente el sistema es directo, está previsto como mejora futura introducir una fase de interacción con cada candidato, en la que el jugador deberá convencer al tripulante a través de un prompt procesado por el LLM para que acepte unirse a la misión.

Para realizar la exploración de un planeta primero debes escoger un tripulante para que forme parte de la expedición y este se mete en una lista. Al iniciar la exploración todos los tripulantes que están en esa lista se les mete en la cola de acciones la acción de ir a la exploración. Una vez llegan a la puerta por donde salen a la exploración todos los participantes se inicia una subrutina que va calculando el tiempo que llevan de exploración. Cuando el usuario decide terminar la exploración se calculan las recompensas a obtener y se meten en el inventario de la nave. La cantidad de recompensas se obtiene aleatoriamente entre un mínimo establecido en el JSON de recompensas y el máximo que va aumentando gradualmente con el tiempo que lleva el tripulante en la expedición hasta llegar al 100%.

5.17 Interfaces

En esta fase se desarrollaron el resto de las interfaces principales del juego, tanto para la experiencia del jugador como para facilitar el proceso de pruebas y localización de errores. Entre las más complejas, interfaz de navegación (Ilustración 13), que actúa como panel central de gestión planetaria. Esta incluye un visor completo del planeta actual con una imagen animada (seleccionado aleatoriamente de entre 25 modelos), nombre también aleatorio extraído de un JSON con una gran variedad de opciones, y los materiales disponibles según la generación aleatoria de planeta (Ilustración 14). También se muestra el nivel actual de combustible, el requerido para viajar y botones para iniciar el viaje. La interfaz cambia dinámicamente en función del estado de reclutamiento (completo o no), lo que determina si se puede iniciar o no una expedición. Tras iniciar esta, los botones se sustituyen por uno para finalizar la exploración y se bloquea la opción de viajar (Ilustración 16).

Desde la navegación se accede directamente a la interfaz secundaria de reclutamiento (Ilustración 15). Además, se implementaron otras interfaces como la de cocina, que muestra la comida disponible y se actualiza en tiempo real con cada consumo o recarga. Junto a esta, se retocaron las ya mencionadas interfaces de inventario, tripulante y comunicación con el LLM.

Todas estas interfaces se diseñaron cuidadosamente con una lógica interactiva robusta que previene errores del jugador (por ejemplo, realizar acciones imposibles en un estado concreto del juego), lo que no solo mejora la experiencia general, sino que también reduce notablemente los posibles errores lógicos del sistema, simplificando su gestión a nivel de código. Está asimismo previsto para la fácil integración de las interfaces de investigación y fabricación.

Ilustración 12: Interfaz de cocina

Ilustración 13: Interfaz de navegación

Ilustración 14: Interfaz de navegación tras viajar a planeta

Ilustración 15: Interfaz de reclutamiento

Ilustración 16: Interfaz de navegación después de exploración

5.18 Versiones finales

Al ya tener todas las demás instalaciones funcionando bien, decido realizar la versión final de investigación y fabricación. En esta versión se implementó la interacción entre ambas acciones e instalaciones.

En la investigación se añadió una referencia al taller para poder usar una función para desbloquear las recetas cuando la investigación correspondiente finaliza. Además, se añadió una función que se utilizaría más tarde para desbloquear las nuevas investigaciones cuando la nave suba de nivel para así ya poder acceder a todo el contenido del juego.

En fabricación se implementó una función para comprobar si era posible fabricar la receta con los materiales del inventario y además el uso de los parámetros de la acción para poder fabricar más de un material a la vez.

Durante todo el proyecto hemos ido poco a poco mejorando el prompt y contexto que mandamos al modelo para así hacer que los NPC cada vez tomasen decisiones más correctas. En las últimas etapas nos centramos en hacer que todas las instalaciones y acciones tengan bien definido su contexto para dar una idea más clara al tripulante del entorno. Añadimos en las instalaciones de fabricación e investigación toda la información sobre las recetas e investigaciones disponible, así como el progreso y con eso conseguimos mejorar la toma de decisiones, eso se pudo ver reflejado en que los tripulantes cuando no tienen recetas disponibles o no quedan materiales autónomamente deciden ir a investigar la próxima receta. Además, se añadió al prompt un contexto de cada acción en la que se describe cada parámetro y qué repercusión va a tener en la partida.

Después de estos cambios y de modificar las prioridades que le decíamos a los tripulantes para que no se centraran solo en sobrevivir y priorizaran avanzar la partida pudimos observar que ya no se quedaba atascado realizando la misma acción sin

sentido y consiguen convertir todos los recursos almacenados en recursos más avanzados ellos solos.

Por último, implementamos la subida de nivel de la nave. Esto se hizo mediante una nueva acción que solo puede ser realizada mediante una orden y que si tienes los materiales necesarios para mejorar la nave hace que la suba de nivel y desbloquee las nuevas investigaciones y materiales en planetas.

6 Pruebas

6.1 Objetivo de las pruebas

El objetivo principal de las pruebas de usabilidad es evaluar si los jugadores comprenden de manera intuitiva cómo comunicarse con los tripulantes y cómo darles órdenes y comprobar si el LLM responde de forma correcta.

Se quiere analizar:

- La eficacia de la IA para interpretar órdenes en lenguaje natural
- Los posibles bloqueos durante el proceso
- La facilidad de uso del sistema de interfaces de interacción con los NPCs.

6.2 Hipótesis

A continuación, se definen las hipótesis a validar:

1. **Hipótesis 1:** Los jugadores pueden sin ayuda previa interactuar con los tripulantes.
2. **Hipótesis 2:** Las órdenes dadas en lenguaje natural son correctamente interpretadas por el LLM.
3. **Hipótesis 3:** Los usuarios perciben que los tripulantes actúan de forma coherente y autónoma.

