



**Escuela Superior
de Ingeniería y Tecnología**
Universidad de La Laguna

Trabajo de Fin de Grado

**Sistema de puzzles para videojuegos basado
en inteligencia artificial generativa**

*Puzzle system for videogames based on generative artificial
intelligence*

Diego Herrera Mendoza

La Laguna, 3 de diciembre de 2024

D. Jose Ignacio Estévez Damas, con N.I.F. 43.786.097-P profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

“Sistema de puzzles para videojuegos basado en inteligencia artificial generativa”

ha sido realizada bajo su dirección por D. **Diego Herrera Mendoza**, con N.I.F. 43.835.937-F.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firma la presente en La Laguna a 3 de diciembre de 2024

Agradecimientos

A mi tutor, Jose Ignacio Estévez Damas, por su trato respetuoso y su forma de guiar el proyecto.

A mis padres y a mis hermanos, por brindarme no solo los recursos necesarios para alcanzar mis metas, sino también por su ánimo y apoyo emocional.

A mis amigos y compañeros de carrera, con quienes he compartido tardes de estudio y momentos divertidos.

A mi pareja. Su presencia constante, tanto en los buenos como en los malos momentos, y su incondicional apoyo han sido esenciales.

Y a todas las personas que han probado el prototipo y han ofrecido sus valiosos comentarios y sugerencias.

A todos ellos, muchas gracias.

Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.

Resumen

Este trabajo de fin de grado se hace partiendo de la curiosidad y el estudio de los modelos de inteligencia artificial generativa y su uso dentro del campo de los videojuegos. Ya existiendo muchas herramientas que ayudan a la creación de los mismos, se ha optado por tratar de integrar una de estas inteligencias artificiales dentro del propio videojuego, convirtiéndola en una mecánica más que el jugador tendrá que utilizar para poder avanzar y completar el juego. En este proceso se han tenido que valorar decisiones respecto al modelo a utilizar, sopesando sus ventajas y desventajas, así como el lugar en el que se alojaría la inteligencia artificial. Para el desarrollo del videojuego, se ha tomado como mayor referencia el juego Scribblenauts, así como sus secuelas que poseen la misma mecánica principal, en el que se hacen aparecer objetos mediante texto para entregarlos a los personajes que tengan problemas dentro del mundo del videojuego. Los elementos que utiliza Scribblenauts son todos dibujados y clasificados manualmente, pero en este trabajo se propone una implementación en Unity3D que permita la creación de estos objetos mediante la IA generativa DALL-E 3 y la herramienta de Google para clasificar imágenes Cloud Vision API. Aunque más tarde esta herramienta terminó intercambiándose por el modelo de lenguaje de Google Gemini 1.5

Palabras clave: videojuegos, inteligencia artificial generativa, mecánicas, Cloud Vision, DALL-E 3, Gemini 1.5.

Abstract

This final degree project is based on the curiosity and the study of generative artificial intelligence models and their use in the field of video games. Since many tools that help create them already exist, the decision was made to integrate one of these AI models into the video game as a mechanic that players will use to progress and complete the game. In this process we have had to evaluate decisions regarding the model to be used, assessing its advantages and disadvantages, as well as the place where the artificial intelligence would be hosted. For the development of the video game, we have taken as a major reference the game Scribblenauts, as well as its sequels that have the same main mechanics, in which objects appear through text to deliver them to the characters who have problems in the world of the video game. The elements used by Scribblenauts were all drawn and classified manually, but this project proposes an implementation in Unity3D that allows the creation of these objects using the generative AI DALL-E 3 and the Google tool for classifying images Cloud Vision API. However, this tool was later exchanged for the Google Gemini 1.5 language model.

Keywords: video games, generative artificial intelligence, game puzzles, Cloud Vision, DALL-E 3, Gemini 1.5.

Índice general

1. Motivación y objetivos	1
1.1. Motivación del proyecto	1
1.2. Objetivos principales	2
1.3. Objetivos específicos	3
1.4. Material de consulta y organización de la memoria	3
1.4.1. Descripción de capítulos	3
1.4.2. Material de consulta	3
2. Antecedentes y estado del arte	4
2.1. Inteligencia artificial generativa	4
2.2. Inteligencia artificial generativa en videojuegos	6
2.3. Idea e inspiración para el proyecto	7
2.4. Técnicas de generación de imágenes en este proyecto	8
2.4.1. Difusión latente	8
2.4.2. Transformers	9
3. Decisiones tecnológicas	10
3.1. Herramientas utilizadas	10
3.1.1. Unity	10
3.1.2. Librerías y assets	10
3.2. Elección del modelo generativo	14
3.2.1. StableDiffusion	15
3.2.2. ThinkDiffusionXL	16
3.2.3. DALL-E 3	18
3.2.4. Conclusiones	19
3.3. Elección del clasificador	19
3.3.1. Amazon Rekognition	20
3.3.2. Google Vision	21
3.3.3. Conclusiones	22
3.3.4. Alternativa al clasificador	23
4. Desarrollo del proyecto	24
4.1. Flujo común del juego	24
4.2. Aspectos convencionales de juegos 2D	24
4.2.1. Controlador del jugador	24
4.2.2. Diseño de niveles	25
4.2.3. Post-procesado y extras	25
4.3. Integración de inteligencia artificial	26
4.3.1. Creación de objetos	31

4.3.2. Gestión de los tiempos de carga	33
4.4. Diseño de los puzzles	33
4.5. Experiencia del usuario	35
4.6. Problemas encontrados con la integración	35
4.6.1. Estilo	35
4.6.2. Posición lógica	35
5. Evaluación y resultados del proyecto	36
5.1. Procedimiento de evaluación	36
5.2. Resultados de la evaluación	37
5.3. Problemas encontrados	43
6. Conclusiones y líneas futuras	45
7. Conclusions and future work	46
8. Presupuesto	47
8.1. Costes del desarrollo	47
8.2. Costes de generación de imágenes	47
8.3. Costes de clasificación de imágenes	48
8.4. Resumen de costes	48

Índice de Figuras

1.1. Inteligencias artificiales generativas usadas en los videojuegos (1)	2
2.1. Comparativa de uso entre ChatGPT y GoogleBard según SimilarWeb (3) . .	5
2.2. Mejora de calidad de código respecto al uso de Copilot Chat según Mario Rodríguez (4)	6
2.3. Captura del videojuego Scribblenauts (7)	8
2.4. Funcionamiento de “difusión latente” explicado por Guodong (Troy) Zhao (8)	8
2.5. Representación de posibilidades de los transformers o “modelos base” (9) .	9
3.1. Diagrama de clases de la librería HFSM	11
3.2. Foto principal del Asset “Mossy Cavern”	12
3.3. Foto principal del Asset “Vector GUI Icons”	13
3.4. Configuración en el proyecto de las rule tiles	14
3.5. Pruebas de Stable Diffusion	16
3.6. Comparativa de ThinkDiffusionXL con un modelo promedio (13)	17
3.7. Pruebas de Think Diffusion	17
3.8. Pruebas de DALL-E 3	18
3.9. Pruebas de Amazon Rekognition	21
3.10 Pruebas de Google Vision	22
4.1. Diagrama de flujo básico del juego	24
4.2. Comparaciones post-procesado	26
4.3. Diagrama de clases para la generación de objetos	27
4.4. Código para la conexión con el modelo de IA generativa	28
4.5. Clases extra para la petición de DALLE	29
4.6. Clases extra para la petición de Google Cloud Vision	29
4.7. Código para la conexión con el clasificador	30
4.8. Diagrama de generación de un objeto dentro del proyecto	32
4.9. Funciones para la generación de objetos dentro del juego	32
4.10 Variables principales de la clase ImageItemGenerator	33
4.11 Recorrido por un puzzle del videojuego	34
5.1. Resultados de la pregunta 1	38
5.2. Resultados de la pregunta 3	38
5.3. Resultados de la pregunta 4	39
5.4. Resultados de la pregunta 5	39
5.5. Resultados de la pregunta 6	40
5.6. Resultados de la pregunta 7	40
5.7. Resultados de la pregunta 8	41
5.8. Resultados de la pregunta 9	41

5.9. Resultados de la pregunta 10	41
5.10 Resultados de la pregunta 11	42
5.11 Resultados de la pregunta 12	42
5.12 Resultados de la pregunta 13	43
5.13 Error en la clasificación	44

Índice de Tablas

3.1. Evaluación final ejemplo	15
3.2. Evaluación final de Stable Diffusion	16
3.3. Evaluación final de Think Diffusion XL	18
3.4. Evaluación final de Think Diffusion XL	19
3.5. Comparación de los modelos	19
3.6. Evaluación final ejemplo	20
3.7. Evaluación final de Think Diffusion XL	21
3.8. Evaluación final de Think Diffusion XL	22
3.9. Comparación de los clasificadores	23
5.1. Preguntas de la sección 1	36
5.2. Preguntas de la sección 2	37
5.3. Preguntas de la sección 4	37
5.4. Preguntas de la sección 5	37
8.1. Resumen de costos	47
8.2. Resumen de costos	48

Capítulo 1

Motivación y objetivos

1.1. Motivación del proyecto

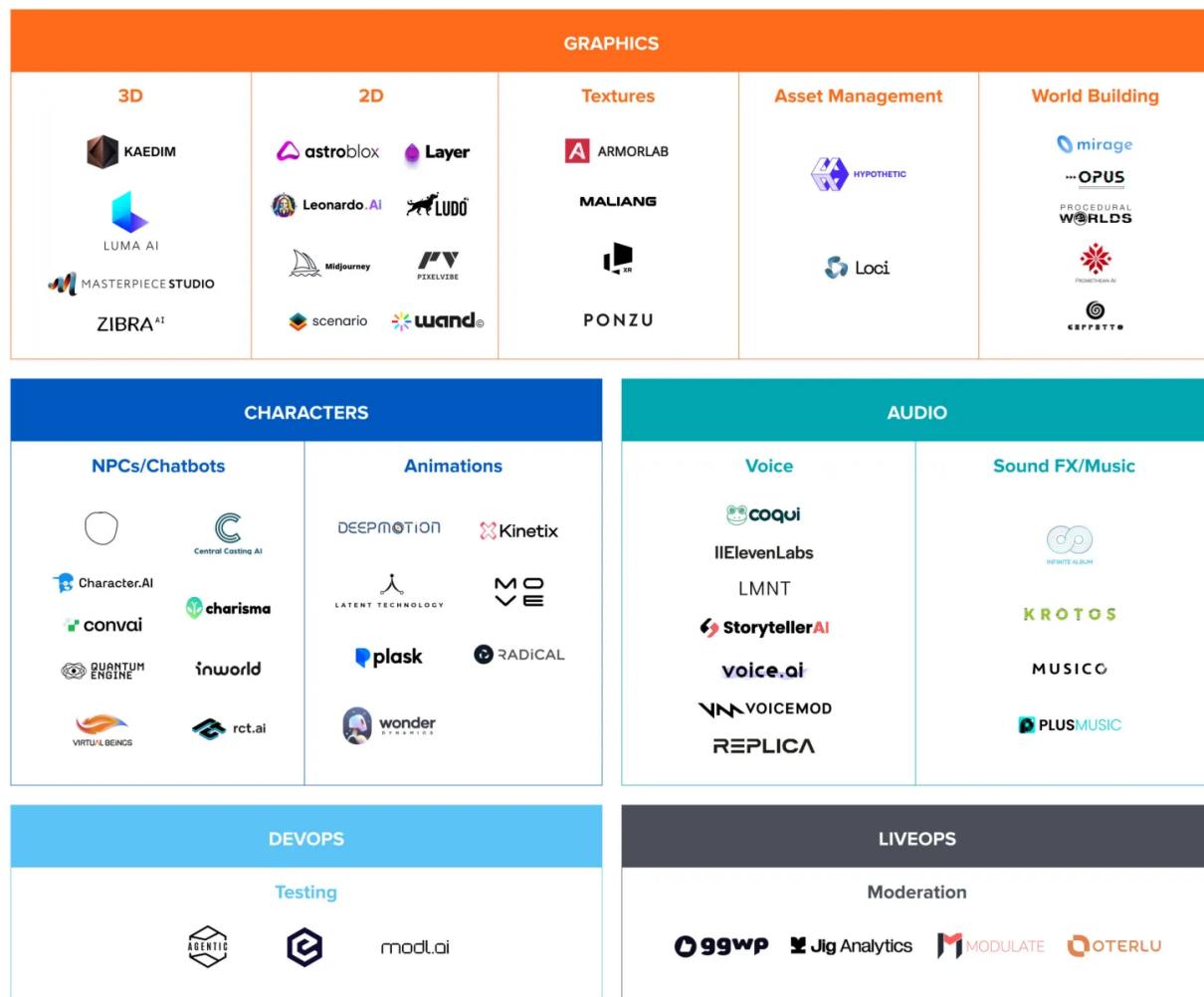
Este trabajo está motivado por los últimos avances en inteligencia artificial dentro del ámbito de los videojuegos. En particular, nos hemos enfocado en la integración de modelos de inteligencia artificial generativa en la jugabilidad, intentando comprobar si existe una capacidad en estas tecnologías para enriquecer y transformar la experiencia del jugador.