6.3 Diseño de las pruebas

6.3.1 Perfil de los participantes

Para evaluar la usabilidad y la accesibilidad de la interfaz basada en comandos de lenguaje natural, se reclutó una muestra de ocho participantes con edades comprendidas entre 17 y 68 años. Este rango generacional permitió observar cómo usuarios de distintos grupos de edad comprenden y utilizan las comunicaciones de texto para interactuar con el prototipo. Entre ellos se incluyeron estudiantes universitarios, profesionales del sector tecnológico, aficionados a los videojuegos y usuarios sin experiencia previa en desarrollo de software ni experiencia significativa en videojuegos,

asegurando así una visión amplia de distintos niveles de familiaridad con interfaces digitales y mecánicas de juego.

El objetivo de seleccionar un perfil heterogéneo fue analizar diferencias en la rapidez de aprendizaje, precisión en la formulación de comandos y nivel de satisfacción con la interacción por lenguaje natural.

6.3.2 Metodología

La prueba se divide en dos partes:

Fase 1:

Tarea: El jugador debe lograr que el tripulante realice la acción que él quiera de las posibles del juego sin explicación de cómo hacerlo.

Explicaciones previas: Se les dijo las 4 acciones posibles (Comer, investigar, fabricar, rellenar combustible).

Al acabar esta fase se realizará una serie de preguntas para evaluar las hipótesis:

- ¿Lograste dar una orden a la IA con éxito?
- ¿Cómo describirías la experiencia de dar órdenes a los tripulantes?
- ¿Te hubiera ayudado tener un tutorial o guía previa?

En esta fase se pretende observar la facilidad de entender el sistema y cómo dar órdenes a los tripulantes sin ninguna explicación previa.

Fase 2:

Tarea: El jugador debe conseguir fabricar 1 objeto de cada del nivel 1 de la nave (carbonite y timberite)

Explicaciones previas: Se les hará una breve explicación de cómo se desbloquean las recetas y que recetas tiene que conseguir completar y de cómo obtener más recursos.

Al acabar esta fase se realizará una serie de preguntas para evaluar las hipótesis:

- ¿Lograste dar una orden a la IA con éxito?
- ¿La IA hizo lo que tú querías que hiciera?
- ¿Lograste obtener los materiales pedidos?

En esta fase se quiere ver si el LLM responde bien a las órdenes dadas y si es posible terminar un nivel de nave.

6.4 Resultados

Hipótesis 1: Validada: Todos los participantes (8 de 8) lograron dar órdenes a la IA con éxito sin recibir instrucciones previas (Ilustración 17, Ilustración 20). Y aunque la mayoría (5 de 8) calificó la experiencia de "aceptable" o "intuitiva" (2 de 8) una persona la calificó de confusa (Ilustración 18). Esto indica que la mecánica es comprensible y funcional desde el primer contacto, si bien podría beneficiarse de una mejora en la claridad o accesibilidad inicial.

Hipótesis 2: Parcialmente validada: La mayoría de los usuarios (6 de 8) indicaron que la IA hizo lo que esperaban "siempre" o "casi siempre", mientras que 2 señalaron que solo lo consiguió "a veces" (Ilustración 21). Esto demuestra una buena capacidad por parte del LLM al entender ordenes, aunque existen casos en los que la respuesta no se ajusta a intención del usuario, especialmente en peticiones poco descriptivas o muy directas como "comer" o "rellena".

Hipótesis 3: Parcialmente validada. Aunque todos los jugadores completaron tareas con éxito y en general consiguieron los materiales solicitados, dos de ellos indicaron que solo obtuvieron parcialmente lo pedido. Asimismo, aunque no hubo comentarios explícitos sobre incoherencia, la necesidad de un tutorial fue señalada por 6 de los 8 participantes, lo que sugiere que la autonomía del sistema no es siempre percibida como clara o predecible. Esto podría estar relacionado con la falta de retroalimentación visual o explicativa del comportamiento de los NPCs punto que fue mencionado por los participantes en diferentes ocasiones.

Conclusión general

Los resultados indican que la funcionalidad básica del sistema es funcional y comprensible para los jugadores, incluso sin formación previa. No obstante, se identifican oportunidades claras de mejora en cuanto a la interpretación del lenguaje natural en contextos ambiguos y en la percepción de la autonomía de los NPCs. Se recomienda incluir una breve guía inicial y reforzar la retroalimentación tras las acciones de los tripulantes para mejorar la experiencia global del jugador.