En la actualidad, los videojuegos, en su gran mayoría, solo utilizan las inteligencias artificiales generativas como herramientas durante el desarrollo, y no las vemos integradas a modo de mecánica o en algún apartado dentro del propio juego con el que el jugador pueda interactuar. Esto se puede ver en la figura 1.1 en la que tenemos un amplio catálogo de IAs generativas para distintos ámbitos.

Con el boom de esta tecnología ahora están apareciendo algunos videojuegos que introducen inteligencias artificiales generativas como mecánica principal, y algunos de ellos están ganando cierta popularidad dentro del sector. Los más destacados suelen hacer uso de modelos conversacionales para dar vida a los personajes dentro del juego y así abrir un árbol de posibilidades muy amplio.

Es este pequeño paso el que ha servido de inspiración para tratar de investigar más a fondo otras implementaciones de estas tecnologías dentro del mundo de los videojuegos.

Generative AI for Games Market Map



Updated: March 2023
 Graphics provided herein are for informational purposes only and should not be relied upon when making any investment decision.



Figura 1.1: Inteligencias artificiales generativas usadas en los videojuegos (1)

1.2. Objetivos principales

El objetivo principal de este trabajo de fin de grado es investigar las inteligencias artificiales generativas y su posible uso dentro de los videojuegos. Para ello se tendrá que indagar primeramente sobre qué uso se le dará en el trabajo, es decir, si se le dará uso de herramienta auxiliar o tendrá un papel más protagonista dentro del videojuego. Una vez determinado esto, el objetivo será verificar los requisitos mínimos que se requieren para su uso, como pueden ser:

- Creación o búsqueda del modelo de IA generativa
- Velocidad de procesamiento
- Coherencia de los resultados
- Alojamiento del modelo

1.3. Objetivos específicos

El campo del estudio del TFG es la integración de una IA generativa en el campo de los videojuegos. En este caso, para formar parte de las mecánicas jugables. En este TFG, los objetivos específicos para conseguir ese cometido son los siguientes:

- Desarrollar o encontrar un modelo de IA generativa capaz de generar los recursos necesarios que sean utilizables dentro del juego.
- Integrarla en el videojuego de manera que el jugador pueda usar los resultados de la IA generativa en alguna mecánica del juego.
- Evaluar las ventajas e inconvenientes de esta clase de IA dentro del campo de los videojuegos, así como el hecho de usarlas como parte de las mecánicas.
- Analizar los resultados del videojuego mediante la prueba de usuarios y sus valoraciones personales.

1.4. Material de consulta y organización de la memoria

1.4.1. Descripción de capítulos

- **Capítulo 2 - Antecedentes y estado del arte:** presentación de la actualidad de la inteligencia artificial generativa en el mundo de los videojuegos, así como la principal idea del proyecto.
- **Capítulo 3 - Decisiones tecnológicas:** enumeración de las herramientas utilizadas, realizando comparaciones y razonando decisiones específicas.
- **Capítulo 4 - Desarrollo del proyecto:** descripción de todo el desarrollo del proyecto, desde las partes más básicas hasta las más complejas.
- **Capítulo 5 - Resultados y evaluación del proyecto:** análisis de los resultados de la prueba del proyecto por parte de terceros.
- **Capítulo 6/7 - Conclusiones y líneas futuras**
- **Capítulo 8 - Presupuesto:** evaluación de gastos de las herramientas utilizadas para el proyecto.

1.4.2. Material de consulta

El repositorio del proyecto (2) se encuentra en el apartado de la bibliografía de esta memoria.

Capítulo 2

Antecedentes y estado del arte

2.1. Inteligencia artificial generativa

Las IAs (Inteligencias artificiales) generativas no son más que el fruto del campo del aprendizaje automático, que siempre ha utilizado modelos estadísticos y generativos para modelar y predecir datos.

A finales de los 2000, el surgimiento del aprendizaje profundo (Deep Learning) impulsó el progreso y la investigación en el procesamiento de imágenes y vídeos, así como el análisis del lenguaje natural y otras numerosas tareas. Los motivos de este surgimiento fueron básicamente 2 razones: mejora en la potencia de computo y el acceso a enormes cantidades de datos de diferente naturaleza gracias a internet.

Un sistema de inteligencia artificial generativa se construye a partir de un aprendizaje automático, ya sea supervisado o no, a un conjunto de datos específico. Las capacidades de estas IAs dependen de la modalidad o el tipo de información que se le ha asignado. Existen las IAs unimodales, que aceptan únicamente un tipo de entrada, así como existen las IAs multimodales, que pueden tomar varios tipos distintos de datos como su entrada. Dentro de estas modalidades, se encuentran las siguientes:

- **Texto:** Los sistemas de IA generativa centrados en reconocer y generar lenguaje natural, entrenados a partir de palabras o tokens de palabras incluyen a Chat-GPT o Google Bard entre otros muchos existentes. Mucha gente tiende a comparar este tipo de modelos como se puede ver en la figura 2.1 ya que su finalidad suele ser la misma.

ChatGPT is dominating Google Bard

ChatGPT (chat.openai.com) vs. Google Bard (bard.google.com)

ChatGPT was launched in Nov-22 while Google Bard removed waitlist and launched publicly on May 10, 2023

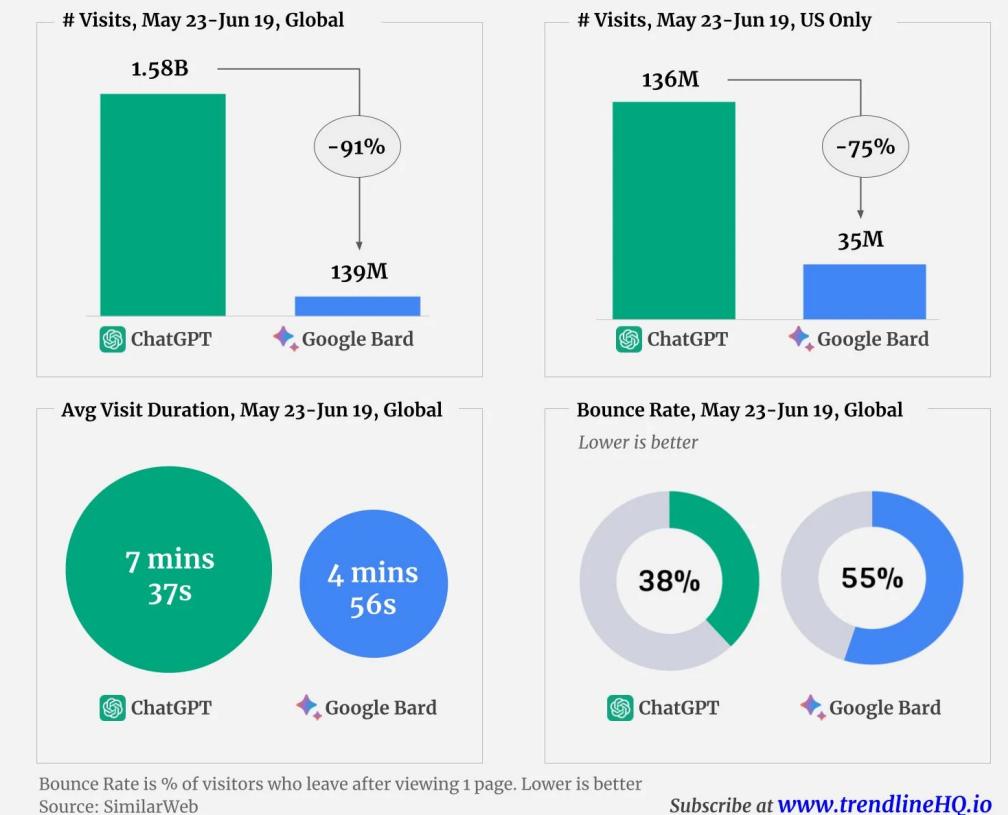


Figura 2.1: Comparativa de uso entre ChatGPT y GoogleBard según SimilarWeb (3)

- **Código:** Además de texto en lenguaje natural, también hay inteligencias artificiales capaces de generar código fuente para nuevos programas. Estas se entrena con código fuente del lenguaje de programación del que se quiera conseguir resultados. La más popular y controvertida actualmente es Github Copilot, y su versión de chat está demostrando ser beneficiosa para el código como se ve en la figura 2.2.

Using GitHub Copilot Chat correlates with better code quality

85% of developers felt more confident in their code quality when authoring code with GitHub Copilot and GitHub Copilot Chat

85%



Code reviews were more actionable and completed 15% faster than without GitHub Copilot Chat

15%



88% of developers reported maintaining flow state with GitHub Copilot Chat

88%



Figura 2.2: Mejora de calidad de código respecto al uso de Copilot Chat según Mario Rodríguez (4)

- **Imágenes:** Los sistemas de inteligencia artificial entrenados con conjuntos de imágenes se utilizan comúnmente junto al análisis del lenguaje natural para crear imágenes a partir de texto. Los ejemplos más populares de estas IAs son DALL-E y Stable Diffusion que ya están utilizándose incluso con fines médicos según este artículo (5).

2.2. Inteligencia artificial generativa en videojuegos

Los resultados de una inteligencia artificial generativa, tal y como ya se está haciendo, se pueden usar como herramientas en la creación de videojuegos. Se podría dividir el uso de estas IAs generativas en tres categorías de uso dentro de los videojuegos, siendo estas:

- **Creación de “assets”:** Las inteligencias artificiales generativas han dado muchos resultados en cuanto a la creación de “assets” se refiere. Como se mencionaba anteriormente, estas IAs generativas son capaces de crear multitud de elementos de distintas naturalezas. Para acelerar el trabajo de los diseñadores, se está optando por utilizar esta generación para crear componentes como: mallas, animaciones o audio.
- **Aspectos técnicos del juego:** En cuanto al funcionamiento específico del juego, están apareciendo tecnologías que ayudan a mejorar el rendimiento del mismo utilizando inteligencia artificial generativa. El ejemplo más popular es en el proceso de renderizado de gráficos, en el que se está utilizando la IA para generar los fotogramas del videojuego y así aligerar la carga de la propia GPU. La empresa pionera en implementar esta tecnología fue Nvidia con su DLSS (Deep Learning Super Sampling) (6).

- **Funcionamiento del juego:** Últimamente, con el boom de las inteligencias artificiales generativas, están apareciendo videojuegos que hacen uso de las mismas de una manera menos convencional: integrándola dentro del propio videojuego como parte del mismo. Esto está ocurriendo ya que no se encuentran demasiadas dificultades a la hora de integrar los resultados que da una IA generativa en un videojuego siempre y cuando las lo produzca en el formato esperado. Lo más popular a día de hoy es hacer crear diálogo con personajes en tiempo real, ya que es sencillo, efectivo y vistoso. Pero cuanto más complejo sea el recurso que se requiere, más difícil es conseguir un resultado acorde a lo esperado. Este proyecto se basa mayormente en este último uso de la IA generativa dentro de los videojuegos.

2.3. Idea e inspiración para el proyecto

Antes de empezar el proyecto se valoraron distintas ideas iniciales. Estas ideas fueron evaluadas en su día y algunas incluso llegaron a tener sus propias pruebas, pero no todas llegaron a cumplir. Había ideas que no eran posibles a corto plazo, otras que no daban rienda suelta a una experiencia de juego cómoda, o incluso algunas que no se correspondían bien con los objetivos del proyecto.

Todo el conjunto de ideas terminó evolucionando en la que se ha trabajado para este proyecto. Un juego de puzzles 2D en el que se tiene que crear el objeto adecuado a las circunstancias para poder completar cada puzzle.

La principal inspiración para este videojuego fue Scribblenauts, un videojuego desarrollado por 5th Cell y publicado por Warner Bros. Interactive Entertainment, lanzado inicialmente en 2009 para la consola Nintendo DS. Este videojuego destaca por su mecánica que desafía la creatividad del jugador al permitirle escribir cualquier objeto que pueda imaginar para resolver los diversos problemas y desafíos presentados en el juego. Se puede encontrar una captura del videojuego en la figura 2.3.

La mecánica central de Scribblenauts consiste en que el jugador interactúa con el entorno del juego escribiendo palabras en un cuadro de texto para hacer aparecer objetos y/o personajes que luego pueden interactuar entre sí para resolver los problemas presentados en cada nivel. El juego utiliza un extenso diccionario interno que permite reconocer miles de palabras y convertirlas en elementos jugables en tiempo real.

La idea de este proyecto es intercambiar ese extenso diccionario y dibujos hechos a mano por una inteligencia artificial generativa para crear los objetos y un clasificador para tener un apoyo sobre el que hacer las comprobaciones pertinentes.



Figura 2.3: Captura del videojuego Scribblenauts (7)

2.4. Técnicas de generación de imágenes en este proyecto

2.4.1. Difusión latente

Esta tecnología comienza por un proceso de refinamiento iterativo que se basa en una imagen de ruido aleatorio. El modelo ajusta los píxeles gradualmente a lo largo de varias iteraciones a medida que procesa la imagen, mejorando los detalles y reduciendo el ruido hasta que la imagen final se ajuste a la descripción textual proporcionada. Un diagrama simple de este proceso se puede observar en la figura 2.4.

Todo el proceso generativo se lleva a cabo en varias etapas, cada una de las cuales se enfoca en mejorar ciertos aspectos de la imagen, como la claridad, los detalles y la coherencia con el texto original.

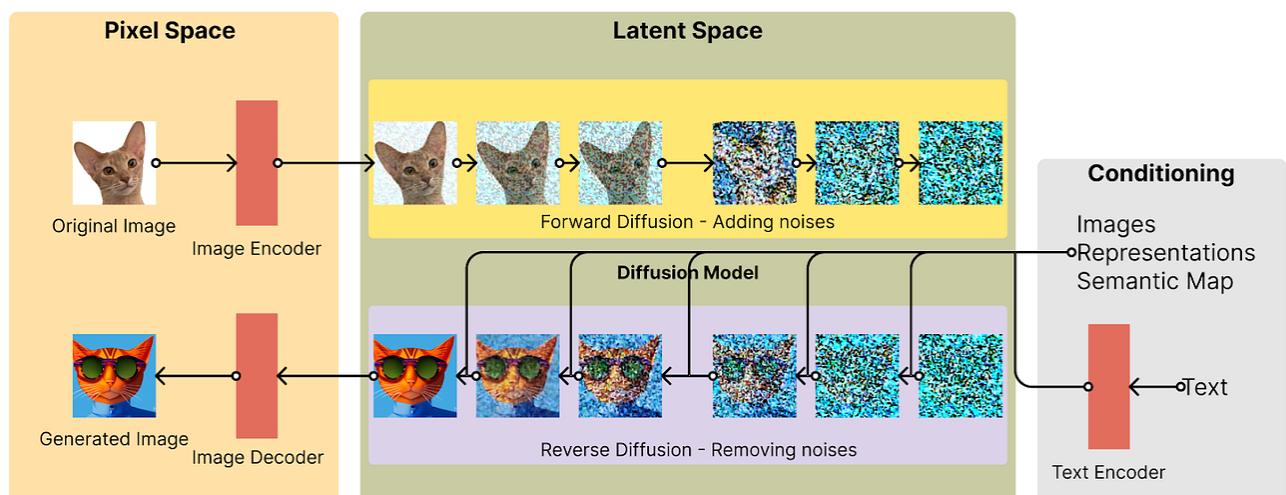


Figura 2.4: Funcionamiento de “difusión latente” explicado por Guodong (Troy) Zhao (8)

2.4.2. Transformers

Los transformers son una arquitectura de red neuronal que ha cambiado significativamente el procesamiento del lenguaje natural y la generación de contenido visual. Esta arquitectura fue introducida en 2017, y utiliza mecanismos de atención para procesar y detectar dependencias en secuencias de datos, lo que los hace especialmente adecuados para tareas como la generación de imágenes a partir de descripciones textuales.

La clave de los transformers radica en el mecanismo de atención, específicamente la auto-atención, que permite al modelo enfocarse en diferentes partes de la entrada para generar la salida correspondiente. Este mecanismo es también lo que permite que esta técnica pueda ser utilizada para multitud de tareas, como se puede observar en la figura 2.5.

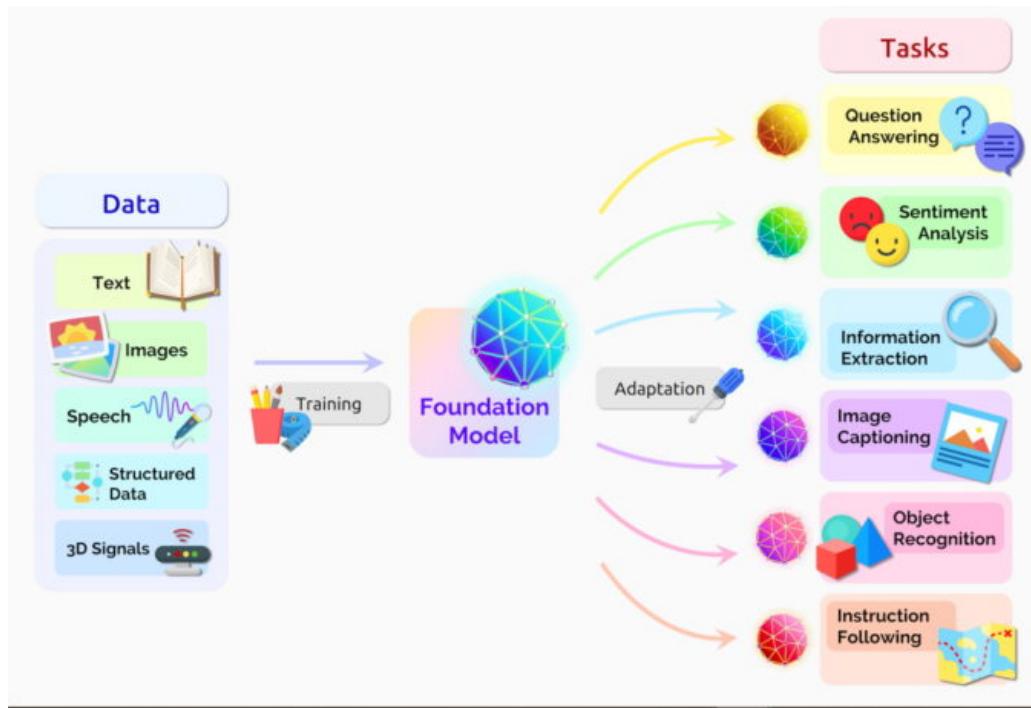


Figura 2.5: Representación de posibilidades de los transformers o “modelos base” (9)

Capítulo 3

Decisiones tecnológicas

3.1. Herramientas utilizadas

3.1.1. Unity

Unity (10) es una plataforma de desarrollo de videojuegos ampliamente conocida por los desarrolladores. Esta herramienta fue seleccionada para el desarrollo de nuestro proyecto debido a una variedad de razones, entre las que se encuentran:

- **Experiencia previa:** De todos las plataformas de desarrollo de videojuego reconocidas, Unity era con la única con la que se había trabajado en profundidad, lo que da pie a centrarse más en la investigación y tener mucha menos carga en la parte del desarrollo.
- **Motor gráfico competente:** Unity ofrece un motor gráfico 2D/3D robusto y ampliamente probado, que a día de hoy sigue recibiendo actualizaciones y soporte. Es un motor con una larga historia en el desarrollo de juegos 2D, que fue extendiendo su alcance hasta llegar al motor versátil y multiplataforma que se conoce hoy en día. Esa versatilidad sobre la decisión del tipo de juego al que orientar el proyecto es una razón clave para utilizarlo.
- **Extensible y personalizable:** Unity permite la integración de diversas bibliotecas y funcionalidades creadas por la comunidad. En particular, se han usado librerías que facilitan tareas tediosas que consumirían mucho tiempo si se hicieran desde cero. A parte de esto, también se tiene en cuenta la gran cantidad de información y tutoriales que la propia comunidad ha creado, ya que hace el trabajo desconocido mucho más ameno.

3.1.2. Librerías y assets

Unity HFSM

“Unity Hierarchical state machine” es una librería que, tal y como expresa su nombre, implementa máquinas de estados jerárquicas en Unity. Esta librería proporciona una implementación simple a la vez de poderosa, puesto que es muy escalable y está completamente basada en clases. En la figura 3.1 se puede observar un diagrama básico de sus clases.

Las razón principal para usar esta librería fue que, para desarrollar las mismas facilidades que tiene esta librería a la hora de crear maquinas de estados habría sido un gasto de tiempo considerable.

En este proyecto se ha utilizado para la lógica de los personajes no jugables, así como para el movimiento de todos los personajes dentro del videojuego.

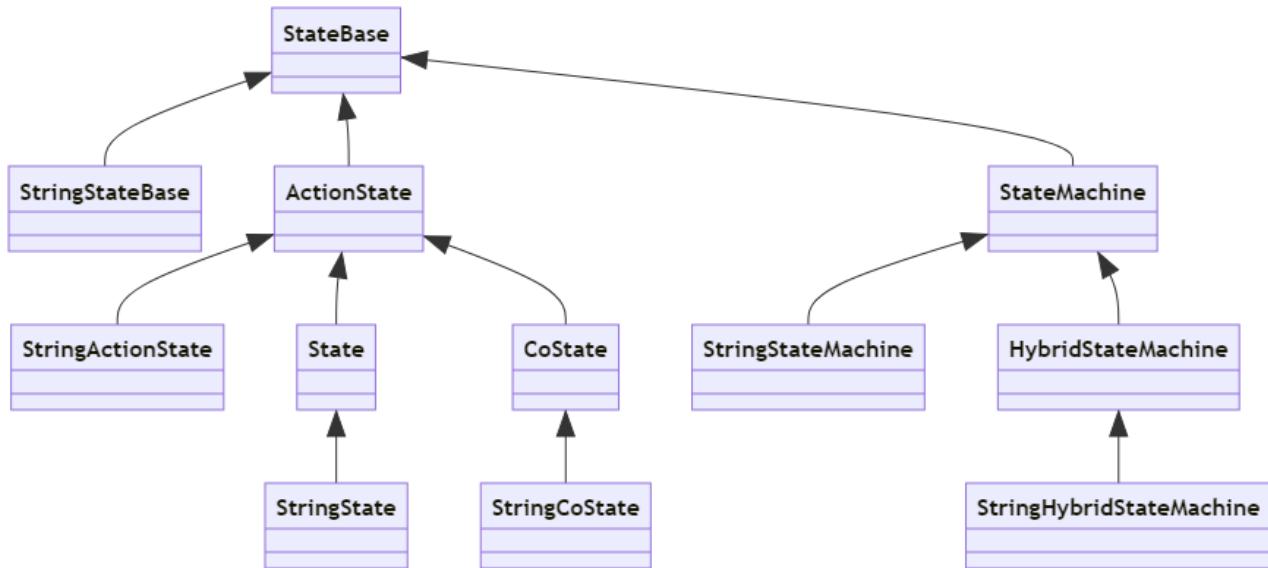


Figura 3.1: Diagrama de clases de la librería HFSM

Mossy Cavern

Mossy Cavern es el asset principal utilizado para el apartado estético del videojuego. El paquete es gratuito y ha sido creado por el usuario de itch.io “maaot”. Este paquete contiene:

- Tileset preparado para Autotiling
- Personaje con animaciones básicas
- Decoraciones para mejorar la escena y hacerla más orgánica. Incluye elementos como plantas, colinas, pequeños arboles...
- Animaciones para algunas de las plantas decorativas
- Dos slimes con animaciones propias

Un ejemplo de los elementos que incluye este paquete se puede observar en la figura 3.2. Para este proyecto se han utilizado únicamente los cuatro primeros elementos listados anteriormente.