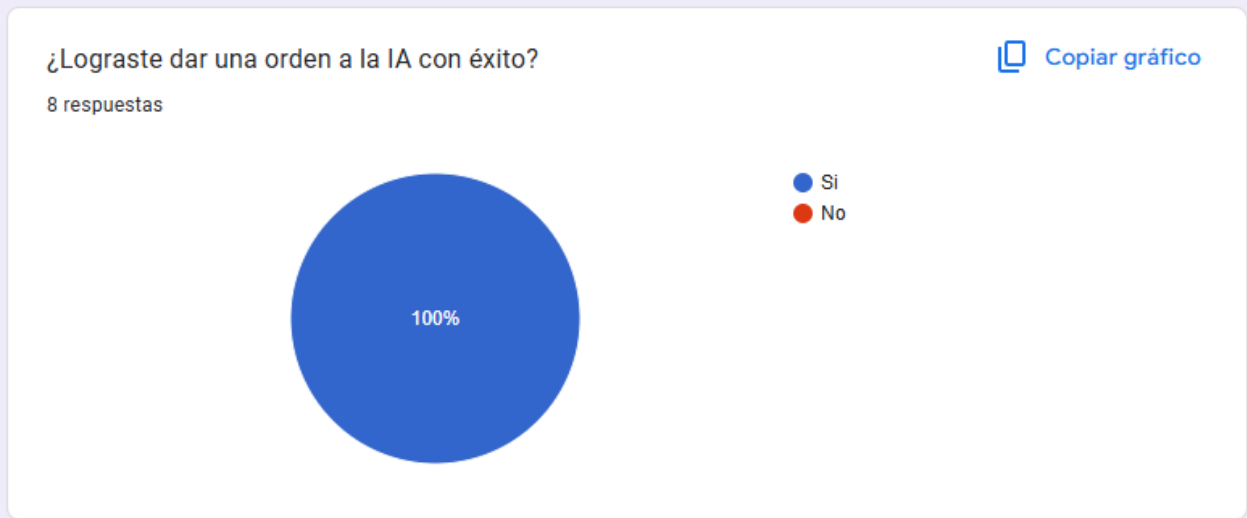

Ilustración 17: Gráfico fase 1.a

Ilustración 18: Gráfico fase 1.b

Ilustración 19: Gráfico fase 1.c

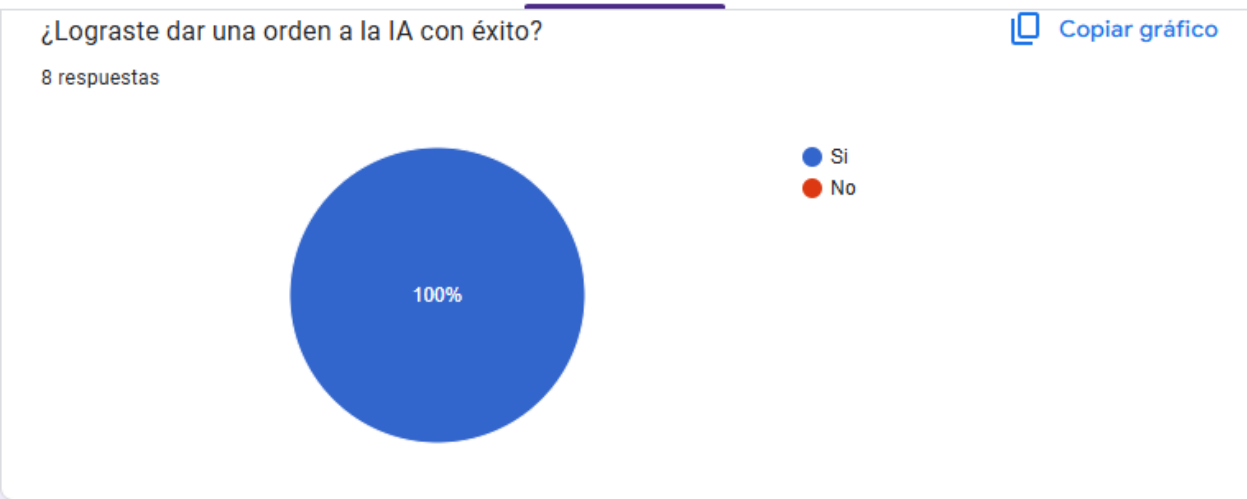

Ilustración 20: Gráfico fase 2.a

Ilustración 21: Gráfico fase 2.b

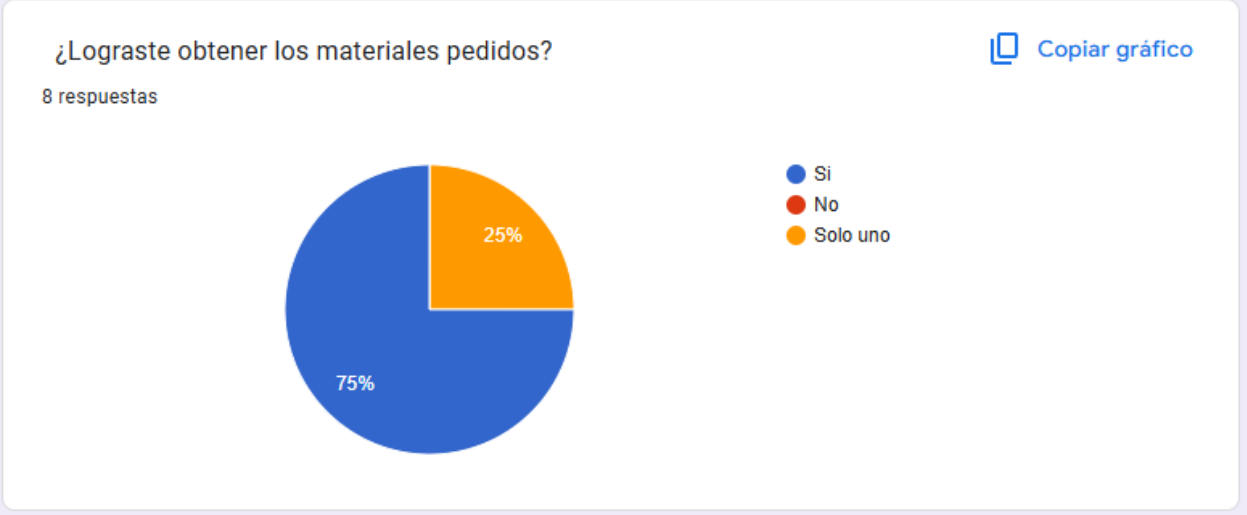

Ilustración 22: Gráfico fase 2.c

7 Conclusiones y trabajo futuro

7.1 Conclusiones

Con este trabajo se ha podido demostrar la viabilidad de utilizar un LLM como controlador de NPC dentro de un videojuego de gestión de una tripulación. Implementando un prototipo funcional en Unity y con la API de Gemini como motor de razonamiento, se ha conseguido que los tripulantes sean capaces de tomar decisiones autónomas coherentes con el contexto actual del juego. Estas decisiones incluyen saber cuándo investigar nuevas recetas y cuándo fabricarlas para poder seguir subiendo niveles de la nave además de mantener siempre su nivel de hambre y el nivel de combustible de la nave para así poder viajar a nuevos planetas.