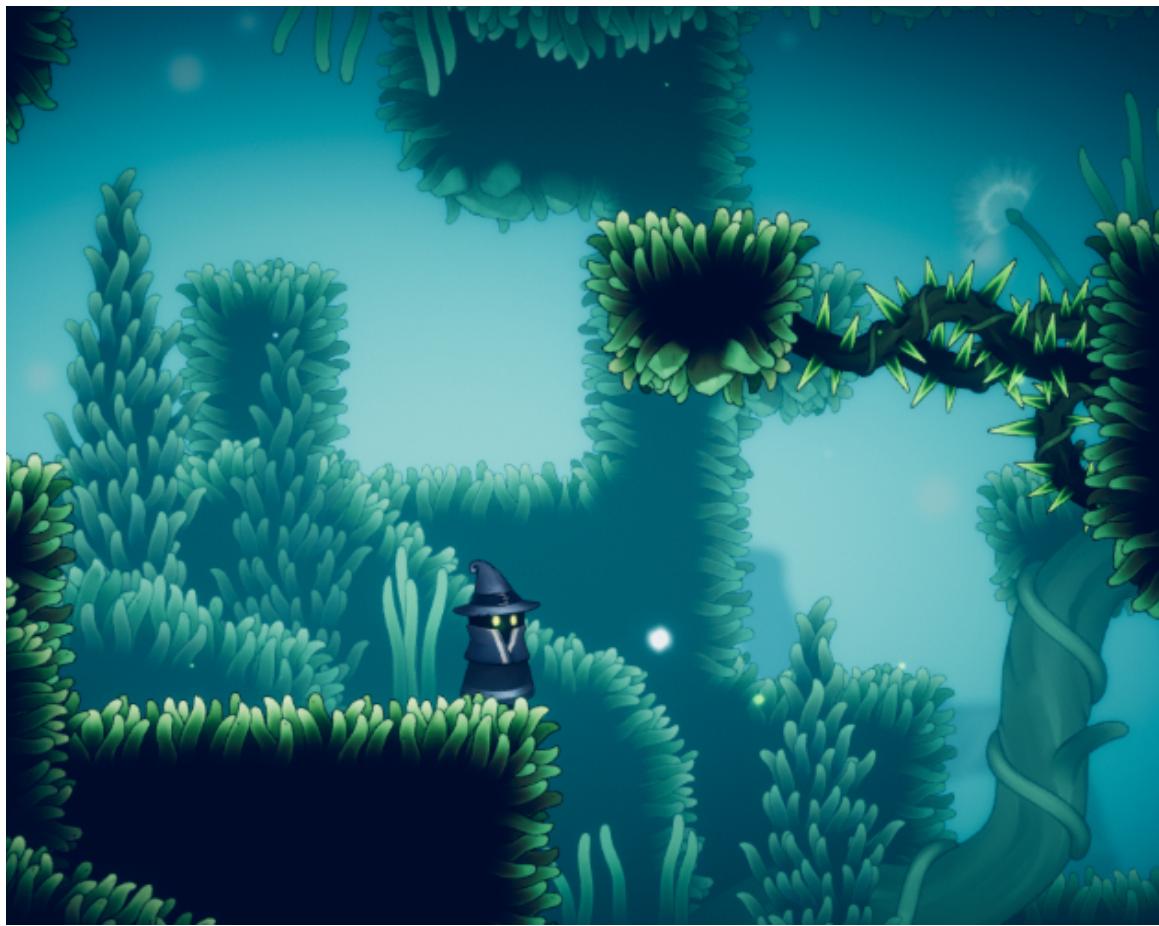


Figura 3.2: Foto principal del Asset “Mossy Cavern”

Vector GUI Icons

Vector GUI Icons se ha usado como asset secundario para la interfaz de usuario del videojuego. Este paquete de iconos también es gratuito y ha sido creado por el usuario “Penzilla” en itch.io.

Este paquete contiene 87 elementos únicos que serán de gran ayuda para crear las interfaces con las que tendrá que interactuar el usuario en su partida. Un ejemplo de los iconos de este paquete se puede ver en la figura 3.3

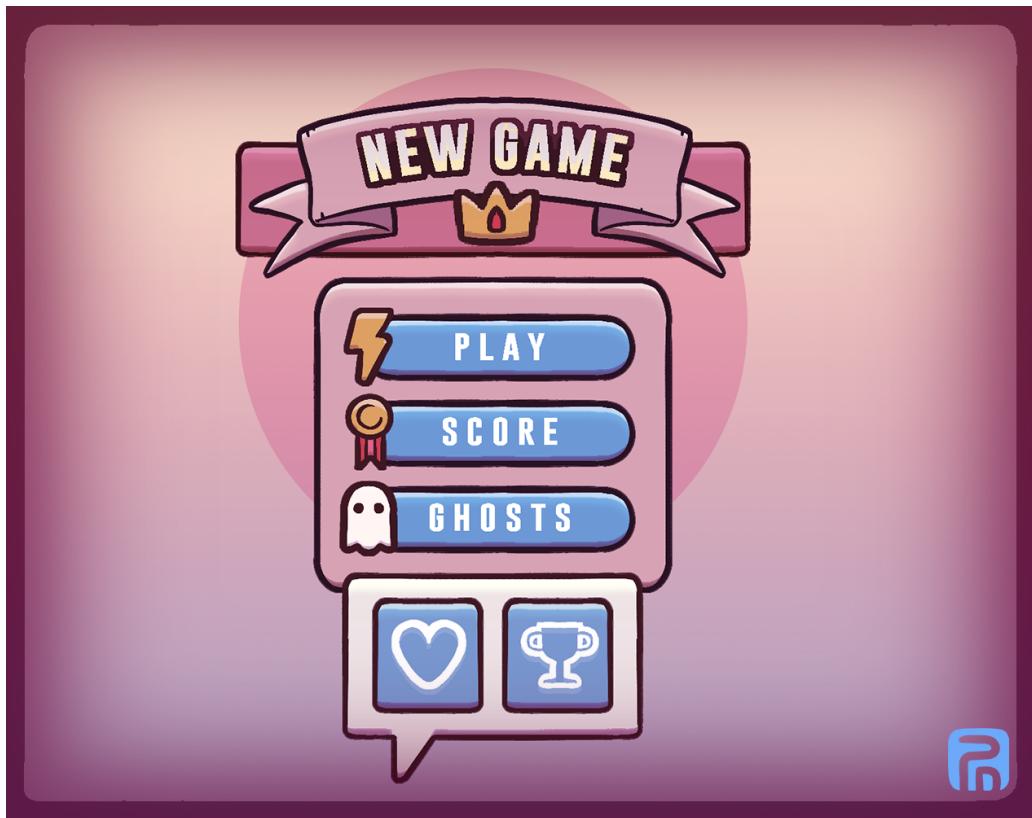


Figura 3.3: Foto principal del Asset “Vector GUI Icons”

Tilemap Auto Rule Tile

TileMap Auto Rule Tile es una herramienta diseñada para acelerar el flujo de trabajo al importar y configurar nuevos Tilemaps utilizando Rule Tiles en Unity. Facilita la creación y configuración automática de Rule Tiles, permitiendo así centrarse en el diseño y la creación de sus niveles sin tener que realizar la configuración manual.

En este proyecto la herramienta se ha utilizado con el tileset antes mencionado y parte del resultado de las reglas se puede observar en la figura 3.4.

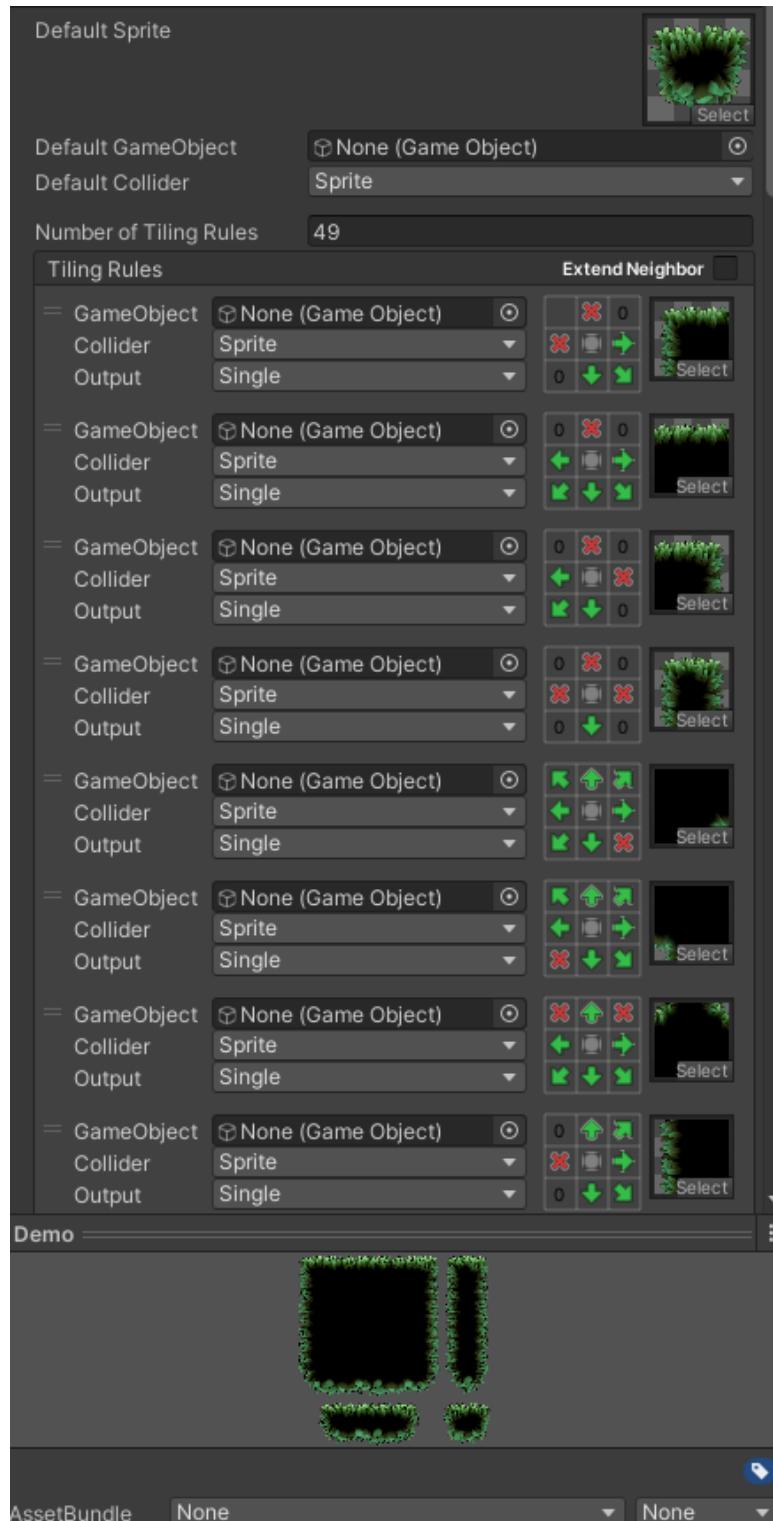


Figura 3.4: Configuración en el proyecto de las rule tiles

3.2. Elección del modelo generativo

Para elegir los modelos generativos se han realizado varias pruebas de generación de imágenes. Estas pruebas se diferenciaban en tres niveles de dificultad.

- El primer nivel era el más sencillo, puesto que se le pedía dibujar elementos muy convencionales como podía ser un plátano, una espada o una simple caja de cartón.

Esta prueba sirve de primera toma de contacto para ver si el modelo que estamos analizando funciona con objetos, ya que hay modelos que están preparados para rostros o cuerpos humanos o algún otro ámbito.

- En el segundo nivel se le preguntaban por elementos más específicos. Estas pruebas podían ser: un ordenador, un paraguas abierto, una pistola de agua o un plátano verde. Con esta prueba, afianzamos el abanico de objetos al que podemos acceder y comprobamos que la inteligencia artificial generativa puede servir para nuestro propósito, al menos de manera provisional
- Por último, tenemos las pruebas difíciles, en las que por lo general todos los modelos fallaban. Estas últimas pruebas se centraban en intentar pedir elementos poco convencionales, como podrían ser conceptos o elementos inexistentes. Unos ejemplos de lo que se pedía en estas pruebas eran: un tomate azul, una hamburguesa verde, un helado de tornillos o un jet-pack.

Las pruebas se realizarían a todos los modelos por igual. Independientemente de que fallasen en las primeras pruebas, se realizarían todas las demás. Se ha seguido esta metodología dado que se ha observado que los modelos pueden tener fallos puntuales, y al descartarlos por esa razón se perdería un posible modelo a utilizar.

Se ha evaluado cada modelo con la tabla de criterios 3.1. En esta tabla se enumeran cada uno de los criterios evaluados y se da una pequeña explicación de qué es lo que se evalúa en los mismos. Todos los criterios se evaluarán del 0 al 10, siendo el 0 la mínima nota y 10 la máxima.

Criterio	Puntuación
Facilidad de uso	API (Application Programming Interface) “User friendly”, servicios en la nube y precio
Versatilidad	Capacidad del modelo para crear elementos distintos
Rapidez	Velocidad del modelo a la hora de generar imágenes
Consistencia	Capacidad del modelo para mantener los resultados entre varias imágenes con la misma descripción sin fallar
Coherencia	Capacidad del modelo para crear elementos que concuerden con la descripción dada

Tabla 3.1: Evaluación final ejemplo

3.2.1. StableDiffusion

Descripción general

Stable Diffusion (11) es un modelo de inteligencia artificial generativa desarrollado por Stability AI, y ha sido diseñado específicamente para crear imágenes a partir de descripciones textuales.

Stable Diffusion se utiliza ampliamente en una variedad de industrias, incluyendo el entretenimiento, la publicidad, y la educación, gracias a su capacidad para producir imágenes de alta calidad y fieles a las descripciones proporcionadas. La facilidad con la que se pueden generar imágenes detalladas a partir de texto ha hecho que esta herramienta sea un perfecto candidato para este proyecto

Resultados en pruebas



(a) Prueba fácil



(b) Prueba media



(c) Prueba difícil

Figura 3.5: Pruebas de Stable Diffusion

En los resultados de las pruebas de la figura 3.5 se puede observar como Stable Diffusion es capaz de crear objetos fácilmente, a la vez que demuestra la versatilidad necesaria para nuestro proyecto. El problema principal que tiene este modelo es la gran inconsistencia seguida de incoherencia que presenta. Esto genera una gran dificultad a la hora de clasificar los objetos generados y podría influir mucho en la experiencia de juego.

Tabla de criterios de evaluación

Criterio	Puntuación
Facilidad de uso	9
Versatilidad	8
Rapidez	9
Consistencia	2
Coherencia	3

Tabla 3.2: Evaluación final de Stable Diffusion

3.2.2. ThinkDiffusionXL

Descripción general

ThinkDiffusionXL (12) fue desarrollado por un equipo de investigadores enfocados en la innovación visual. Al igual que Stable Diffusion, Think Diffusion utiliza “difusión latente” para transformar texto en imágenes detalladas y coherentes, proporcionando una herramienta poderosa para artistas, diseñadores y creadores de contenido. Este modelo es particularmente útil para este proyecto, ya que se busca una solución eficiente y efectiva para generar imágenes de alta calidad que reflejen fielmente las descripciones proporcionadas, y esto es justo lo que prometen en la comparativa que se muestra en la 3.6.

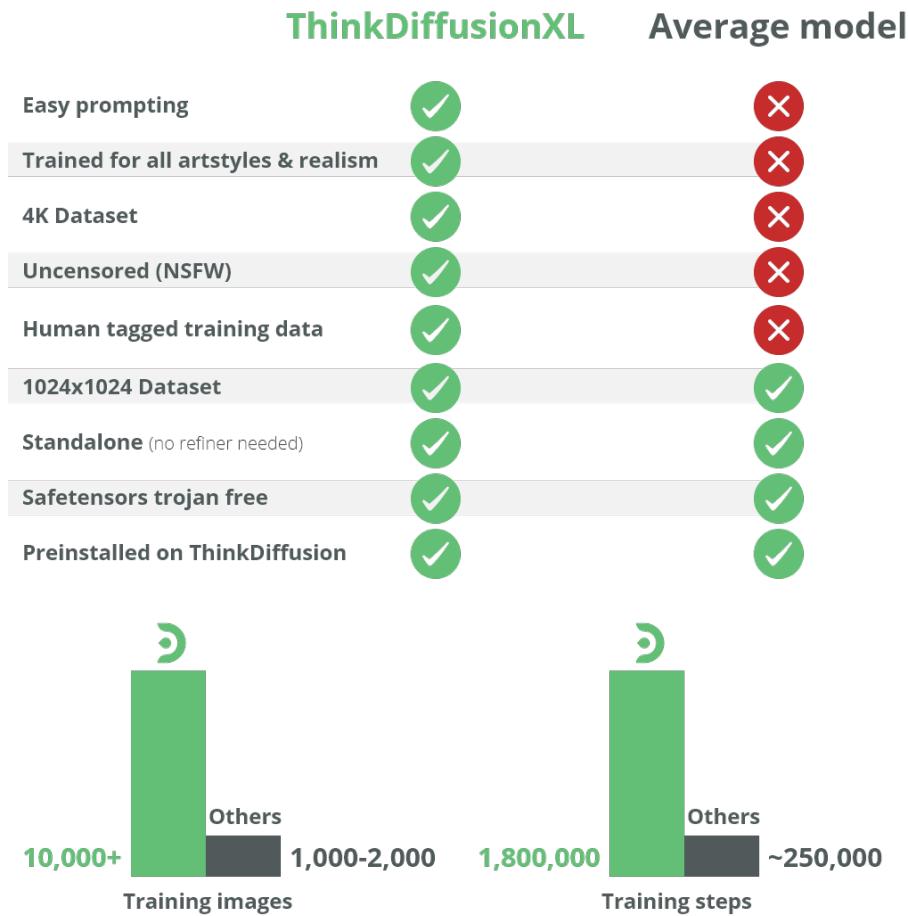


Figura 3.6: Comparativa de ThinkDiffusionXL con un modelo promedio (13)

Resultados en pruebas



(a) Prueba fácil



(b) Prueba media



(c) Prueba difícil

Figura 3.7: Pruebas de Think Diffusion

Y así como se suponía, ThinkDiffusionXL, gracias a su conjunto de datos mayor, obtiene unos resultados muy buenos y coherentes en distintas peticiones como se puede observar en la figura 3.7. Esto se añade a que la diferencia en el tiempo de generación no es muy notable en comparación con los demás.

La parte mala de este modelo es que, a parte de ser pesado, no se ha encontrado en ninguna nube, ni siquiera a través de un muro de pago. Y de ejecutarlo en un servidor

propio sería un consumo de recursos considerable teniendo en cuenta el tamaño del modelo.

Tabla de criterios de evaluación

Criterio	Puntuación
Facilidad de uso	2
Versatilidad	9
Rapidez	8
Consistencia	7
Coherencia	7

Tabla 3.3: Evaluación final de Think Diffusion XL

3.2.3. DALL-E 3

Descripción general

DALL-E (14) es un modelo de inteligencia artificial generativa desarrollado por OpenAI. Como los anteriormente mencionados, DALL-E es capaz de generar imágenes a partir de texto de manera versátil y consistente.

DALL-E, al ser creado por la propia OpenAI, dispone de muchísimos recursos con los que trabajar y una base robusta gracias a los múltiples modelos de los que dispone la empresa. Esto lo convierte en un candidato muy bueno para nuestro proyecto.

A diferencia de sus competidores, este modelo utiliza la tecnología de transformers para cumplir su cometido, por lo que sus resultados serán mucho más diferentes a los anteriores vistos.

Resultados en pruebas



(a) Prueba fácil



(b) Prueba media



(c) Prueba difícil

Figura 3.8: Pruebas de DALL-E 3

Para empezar, hay que destacar que las imágenes de DALL-E 3, al momento de las pruebas, se hicieron con la herramienta de creación de imágenes de Bing. Esta herramienta por defecto crea imágenes de 1024x1024, mientras que las otras pruebas se han realizado con 512x512.

Independientemente de ese factor, se puede observar en la figura 3.8 una gran versatilidad y creatividad en las imágenes. Todas las pruebas se realizaron con las mismas peticiones, pero solo DALL-E se tomó libertades creativas en algunas de ellas. Ahora bien, hay que valorar si esa creatividad es buena o mala para lo que se está intentando conseguir.

Tabla de criterios de evaluación

Criterio	Puntuación
Facilidad de uso	8
Versatilidad	9
Rapidez	8
Consistencia	9
Coherencia	8

Tabla 3.4: Evaluación final de Think Diffusion XL

3.2.4. Conclusiones

Una vez analizados los modelos de generación de imágenes, se ha decidido utilizar DALLE-3 en el proyecto final. Esto se debe a que, a pesar de sus pequeñas desventajas, es un modelo muy versátil y consistente, a la vez que ligero y rápido. Estas diferencias se pueden apreciar más en la tabla 3.5, en la que se ha recogido la información de las tablas 3.2, 3.3 y 3.4 para hacer una comparación final más clara.

A pesar de que la decisión final haya sido DALL-E 3, en las fases de desarrollo tempranas e intermedias se ha utilizado Stable Diffusion para las pruebas generales. Stable Diffusion no es de los mejores modelos, pero su gran ventaja de accesibilidad a través de la nube sin muros de pago permite un desarrollo cómodo y sin malgastar nada de dinero en numerosas pruebas.

ThinkDiffusionXL fue descartado en cualquier instancia ya que, incluso teniendo la posibilidad de tener un servidor propio donde alojarlo y ejecutarlo, no valía la pena el esfuerzo teniendo en cuenta la similitud de resultados con DALL-E 3.