También se ha logrado que los NPC comprendan y ejecuten acciones dadas en lenguaje natural por el usuario, transformándose correctamente en acciones disponibles dentro del juego. Con esto conseguimos una interacción más fluida entre el jugador y el sistema de juego, obteniendo así un avance en la jugabilidad y la experiencia del usuario.

El usar un LLM como controlador supone un reto técnico mayor al método tradicional. Sin embargo, los resultados obtenidos demuestran que este enfoque permite crear videojuegos mucho más inteligentes y adaptativos. Alcanzar un nivel similar de comportamiento dinámico y contextual con técnicas tradicionales sería extremadamente complejo y costoso. Pese a las limitaciones comerciales por el uso de Gemini, principalmente el coste asociado a cada interacción o la necesidad de una conexión a la red, el sistema tiene un rendimiento adecuado para un prototipo funcional.

7.2 Trabajo futuro

Aunque el prototipo desarrollado alcanzó los objetivos principales, aún hay funcionalidades previstas que no ha dado tiempo a desarrollar dentro del plazo del

trabajo. Estas funcionalidades se proponen como trabajo futuro para enriquecer y completar la experiencia de juego. Entre ellas se encuentran:

- **Gestión de trabajos**: Implementar la posibilidad de cambiar los trabajos de los tripulantes para ajustarlas al estilo de juego de cada usuario,
- **Sistema de convencer tripulantes:** Permitir que los tripulantes puedan negarse a ir a una expedición y tengan que ser convencidos mediante lenguaje natural a formar parte de esta.
- **Gestión de partidas:** Implementar un sistema de guardar, cargar y gestionar múltiples partidas.
- **Sistema de misiones:** Introducir un sistema de misiones para facilitar la progresión y guiar al jugador en el desarrollo de una partida.
- **Ciclo día-noche y horarios:** Implementar un sistema de día y noche y horarios en los trabajados de los tripulantes, así como un medidor de sueño para dar motivo a esta funcionalidad.
- **Selector de tripulación inicial:** Permitir al usuario elegir una tripulación inicial de un grupo generado aleatoriamente para darle más control sobre la partida.
- **Eventos aleatorios:** Incorporar un sistema de eventos inesperados que afecten el rumbo de la partida y generen variedad en las sesiones de juego.

Además, una línea diferente para seguir investigando sería desarrollar un modelo de lenguaje propio, entrenado con datos específicos del juego. Esto permitiría eliminar el coste y la dependencia de un servicio externo y haría que fuera mucho más factible en un ambiente comercial. Además, esto daría más control sobre el comportamiento de los NPC y permitiría que hicieran acciones más complejas.

Introduction

Motivation

Today, the video game industry is undergoing a paradigm shift: players increasingly demand personalized, immersive experiences that respond dynamically to their decisions. Traditional architectures based on menus and predefined systems limit both the naturalness of interaction and narrative depth. At the same time, Large Language Models (LLMs) have matured to the point where they can understand and generate high-quality natural language, enabling new forms of dialogue, decision-making, and dynamic content generation.

Our project leverages this advancement by integrating an LLM directly into Unity, allowing non-playable characters (NPCs) in a space simulation game to not only execute precise actions based on textual commands but also interpret the context of the ship, their internal states (hunger, mood, skills), and the procedural environment of planets. In this way, each interaction becomes an emergent experience: crew members can justify their decisions, prioritize tasks according to their personalities, and adapt to unforeseen events generated by the AI.

This approach brings several key innovations:

- **Dynamic Narrative**: NPC reactions are generated in real-time, avoiding static interactions.
- **Natural Control**: Players communicate through free-form text, moving away from control schemes based on buttons and menus.
- **Simulation Depth**: By combining LLMs, technological progression systems, and resource management, we create a coherent game world where each player's decision has tangible consequences.

By exploring this convergence between conversational AI and interactive entertainment, we lay the foundation for future video games that are more human-like, adaptive, and capable of offering unique experiences in every playthrough.

Goals

The main objective of this Bachelor's Thesis is to develop a 2D space simulation video game where the control of NPCs and core game mechanics is handled through natural language commands processed by an LLM. To achieve this, several specific objectives must be met:

- Acquire a solid understanding of the capabilities and limitations of LLMs in video game contexts by reviewing academic literature and API documentation to select the most suitable solution.
- Design and validate a unified communication format between the game engine and the LLM, defining the structure of queries (game context, action parameters, NPC status) and the response schema in a structured format to translate model decisions into concrete in-game actions.
- Develop the LLM interaction core, implementing a C# module capable of sending asynchronous requests to the API, managing retries due to network or formatting errors, and exposing an internal interface that feeds the crew members' logic.
- Implement automated crew member generation using prefabs, assigning them random names and personalities extracted from JSON (JavaScript Object Notation) files, and ensure this process can occur both at game start and during simulation runtime.
- Create resource management and technological progression mechanics for the ship, including the definition and loading of inventories (resources and materials),

storage of action parameters, and real application of their effects (hunger, research, fuel, manufacturing) on game state.

- Develop core interaction interfaces that ensure a clear experience, prevent invalid actions, and enhance immersion.
- Test and debug concurrent behavior among multiple crew members, refining action stack handling and module input/output conditions to ensure a smooth simulation.

Successfully completing these objectives ensures that the main goal of the project is achieved.

Work plan

An agile methodology based on Scrum [6] was chosen for the planning and execution of this project, dividing work into "Sprints"—fixed-duration cycles that allow for rapid iteration at the project's start and structured development phases.

Each sprint includes specific tasks. In early phases, clear deliverables were defined, and in later development phases, user stories (US) were used.

Tasks could be in one of five different states:

1. TO DO
2. IN PROGRESS
3. ON HOLD
4. UNDER REVIEW
5. DONE

At the end of each sprint, meetings were held to review completed tasks and plan those to be continued in the next sprint.

Project management was handled using **Jira** [7] (Figure 1), where all tasks and user stories were organized (Figures 3 and 4), enabling clear traceability of the work completed in each sprint. The project was stored and managed on **GitHub** as a collaborative repository: https://github.com/TGF-2024-25/sims

Project Phases

Development was organized into variable-length sprints using Scrum, with **Jira** as the tracking tool (Figure 2). Each phase corresponds to one or more sprints and ensures an orderly progression from initial research to final delivery.

The project began in September 2024, and progressed through the following phases:

Research and Design

This phase corresponds to Chapters 2.2 "State of the Art" and 2.3 "Technologies". In the first weeks (September–October), a literature review was conducted on the integration of LLMs in video games, client-server architectures in game engines, and APIs. Technical feasibility of communicating with an LLM via HTTP/REST and JSON protocols was explored. The outcome was a viability assessment establishing the project's conceptual and technological foundations.

Initial Requirements Definition

This phase and the next correspond to Chapters 3.1 "Functional Requirements" and 3.2 "Technological Progression and Resources". Using the acquired knowledge, an initial Jira backlog was created in October–November, breaking down early functional requirements into user stories and tasks. Critical features like LLM-Unity communication and basic crew member movement were prioritized.

Requirements Refinement

During several Jira review sessions (December–January), both functional and non-functional backlogs were refined. Acceptance criteria were adjusted, and the MVP scope was defined.

Diagram Design

Corresponding to Chapter 4 "Design Diagrams", between January and February, the main UML [8] artifacts were modeled:

- Class diagram (Figure 5) for main elements such as GameSession, Ship, CrewMember, Facilities, and LLM Manager.
- Sequence diagrams for API communication and crew decision-making.
- Event flow and technological progression diagrams.
  These diagrams guided development and are included in Chapter 3.

Core Functionality Development

This phase corresponds to Chapters 5.1 "Initial LLM Communication Script" to 5.14 "Automated NPC Generation". From January to March, the LLM communication pipeline was implemented (context sending, action reception), the basic crew control system, interaction manager, and adaptable UI with canvas sorting and animation controllers. By the end, crew members could understand commands, move, generate autonomous actions based on internal and external data, and apply effects to the ship.

Advanced Mechanics Development

This phase covers Chapters 5.15 "Research and Manufacturing Mechanics" to 5.18 "Final Versions". From March to April, the following were added:

- Resource management and technological progression module (ship level, manufacturing, research).

- Generation and management of multiple crew entities.
- Navigation system, planet generation, crew recruitment, exploration, and reward handling.
- Final version of the LLM communication and data conversion model.
- Graphical interfaces for managing crew, inventory, facilities, etc.

Final Development and Testing

This phase corresponds to the end of Chapter 5 "Implementation" and Chapter 6 "Testing". At the end of April, gameplay testing helped fix UI scaling and NPC behavior bugs, as well as issues caused by multiple crew instances. These were resolved by adjusting installation collisions and handling crew instance management during actions. Once fixed, test cases were written, executed, and documented.

Final Report Writing

In May, Chapters 1 and 2 (Introduction, State of the Art, Technologies) were reviewed, diagrams and comparison tables were added to Chapter 3, the development section was completed, and conclusions and future work were written. Finally, citations were revised, the automatic table of contents was updated, and the final PDF version was exported for submission.

Conclusions and future work

Conclusions

This project has demonstrated the feasibility of using a Large Language Model (LLM) as a controller for NPCs within a crew management video game. By implementing a functional prototype in Unity and using the Gemini API as the reasoning engine, crew members were able to make autonomous decisions consistent with the game's current context. These decisions include knowing when to research new recipes and when to craft them in order to continue leveling up the ship, as well as maintaining appropriate hunger levels and fuel reserves to allow travel to new planets.

The NPCs are also capable of understanding and executing actions given in natural language by the user, successfully transforming them into available in-game actions. This results in a more fluid interaction between the player and the game system, enhancing both gameplay and user experience.

Using an LLM as a controller poses a greater technical challenge compared to the traditional method. However, the results obtained show that this approach enables the creation of much smarter and more adaptive video games. Achieving a similar level of dynamic and contextual behavior with traditional techniques would be extremely complex and costly. Despite the commercial limitations of using Gemini—mainly the cost associated with each interaction and the need for a network connection—the system performs adequately for a functional prototype.

Future Work

Although the developed prototype achieved the main objectives, some planned features could not be implemented within the project's timeframe. These features are

proposed as future work to further enrich and complete the gameplay experience. They include:

- **Job management**: Allow players to change crew members' roles to better fit their preferred playstyle.
- **Crew persuasion system**: Implement scenarios where crew members can refuse to go on expeditions and must be persuaded through natural language to participate.
- **Game session management**: Introduce a save/load system to manage multiple game sessions.
- **Mission system**: Add a mission structure to guide player progression and enhance the gameplay experience.
- **Day-night cycle and scheduling**: Implement a time system with crew schedules and a sleep meter to support this mechanic.
- **Initial crew selector**: Let users choose their starting crew from a randomly generated pool, giving them more control at the start of a session.
- **Random events**: Introduce unexpected events that can alter the course of a session and bring greater variety to gameplay.

Additionally, a separate line of investigation would be to develop a custom language model trained on data specific to the game. This would eliminate the cost and dependency on external services, making the system much more viable in a commercial environment. Moreover, it would offer greater control over NPC behavior and enable more complex actions.

CONTRIBUCIONES PERSONALES

Pablo Zapico García

Durante el desarrollo del Trabajo de Fin de Grado he participado en el diseño e implementación del videojuego, centrándome especialmente en los sistemas vinculados a la integración del modelo de lenguaje (LLM), las mecánicas internas de la nave y la progresión del jugador. A continuación, detallo mis principales contribuciones siguiendo el orden en que fueron abordadas a lo largo del proyecto.