Criterio	StableDiffusion	ThinkDiffusionXL	DALL-E 3
Facilidad de uso	9	2	8
Versatilidad	8	9	9
Rapidez	9	8	8
Consistencia	2	7	9
Coherencia	3	7	8

Tabla 3.5: Comparación de los modelos

3.3. Elección del clasificador

A la hora de elegir la manera de clasificar estas imágenes, se ha seguido un procedimiento complementario al anterior. Estas pruebas consisten en intentan etiquetar las

imágenes generadas anteriormente, observando así cual es el clasificador más adecuado para el proyecto. Para ello se ha escogido una imagen generada por cada uno de los modelos y se le han presentado las tres imágenes al clasificador.

A pesar de que solo se va a usar un modelo generativo, se ha decidido realizar las pruebas con todos ellos para escoger no solo el clasificador que etiquete mejor a un modelo, sino a cualquier modelo de características parecidas a las que se buscan en el proyecto. De esta manera, si ocurre algún problema con el modelo elegido, se puede sustituir fácilmente por otro sin ninguna preocupación de si el clasificador rendirá bien en ese escenario. En las pruebas solo se citarán las 3 etiquetas más significativas para no sobrecargar el documento de información.

A parte de los resultados de las pruebas, también se tendrán en cuenta las funcionalidades extra que presenten los clasificadores, como podrían ser funciones para el color o clasificación como contenido adulto. Esto puede llegar a ser muy útil si se quisiese continuar el proyecto, ya que se dispondría de muchas más herramientas con las que ampliar las mecánicas del videojuego. Toda la evaluación queda recogida en la tabla 3.6, en la que se enumeran los criterios y se explica que se evalúa en cada uno de ellos.

Criterio	Puntuación
Facilidad de uso	API “User friendly”, servicios en la nube y precio
Grado de acierto	Capacidad del clasificador para etiquetar las imágenes
Rapidez	Velocidad del modelo a la hora de generar imágenes
Consistencia	Capacidad del modelo para mantener los resultados con la misma imagen sin fallar
Funcionalidades extra	Funcionalidades extra que presente el clasificador

Tabla 3.6: Evaluación final ejemplo

3.3.1. Amazon Rekognition

Descripción general

Amazon Rekognition (15) es un servicio de análisis de imágenes y vídeos basado en Deep Learning. Este servicio permite a los desarrolladores integrar fácilmente análisis de imágenes y vídeos, facilitando tareas como el reconocimiento facial, la identificación de objetos y escenas, la detección de texto en imágenes y la moderación de contenido.

Resultados en pruebas



(a) Prueba fácil



(b) Prueba media



(c) Prueba difícil

- | | | |
|----------------|-------------------|-----------------|
| ■ Banana 100 % | ■ Canopy 99.9 % | ■ Food 99.2 % |
| ■ Food 100 % | ■ Umbrella 99.9 % | ■ Plant 99.2 % |
| ■ Fruit 100 % | ■ Vacío | ■ Tomato 99.2 % |

Figura 3.9: Pruebas de Amazon Rekognition

En las pruebas se observa que este clasificador no tiene un mal desempeño con ninguna de las pruebas tal y como se puede observar en la figura 3.9. En el caso del paraguas solo ha dado dos resultados, pero uno de ellos es el correcto así que no hay ningún problema.

No solo clasifica correctamente sino que además tarda muy poco tiempo en conseguir etiquetar la imagen que se le presenta. Lo único que se podría tomar como desventaja es que cuenta con un muro de pago, pero también tiene una prueba gratuita generosa.

Tabla de criterios de evaluación

Criterio	Puntuación
Facilidad de uso	6
Grado de acierto	9
Rapidez	8
Consistencia	9
Funcionalidades extra	8

Tabla 3.7: Evaluación final de Think Diffusion XL

3.3.2. Google Vision

Descripción general

Google Cloud Vision (16), o también conocido como Google Vision, es un servicio de análisis de imágenes basado en inteligencia artificial de la plataforma de Google Cloud Platform. Google Vision ofrece una amplia gama de funcionalidades, al igual que su competidor Amazon Rekognition, que permiten muchas cosas más a parte del etiquetado de imágenes.

Resultados en pruebas



(a) Prueba fácil



(b) Prueba media



(c) Prueba difícil

- | | | |
|---------------|-----------------|---------------|
| ■ Food 98 % | ■ Product 91 % | ■ Food 97 % |
| ■ Banana 97 % | ■ Azure 90 % | ■ Plant 95 % |
| ■ Plant 94 % | ■ Umbrella 87 % | ■ Tomato 90 % |

Figura 3.10: Pruebas de Google Vision

Nuevamente se puede observar que este clasificador también funciona bastante bien, pero que presenta algún problema con el paraguas. En este caso, como se aprecia en la figura 3.10, este clasificador ha sido menos específico que su competidor, pero de igual manera ha conseguido etiquetar correctamente todos los objetos sin ningún problema.

Google Vision tampoco presenta ningún problema con el tiempo entre imágenes y, al igual que Amazon Rekognition, tiene muro de pago y a su vez da una prueba gratuita equivalente a 300 euros para utilizar en toda la nube.

Tabla de criterios de evaluación

Criterio	Puntuación
Facilidad de uso	8
Grado de acierto	8
Rapidez	8
Consistencia	9
Funcionalidades extra	8

Tabla 3.8: Evaluación final de Think Diffusion XL

3.3.3. Conclusiones

A la hora de escoger clasificador no ha habido mucha diferencia entre los candidatos puestos a prueba. En este caso se ha decidido optar por Google Vision por comodidad, ya que la facilidad de uso es en lo único que se diferencian, como se observa en la tabla 3.9. Esta tabla se ha usado para recoger la información de las tablas 3.7 y 3.8 para hacer la comparación entre ambas más visible. Se podrían usar cualquiera de los dos ya que

ambos desempeñan bastante bien su tarea, pero había que escoger uno, así que se ha optado por el que más sencillo se hacía de utilizar.

Criterio	Amazon Rekognition	Google Cloud Vision
Facilidad de uso	6	8
Grado de acierto	9	8
Rapidez	8	8
Consistencia	9	9
Funcionalidades extra	8	8

Tabla 3.9: Comparación de los clasificadores

3.3.4. Alternativa al clasificador

Una alternativa de los clasificadores que merece mención son los modelos de lenguaje convencionales a los que se les puede enviar ficheros. Estos modelos de lenguaje como GPT 4 o Gemini 1.5 son capaces de interpretar una imagen y sacar información de ella, como los objetos o las características de la misma. Obviamente esto los convierte en herramientas que podrían ser utilizadas con el fin de etiquetar las imágenes generadas.

Podría parecer un poco excesivo utilizar uno de estos modelos gigantes para una tarea tan concreta, pero sus tiempos de respuesta no difieren mucho de los clasificadores antes mencionados. Sin embargo, en cuanto a la clasificación eran más precisos que los clasificadores convencionales, lo que los volvían una opción perfectamente válida.

Capítulo 4

Desarrollo del proyecto

Este capítulo tratará principalmente todo lo relacionado con el desarrollo del videojuego. El capítulo empieza por las partes más generales, como pueden ser el flujo del juego o los aspectos básicos de un juego de dos dimensiones. Posteriormente se tratan elementos más específicos, como las conexiones con las APIs necesarias, la creación de objetos o el diseño de los puzzles. Finalmente el capítulo termina hablando sobre los problemas encontrados en el desarrollo.

4.1. Flujo común del juego

El flujo para el que está planteado el juego es el que se presenta en la figura 4.1. Como se puede observar, no hay un amplio árbol de posibilidades en este sentido, ya que el videojuego se expande sobre todo con la posibilidad de creación dinámica de cualquier objeto. La complejidad del juego variará dependiendo de la dificultad de los puzzles, así como de la capacidad de comprensión del jugador.

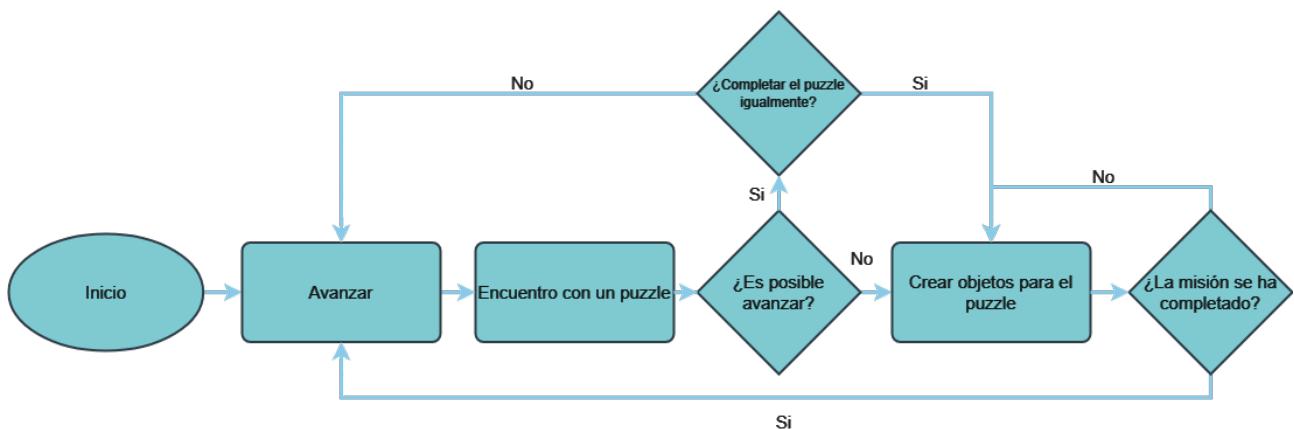


Figura 4.1: Diagrama de flujo básico del juego

4.2. Aspectos convencionales de juegos 2D

4.2.1. Controlador del jugador

Para los controles del jugador se ha optado por seguir un pequeño tutorial de un controlador hecho a partir de físicas personalizadas (17). Tanto la gravedad y las colisiones como el movimiento está todo representado en una clase llamada “Controller2D”, que

nos permite tener un control muy cómodo del jugador, así como una personalización muy amplia.

Las variables de movimiento han sido separadas en un “Scriptable Object” de Unity, para tener la ventaja de poder crear distintos conjuntos de variables y dar más flexibilidad en el control del jugador.

Todo esto se ha implementado en una máquina de estados jerárquica hecha con la librería anteriormente mencionada en 3.1.2. Esta máquina de estados tiene dos estados “base” que son: “Grounded” y “Airborne”. Estos dos estados, al ser parte de una máquina de estados jerárquica, actúan como pequeñas máquinas de estados individuales, y tienen dentro de sí mismas dos estados más: uno cuando el jugador está en movimiento y otro cuando no lo está.

Por último, se ha utilizado el componente de Unity llamado “Cinemachine” para hacer que los movimientos de la cámara del jugador sean mucho menos abruptos y sean más estables y suaves.

4.2.2. Diseño de niveles

En cuanto al diseño de niveles del videojuego, tal y como se mencionó anteriormente, se ha hecho uso de el paquete visto en 3.1.2 en conjunto a la utilidad vista en 3.1.2. Una vez configurado los elementos del paquete “Mossy Cavern” con las reglas automáticas de la herramienta “Tilemap Auto Rule Tile”, se procedió a implementar formas físicas para cada “tile” que se utilizase en la paleta generada.

Una vez se tenía plataformas por las que caminar y saltar, se utilizaron también algunas de las decoraciones que incluía el paquete “Mossy Cavern” para hacer los niveles mucho más vistosos.

4.2.3. Post-procesado y extras

Con el objetivo de hacer el videojuego todavía más llamativo, se implementaron algunas técnicas de post-procesado con la herramienta “Global Volume” de Unity. Los efectos que se han implementado son tres: Vignette, Color Adjustments y Bloom. El impacto de estos efectos se puede ver en la figura 4.2.

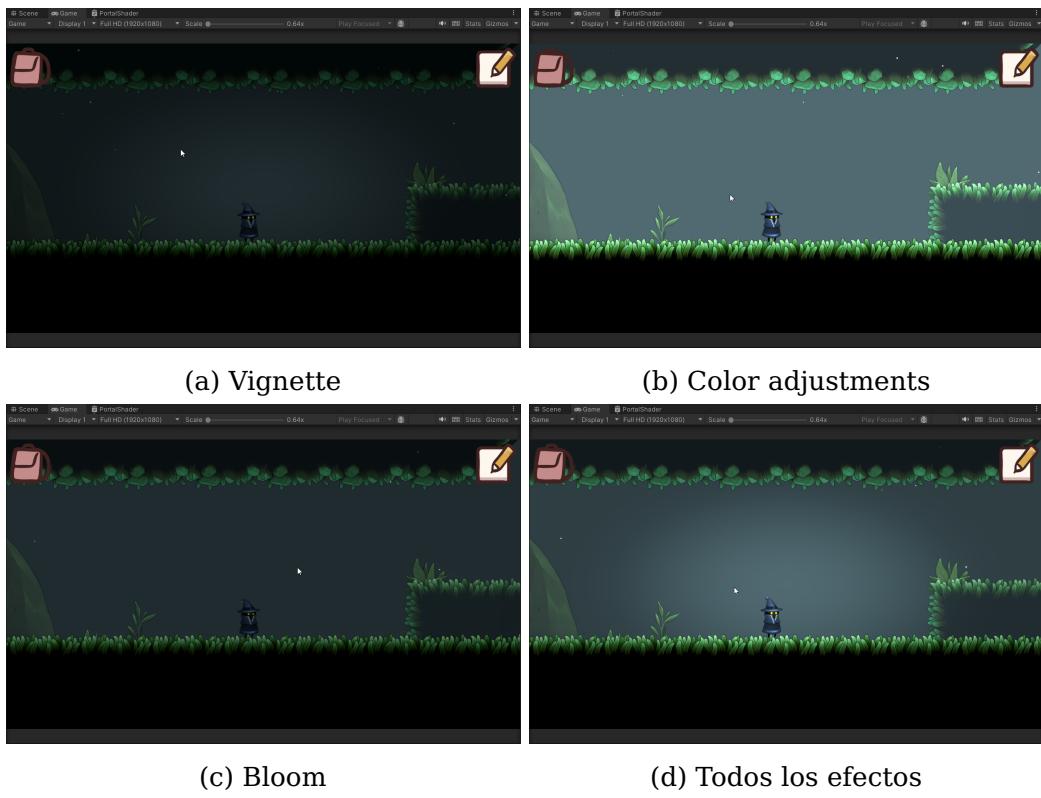


Figura 4.2: Comparaciones post-procesado

4.3. Integración de inteligencia artificial

Para integrar las inteligencias artificiales con el videojuego se ha seguido la estructura de clases que se observa en la figura 4.3. Se ha optado por utilizar la composición de clases en conjunto al patrón de diseño “Singleton” para así tener una instancia dentro del juego que se ocupe únicamente de la generación y etiquetación de los objetos que el jugador requiere.

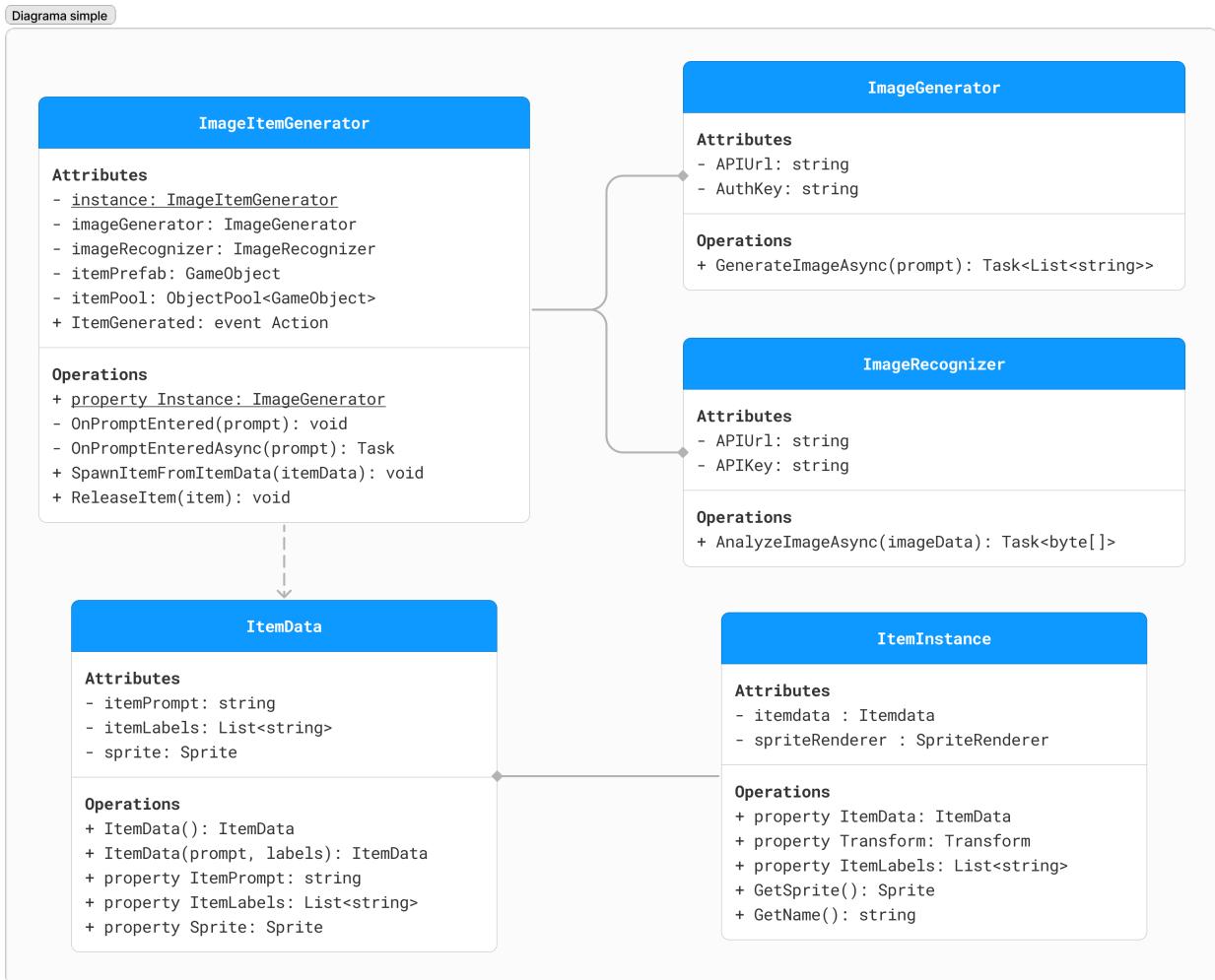


Figura 4.3: Diagrama de clases para la generación de objetos

Conección con IA generativa

Para conectar el modelo de inteligencia artificial generativa con el proyecto se han utilizado las funcionalidades de networking que ofrece Unity. Como se puede observar en la figura 4.4, basta con una petición al servicio en la nube para obtener la imagen.

En el caso de este proyecto se han tenido que utilizar unas clases adicionales para el uso de la API de OpenAI, como se observa en la figura 4.5, así como un método específico para descargar la imagen una vez generada, ya que la API nos devuelve un enlace con la imagen en vez de la imagen en sí.

```

1  public class ImageGenerator
2  {
3      private const string APIUrl = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
4      private const string AuthKey = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
5      public async Task<byte[]> GenerateImageAsync(string inputPrompt)
6      {
7          var requestBody = new OpenAIRequest
8          {
9              model = "dall-e-3",
10             prompt = inputPrompt,
11             n = 1,
12             size = "1024x1024"
13         };
14         var jsonData = JsonUtility.ToJson(requestBody);
15         Debug.Log("Generating image with data: " + jsonData);
16         return await SendRequestAsync(jsonData);
17     }
18     private async Task<byte[]> SendRequestAsync(string jsonData)
19     {
20         var request = new UnityWebRequest(apiUrl, "POST");
21         var bodyRaw = System.Text.Encoding.UTF8.GetBytes(jsonData);
22         request.uploadHandler = new UploadHandlerRaw(bodyRaw);
23         request.downloadHandler = new DownloadHandlerBuffer();
24         request.SetRequestHeader("Content-Type", "application/json");
25         request.SetRequestHeader("Authorization", $"Bearer {apiKey}");
26         request.SetRequestHeader("User-Agent", "Unity");
27
28         var asyncOperation = request.SendWebRequest();
29         while (!asyncOperation.isDone) await Task.Yield();
30         if (request.result is UnityWebRequest.Result.ConnectionError or
31             UnityWebRequest.Result.ProtocolError)
32         {
33             // Log error ...
34         }
35         var response =
36             JsonUtility.FromJson<OpenAIResponse>(request.downloadHandler.text);
37         var imageUrl = response.data[0].url;
38         return await DownloadImage(imageUrl);
39     }
40     private static async Task<byte[]> DownloadImage(string imageUrl)
41     {
42         UnityWebRequest request = UnityWebRequestTexture.GetTexture(imageUrl);
43         var asyncOperation = request.SendWebRequest();
44         while (!asyncOperation.isDone) await Task.Yield();
45         if (request.result is UnityWebRequest.Result.ConnectionError or
46             UnityWebRequest.Result.ProtocolError)
47         {
48             // Log error ...
49         }
50         var texture = DownloadHandlerTexture.GetContent(request);
51         return texture.EncodeToPNG();
52     }
}

```

Figura 4.4: Código para la conexión con el modelo de IA generativa

```

1 [System.Serializable]
2 public class OpenAIRequest
3 {
4     public string model;
5     public string prompt;
6     public int n;
7     public string size;
8 }
9
10 [System.Serializable]
11 public class OpenAIResponse
12 {
13     public string created;
14     public OpenAIData[] data;
15 }
16
17 [System.Serializable]
18 public class OpenAIData
19 {
20     public string revised_prompt;
21     public string url;
22 }

```

Figura 4.5: Clases extra para la petición de DALLE

Conexión con clasificador

A la hora de conectar el clasificador con el proyecto se utilizaron las mismas funcionalidades que con el modelo de IA generativo.

En cuanto al funcionamiento, es exactamente igual al código de la generación de imágenes, exceptuando la parte de la descarga. Se realiza la petición al servicio en la nube y una vez recibida se procesa la información para devolverla.

En este caso, en la figura 4.7, se ha adjuntado un código válido para la obtención de etiquetas mediante la API de Google Cloud Vision. Para completar la petición de una manera más cómoda y sencilla se crearon las clases que se observan en la figura 4.6.

```

1 [System.Serializable]
2 public class BatchAnnotateImagesResponse
3 {
4     public List<AnnotateImageResponse> responses;
5 }
6 [System.Serializable]
7 public class AnnotateImageResponse
8 {
9     public List<EntityAnnotation> labelAnnotations;
10 }
11 [System.Serializable]
12 public class EntityAnnotation
13 {
14     public string description;
15     public float score;
16 }

```

Figura 4.6: Clases extra para la petición de Google Cloud Vision

```

1  public class ImageRecognizer
2  {
3
4      private const string APIKey = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
5      private const string APIUrl = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
6
7      public Task<List<string> AnalyzeImageAsync(byte[] imageData)
8      {
9          var jsonRequest = @"
10         {
11             ""requests"": [
12                 {
13                     ""image"": {
14                         ""content"": """ + System.Convert.ToBase64String(imageData) +
15                         @"""
16                     },
17                     ""features"": [
18                         {
19                             ""type"": ""LABEL_DETECTION"",
20                             ""maxResults"": 10
21                         }
22                     ]
23                 }
24             ]
25         }";
26         using UnityWebRequest www = new UnityWebRequest(APIUrl, "POST");
27         byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(jsonRequest);
28         www.uploadHandler = new UploadHandlerRaw(bodyRaw);
29         www.downloadHandler = new DownloadHandlerBuffer();
30         www.SetRequestHeader("Content-Type", "application/json");
31
32         // Send request
33         var asyncOperation = www.SendWebRequest();
34         while (!asyncOperation.isDone)
35         {
36             await Task.Yield();
37         }
38
39         if (www.result != UnityWebRequest.Result.Success)
40         {
41             Debug.LogError("Request failed: " + www.error);
42             Debug.LogError("Response: " + www.downloadHandler.text);
43             return null;
44         }
45         // Parse response
46         string jsonResponse = www.downloadHandler.text;
47         var response = JsonUtility.FromJson<Response>(jsonResponse);
48         var labels = response.candidates[0].content.parts[0].text.Split(',');
49         List<string> LabelsDetected = new List<string>();
50         foreach (var label in labels)
51         {
52             LabelsDetected.Add(label.Trim());
53         }
54         return LabelsDetected;
55     }
}

```

Figura 4.7: Código para la conexión con el clasificador

4.3.1. Creación de objetos

La creación de objetos dentro del proyecto se ha realizado mediante la instanciación de unas piezas conocidas como “prefabs” en Unity. Estos elementos nos permiten tener una plantilla de objeto que se instanciará con la información que nos proporcionen los modelos.