Lo primero que hice fue la creación del script inicial que permitía establecer la conexión entre el videojuego y el modelo de lenguaje (LLM). Esta tarea fue crucial para validar el enfoque general antes de poder seguir avanzando con el proyecto, ya que con ella conseguimos que los personajes no jugadores (tripulantes) pudieran recibir e interpretar órdenes dadas por el jugador en lenguaje natural. Este primer prototipo sentó las bases de todas las interacciones futuras, y supuso también una exploración inicial del formato de los prompts, las configuraciones de API y la respuesta del modelo.

Una vez validada la comunicación con el modelo, abordé la creación de las instalaciones internas de la nave, como el laboratorio, la cocina, el taller, el almacén y el módulo de navegación. Cada instalación cuenta con sus propios sistemas y comportamientos, y cumple funciones específicas dentro del juego: desde la fabricación de materiales hasta la ejecución de investigaciones científicas. De estas instalaciones yo solo me centré en la parte de la lógica interna y funcionalidad no en la parte de interfaces.

Con el sistema de instalaciones operativo, trabajé en la integración del sistema de movimiento de los tripulantes utilizando Unity NavMesh. Este sistema permite a los personajes moverse de forma dinámica por la nave, esquivando obstáculos y seleccionando rutas eficientes hacia su destino. Su implementación nos ahorró mucho

tiempo y fue sencilla puesto que si hubiéramos tenido que implementar un A* desde 0 habría retrasado mucho el trabajo.

A continuación, desarrollé las acciones iniciales que los tripulantes pueden ejecutar, , iniciar una investigación, comer o fabricar materiales. Estas acciones representan la base del sistema de comportamiento del juego, y fueron diseñadas con estructuras que permiten su expansión y personalización según el estado del tripulante y el contexto de la partida. Aunque en esa etapa aún carecían de funcionalidad que cambiara el estado del juego.

Una de las funcionalidades más interesantes que abordé fue la implementación del sistema de "Free Will". Este sistema permite a los tripulantes tomar decisiones por iniciativa propia cuando no reciben órdenes directas del jugador. En función de su nivel de hambre, motivación o contexto actual, pueden decidir ir a comer, investigar o realizar otras acciones. Este componente refuerza la sensación de autonomía y convierte a los NPCs en agentes con comportamiento creíble dentro del mundo del juego.

Con las acciones en marcha, implementé el inventario de la nave. Este sistema gestiona todos los recursos disponibles, tanto en las instalaciones como en el propio almacén general, y permite el intercambio de materiales entre zonas. El inventario se conecta también con las acciones de fabricación, exploración y mejora, por lo que su estabilidad y organización era clave para el funcionamiento fluido del juego.

Posteriormente, me encargué de implementar el sistema de investigación y fabricación. En el laboratorio, los tripulantes pueden desbloquear nuevos materiales en el taller, pueden combinar materiales básicos para crear componentes más complejos. Este sistema de progresión está directamente conectado con la evolución de la nave.

Con los sistemas internos ya avanzados, pasé a trabajar en la generación de planetas. Cada planeta contiene una combinación concreta de recursos, y solo puede accederse a ellos si la nave ha alcanzado el nivel adecuado. Diseñé un sistema de

desbloqueo progresivo que introduce nuevos materiales a medida que el jugador avanza, reforzando la curva de progresión.

En conexión con los planetas, implementé el sistema de exploración, que permite seleccionar tripulantes para expediciones en planetas no explorados. Estas misiones sirven para obtener nuevos materiales.

Con la experiencia acumulada, reescribí la comunicación con el modelo de lenguaje para crear una versión final optimizada. Esta nueva versión mejoró la estructura de los prompts, e introdujo un control más fino del contexto enviado al modelo. Integré un contexto más personalizado a cada instalación y acción para facilitar el entendimiento de realizar cada acción del LLM. Esto permitió aumentar la coherencia y calidad de las decisiones tomadas por los tripulantes, especialmente en situaciones más complejas.

Por último, diseñé e implementé el sistema de mejora de la nave. Cada mejora requiere materiales específicos y desbloquea nuevas recetas y recursos. Este sistema motiva la exploración y permite al jugador sentir una progresión tangible como recompensa por una buena gestión.

Gabriela Díaz Marín

Cuando comenzamos a plantearnos el Trabajo de Fin de Grado, me sentí especialmente atraída por la posibilidad de combinar mis conocimientos de Unity con las capacidades emergentes de los Modelos de Lenguaje de Gran Tamaño (LLM). Junto a mi equipo, definimos desde el primer día que debíamos construir un prototipo sólido que mostrara cómo un LLM podía dirigir comportamientos de NPCs en un entorno 2D. Para ello, organicé mis aportaciones en dos grandes bloques: el desarrollo de la infraestructura técnica en Unity y la creación de sistemas de datos que permitiesen dinamizar la experiencia de juego.

Mi primera tarea fue diseñar el sistema mapa de mosaicos que sustenta la representación de la nave. Configuré las celdas, las colisiones y las capas de decoración para garantizar un espacio de juego estable, escalable a distintas resoluciones y donde el movimiento de los tripulantes resultara natural. Paralelamente, creé el prefabricado del tripulante inicial, integrando componentes de física, animaciones básicas y un controlador de selección que serviría como base para instanciar nuevos NPCs. Esta fase sentó las bases gráficas y lógicas de toda la simulación.

A continuación, desarrollé la versión primera de la comunicación con el LLM, estableciendo las primeras llamadas a la API de Gemini desde Unity y diseñando el esquema de prompts que incluye contexto de la nave, estado del tripulante y opciones de acción. Para estructurar las posibles decisiones, definí un JSON de parámetros de acción (comer, recargar, investigar, fabricar) y construí el módulo que lo carga al inicio del juego, lo inyecta en los prompts y valida la respuesta del modelo. Implementé la lógica de captador de respuestas que traduce el archivo JSON devuelto por el LLM en objetos de acción concretos, creando una fábrica de instrucciones que aseguran la correcta ejecución de cada tarea por parte del NPC.

Con el mecanismo de acción en marcha, abordé la generación automática de tripulantes. Diseñé una función reutilizable que instancia el prefabricado original y le asigna, de manera aleatoria, un nombre extraído de un amplio listado JSON y una de

cinco personalidades predefinidas. Esta automatización fue clave para poblar la nave de forma dinámica, tanto al inicio de cada partida como durante eventos de juego.

En paralelo, creé la base de datos interna y la interfaz de inventario. Definí la estructura de almacenamiento de recursos y materiales, programé los métodos para aumentar y disminuir cantidades desde las acciones de los NPC, y desarrollé una cuadrícula UI que muestra, en tiempo real, solo aquellos ítems con unidades Este sistema no solo facilita la gestión del jugador, sino que sirve como punto de control para validar la ejecución de acciones que consumen o generan recursos.

Uno de los retos más complejos fue el módulo de navegación planetaria. Diseñé la interfaz que presenta una imagen animada del planeta actual, su nombre aleatorio y los materiales disponibles, junto con el cálculo de combustible necesario para el siguiente viaje. Este componente requiere sincronizar datos procedurales y estados de inventario, y puso a prueba la robustez de la comunicación entre submódulos.

Sobre esa interfaz, implementé el front-end y back-end del sistema de reclutamiento. Creé una sub-interfaz que lista todos los tripulantes activos y permite seleccionar hasta tres para una expedición; al confirmar, genera una colección de referencias que se pasan al controlador de misiones para insertar la acción de explorar en la cola de cada NPC. Dejé preparadas las estructuras de datos para incorporar, en un futuro, interacciones de persuasión vía LLM, aunque no llegaron a implementarse en esta versión.

Finalmente, me ocupé del diseño y la lógica de las interfaces restantes:

- Tripulante: panel individual que muestra nombre, imagen, personalidad, hambre y acción en curso, y que dispara el cuadro de comunicación.
- Comunicación con LLM: diálogo que recopila texto del jugador y lanza el proceso de petición–respuesta.
- Inventario, Navegación y Cocina: pantallas sincronizadas con el estado del juego que impiden al jugador cometer acciones inválidas y facilitan la visualización clara de datos.

Para cada pantalla, trabajé la colocación de controles, las animaciones de apertura/cierre y las comprobaciones de estado, garantizando una experiencia libre de errores y muy orientada a la usabilidad.

A través de estas aportaciones, he adquirido un dominio sólido de Unity 2D, patrones de diseño de sistemas de comunicación asíncrona con APIs y gestión compleja de datos en tiempo real. Este proyecto no solo ha reforzado mis habilidades técnicas, sino que me ha permitido experimentar de lleno el potencial de los LLM en el diseño de experiencias interactivas.

Contribuciones conjuntas

El desarrollo de este proyecto ha sido, desde sus primeras fases, un esfuerzo altamente colaborativo. Ambos miembros del equipo hemos estado implicados de forma conjunta en el diseño conceptual del videojuego, tomando decisiones clave relacionadas con las dinámicas de juego, el comportamiento de los NPCs y la estructura funcional de la nave. Desde el inicio, entendimos que una base sólida en la lógica general del sistema era imprescindible para permitir una implementación escalable y coherente. Por ello, el diseño del sistema de juego —incluyendo las instalaciones disponibles, las mecánicas de acción, los sistemas de progresión y la integración con inteligencia artificial— fue consensuado y construido entre ambos, a través de numerosas sesiones de planificación, discusión de ideas y validación de propuestas.

Uno de los principales desarrollos compartidos fue la creación de la nave inicial, tanto a nivel visual como estructural. El diseño del espacio caminable, la distribución de las instalaciones y el planteamiento lógico de las rutas y accesos fue elaborado en conjunto, buscando no solo una disposición estética coherente, sino también una funcionalidad clara para las futuras interacciones con los tripulantes. Este entorno representaba la base del ecosistema jugable, por lo que ambos participamos activamente en su construcción, pruebas y ajustes.

Otro aspecto clave fue el desarrollo del movimiento de los tripulantes hacia instalaciones específicas. Esta mecánica es esencial en el flujo del juego, ya que representa el momento en el que las decisiones del jugador —transmitidas a través del LLM— se transforman en acciones visibles y tangibles dentro del entorno. La lógica detrás de este comportamiento fue diseñada e implementada colaborativamente. Se buscó que el sistema fuera escalable, eficiente y resistente a errores en situaciones de múltiples instancias simultáneas.

Además, colaboramos estrechamente en la implementación de los efectos de las acciones que los tripulantes ejecutan al llegar a una instalación. Esta parte del desarrollo fue especialmente crítica, ya que suponía conectar la representación visual y la lógica interactiva. Acciones como comer, investigar, fabricar o repostar combustible debían tener consecuencias directas en las variables del sistema —como los niveles de hambre, los recursos disponibles o el estado de las investigaciones—. Para ello, diseñamos un sistema de efectos parametrizados, aplicados en función del tipo de acción y los datos recibidos, sobre el cual ambos trabajamos en paralelo para asegurar consistencia y robustez.

Finalmente, todo el proceso de mejora en la comunicación con el LLM y la posterior conversión de sus respuestas a efectos en el juego fue un esfuerzo conjunto. Desde las primeras pruebas hasta las versiones más completas de la petición y del captador de respuesta, se realizaron múltiples iteraciones compartidas para optimizar la estructura del mensaje, mejorar la calidad de las respuestas, y convertir estas de forma segura en acciones válidas dentro del sistema. Esta parte integraba la IA con el motor de juego, por lo que su diseño y desarrollo fue especialmente delicado, y requirió constante validación por parte de ambos.

En resumen, aunque la distribución de tareas individuales permitió avanzar en paralelo en distintas áreas del desarrollo, fue en estas tareas compartidas —las que sustentan el funcionamiento general del juego y la experiencia interactiva— donde más se reflejó el trabajo en equipo, y que más contribuyeron al éxito del proyecto.

8 Manual de descarga y ejecución

Este manual detalla el proceso necesario para descargar y ejecutar el videojuego de este TFG.

Requisitos previos:

- Unity instalado en su versión 2022.3.46f1.
- Cuenta de Google con acceso a la Generative Langua Api de gemini encontrada en este enlace https://makersuite.google.com/app
- Tarjeta de crédito o débito (requerida por Google para activar el cobro por el uso de la API aunque se otorgan créditos gratis).

Descarga del proyecto

1. Accede al repositorio del proyecto y descarga el código:
    - GitHub: https://github.com/TGF-2024-25/sims

2. Extrae el contenido a una carpeta en tu equipo.

3. Abre Unity Hub y selecciona "*Add*" y después "*Add proyect from disk*" y seleccione la ruta donde había descargado el proyecto.
4. Configuración de API de Gemini.
    4.1 Conseguir la clave para la API.
        - Ve a https://makersuite.google.com/app
        - Inicia sesión con tu cuenta de Google.
        - Activa la Generative Language API generando una clave personal.
        - Copia tu clave API personal.
    4.2 Insertar la clave en el proyecto.
        - En la carpeta del proyecto acceda a Assets/Resources/.

- Crea un archivo llamado key.json con el siguiente contenido:
  { "key": "TU_API_KEY_AQUÍ" }

Ejecución del proyecto

5.2 Abra el proyecto en Unity.

5.3 Haga click en el botón de play en la parte superior de la interfaz de Unity.

Tutorial de uso

A fin de hacer más entendible la jugabilidad, se ha editado y se adjunta un video a modo de manual que cubre las principales características y posibilidades en el juego, a la vez que explica las funcionalidades de todos los elementos disponibles y su efecto en la partida.

Los Sims (LLM) TFG UCM FDI : https://youtu.be/DPPjBwX4AE8

9 Bibliografía

[1] Wikipedia, «Modelo extenso de lenguaje,» n.d. [En línea]. [Último acceso: 15 Enero 2025].

[2] Unity, «Unity,» n.d. [En línea]. [Último acceso: 20 Febrero 2025].

[3] Sydle, «Que es una API?,» n.d. [En línea]. Available: https://www.sydle.com/es/blog/api-6214f68876950e47761c40e7. [Último acceso: 20 Enero 2025].

[4] Google, «Gemini,» n.d. [En línea]. Available: https://gemini.google.com/?hl=es-ES. [Último acceso: 20 Enero 2025].

[5] json.org, «Introducción a JSON,» n.d. [En línea]. Available: https://www.json.org/json-es.html. [Último acceso: 15 Febrero 2025].

[6] K. Schwaber, «La guía Scrum,» 2020. [En línea]. Available: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf. [Último acceso: 7 Enero 2025].

[7] Atlassian, «Jira,» n.d. [En línea]. Available: https://www.atlassian.com/es/software/jira. [Último acceso: 15 Enero 2025].

[8] A. Edu, «Manual de UML,» n.d. [En línea]. Available: https://www.academia.edu/31883917/Manual_de_UML_Paul_Kimmel. [Último acceso: 20 Marzo 2025].

[9] O. AI, «Chat GPT,» n.d. [En línea]. Available: https://chatgpt.com/. [Último acceso: 10 Enero 2025].

[10] Arsys, «Que es bard?,» n.d. [En línea]. Available: https://www.arsys.es/blog/google-bard-que-es-y-como-funciona. [Último acceso: 10 Enero 2025].

[11] Meta, «Llama,» n.d. [En línea]. Available: https://www.llama.com/. [Último acceso: 10 Enero 2025].

[12] TheGoodGamer, «El papel de la IA en videjuegos,» n.d. [En línea]. Available: https://thegoodgamer.es/el-papel-de-la-inteligencia-artificial-en-los-videojuegos/. [Último acceso: 15 Enero 2025].

[13] QNAP, «Máquina de estados finitos,» n.d. [En línea]. Available: https://www.profesionalreview.com/2022/11/26/maquinas-de-estado-finito-que-son-para-que-sirven/. [Último acceso: 16 Febrero 2025].

[14] LucidChart, «Árbol de decisión,» n.d. [En línea]. Available: https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-arbol-de-decision. [Último acceso: 10 Febrero 2025].

[15] E. Games, «The sims official website,» n.d. [En línea]. Available: https://www.ea.com/es-es/games/the-sims. [Último acceso: 30 Enero 2025].

[16] latitude, «AI Dungeon,» n.d. [En línea]. Available: https://aidungeon.com/. [Último acceso: 25 Enero 2025].

[17] Mojang, «Minecraft Official Website,» n.d. [En línea]. Available: https://www.minecraft.net/es-es. [Último acceso: 2 Febrero 2025].

[18] U. C. d. Madrid, «Trabajos Fin de Grado 2023/2024,» 2024. [En línea]. Available: https://informatica.ucm.es/tfgs-2023-2024. [Último acceso: 15 Enero 2025].

[19] O. AI, «OpenAI Models 3.5,» n.d. [En línea]. Available: https://platform.openai.com/docs/models. [Último acceso: 10 Febrero 2025].

[20] H. Face, «HuggingFace,» n.d. [En línea]. Available: https://huggingface.co/. [Último acceso: 5 Febrero 2025].

[21] E. Games, «Unreal Engine,» n.d. [En línea]. Available: https://www.unrealengine.com/en-US. [Último acceso: 28 Enero 2025].

[22] Y. Games, «Game Maker,» n.d. [En línea]. Available: https://gamemaker.io/en. [Último acceso: 2 Febrero 2025].

[23] Chucklefish, «Starbound,» n.d. [En línea]. Available: https://playstarbound.com/. [Último acceso: 5 Febrero 2025].

[24] L. Studios, «RimWorld,» n.d. [En línea]. Available: https://rimworldgame.com/. [Último acceso: 5 Febrero 2025].