Los objetos del juego cuentan con dos elementos principales:

- La cadena de texto con la que se hizo la petición y se genera la imagen
- La lista de etiquetas que devuelve el clasificador con la imagen generada a partir de la cadena.

Esta información es más que suficiente para cumplir el cometido de los puzzles de este proyecto. De la misma forma se podría aumentar su complejidad utilizando las funcionalidades extra del clasificador si hiciese falta.

La creación de objetos a nivel de código se gestiona con la clase de la figura 4.10, que se ocupa de implementar las clases vistas en las figuras 4.4 y 4.7. A su vez también se encarga de hacer de intermediario entre la interfaz de usuario y la generación de objetos, así como de gestionar los objetos en la escena mediante una “Object Pool”.

El código que se ocupa de la generación de los objetos se puede observar en la figura 4.9, en la que hay dos funciones principales. La primera función es la encargada de crear el objeto al completo, mientras que la segunda es una imagen auxiliar para la creación de los objetos una vez se tienen los datos.

En la figura 4.8 se puede observar el flujo de creación de un objeto dentro del juego, pero más concretamente, se realiza lo siguiente:

- Se recibe el prompt del jugador.
- Envía el prompt al generador y espera a recibir la imagen generada por el modelo de inteligencia artificial generativa.
- Una vez se obtiene la imagen, esta se envía al clasificador y se espera a obtener las etiquetas de la imagen.
- Cuando se tienen las etiquetas, se crea una textura 2D para adjuntar la imagen, así como una clase tipo “ItemData” para almacenar la información mencionada anteriormente.
- Se instancia el objeto dentro del juego dándole a la función auxiliar los datos del objeto.

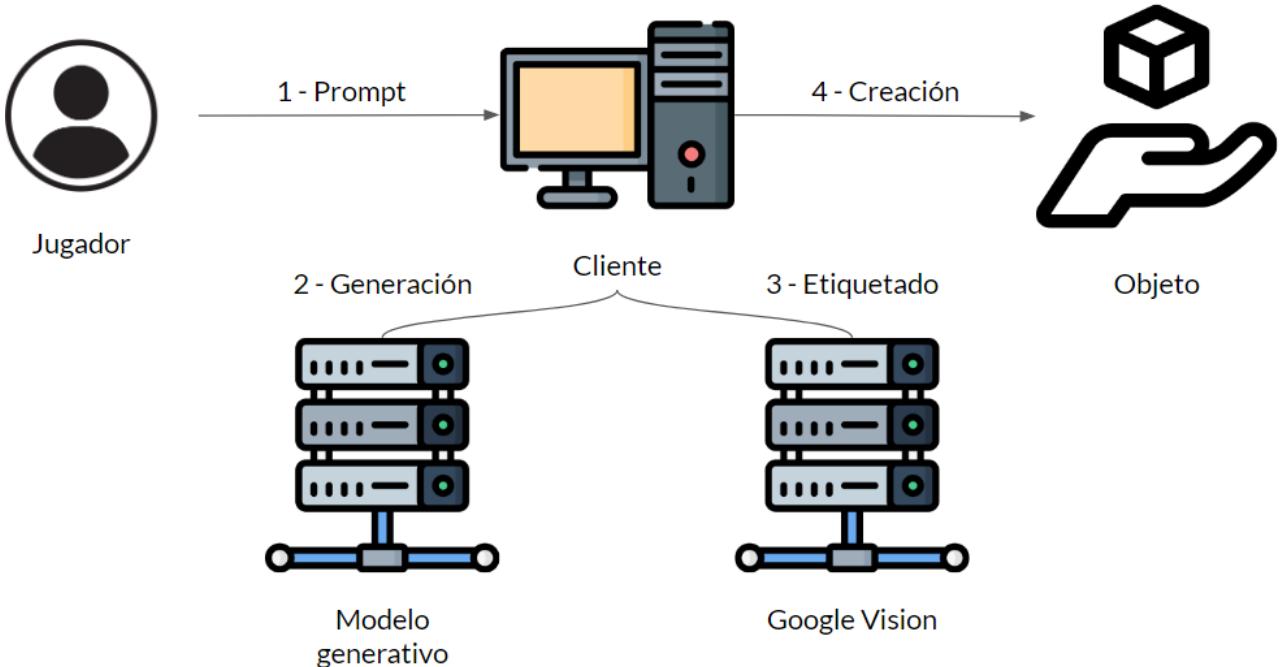


Figura 4.8: Diagrama de generación de un objeto dentro del proyecto

```

1 ...
2 private async Task OnPromptEnteredAsync(string prompt)
3 {
4     var generatedImage = await _imageGenerator.GenerateImageAsync(prompt);
5
6     if (generatedImage == null)
7     {
8         ItemGenerationFailed?.Invoke();
9         return;
10    }
11
12    var labelsDetected = await _imageRecognizer.AnalyzeImageAsync(generatedImage);
13    var texture = new Texture2D(1, 1);
14    var item = new ItemData(prompt, labelsDetected.ToList());
15
16    texture.LoadImage(generatedImage);
17    item.Sprite = Sprite.Create(texture, new Rect(0, 0, texture.width, texture.height),
18        new Vector2(0.5f, 0.5f));
19
20    SpawnItemFromItemData(item);
21    ItemGenerated?.Invoke();
22}
23
24 public void SpawnItemFromItemData(ItemData itemData)
25 {
26     var itemObject = _itemPool.Get();
27     itemObject.GetComponent<ItemInstance>().ItemData = itemData;
28     var screenCenter = Camera.main.ScreenToWorldPoint(new Vector3(Screen.width / 2f,
29         Screen.height / 2f));
30     itemObject.transform.position = new Vector3(screenCenter.x, screenCenter.y, 1);
31 }
```

Figura 4.9: Funciones para la generación de objetos dentro del juego

```

1 public class ImageItemGenerator : MonoBehaviour
2 {
3     private ImageGenerator _imageGenerator;
4     private ImageRecognizer _imageRecognizer;
5
6     [SerializeField] private GameObject itemPrefab;
7     [SerializeField] private InputWindow inputWindow;
8
9     private static ImageItemGenerator _instance;
10    private ObjectPool<GameObject> _itemPool;
11    public event Action ItemGenerated;
12    public event Action ItemGenerationFailed;
13    ...

```

Figura 4.10: Variables principales de la clase ImageItemGenerator

4.3.2. Gestión de los tiempos de carga

Como es lógico, todo este proceso de creación conlleva una espera considerable por parte del jugador. A la hora de generar un objeto se hace de manera asíncrona para no parar el hilo principal durante toda la generación. Se pueden crear objetos en cualquier momento, incluso si ya hay objetos creándose en ese instante y una vez se termina la tarea correspondiente, el objeto aparece en el mundo.

En este proyecto, para dar una forma de que el jugador sepa que su objeto todavía se encuentra en medio de la generación, se muestra un reloj en movimiento en la interfaz de usuario. Habrá tantos relojes como objetos generándose simultáneamente, con un máximo de 8 relojes a la vez.

4.4. Diseño de los puzzles

En cuanto al diseño de los puzzles, se han implementado varios diseños de puzzles con las mecánicas presentadas. Se han creado puzzles de dos tipos concretos. Estos tipos principales son:

- Arrastrar y soltar encima de un elemento. Se consume el objeto.
- Proximidad a un elemento concreto en la escena. No consume el objeto

El segundo tipo de puzzle se creó con la intención de hacer que el jugador tenga una razón clara de usar el inventario, ya que si su objeto no es consumido al completar el puzzle, puede guardarlo para crearlo más tarde en caso de encontrarse con un puzzle similar.

Algunos de los puzzles creados son obligatorios para avanzar, mientras que otros no es necesario completarlos para seguir adelante. Los puzzles obligatorios, en este proyecto, presentan un obstáculo ineludible que el jugador no puede atravesar hasta que complete el puzzle en concreto. Por ejemplo, un puzzle obligatorio para continuar requiere que el jugador acerque algún tipo de luz a una planta, haciendo que la planta crezca y permita saltar más alto, como se ve en la figura 4.11.

Una ayuda para el jugador es la utilización de los signos de exclamación mencionados anteriormente, que permiten que el usuario identifique que elementos en la escena contienen una misión sin completar.



(a) Menú descriptivo del objeto interactivo



(b) Introducción de prompt



(c) Misión completada por proximidad

Figura 4.11: Recorrido por un puzzle del videojuego

4.5. Experiencia del usuario

Para una experiencia de usuario mucho más amena, el juego al iniciar presenta un tutorial guiado al usuario en el que se explica todo lo que se puede hacer en el videojuego. El usuario solo tiene que seguir las instrucciones simples que se le presentan para conocer las mecánicas principales del videojuego y como completar puzzles utilizando lo que tiene a su disposición.

En una parte de ese tutorial se hace mención de un menú descriptivo que cada elemento interactivo en la escena. Gracias a este menú, el usuario es capaz de identificar, no solo los objetos que ha creado, sino también cualquier otro elemento en la escena. Esto es muy importante para tener una forma de hacer saber al usuario que se necesita para resolver cada puzzle, dando un texto descriptivo en caso de que el interactivo clicado tenga una misión activa.

4.6. Problemas encontrados con la integración

4.6.1. Estilo

Una pieza esencial de un videojuego es su estilo, así como la coherencia y cohesión entre todos sus elementos. Para este proyecto se pretendía tener un estilo de dibujos animados simple añadiendo al prompt el estilo, pero esto generaba una serie de problemas.

En primer lugar, el querer un estilo específico conllevaba la necesidad de que las imágenes generadas no tuviesen fondo, algo que los modelos generativos probados no controlaban. La alternativa más sencilla era añadir una capa extra con otra llamada más a otro servicio para quitar el fondo, pero esto aumentaba los tiempos de carga.

Por otro lado está el control del jugador sobre el prompt. Si se quisiera mantener un estilo específico, habría que limitar el prompt del jugador de cierta manera para que no pueda sobrepasar el estilo predeterminado. Esto rompería la libertad de creación del jugador y haría mucho más tedioso el procesado del prompt.

Hay algunos modelos que permiten el añadido de un “prompt negativo”, que hace justo lo contrario que el prompt principal. Todo lo que se añada en el prompt negativo intentará ser evitado por el modelo generativo.

4.6.2. Posición lógica

Si se quisiese seguir el mismo esquema de puzzles no habría ningún problema. Pero cuando se quiere expandir las posibilidades todavía más el problema más molesto para conseguirlo se encuentra en la generación.

En el caso de querer que los objetos fuesen utilizables de alguna manera, necesitaríamos conocer su posición lógica en el espacio para no romper la inmersión. Por ejemplo, si se genera una silla, se querrá que esta se vea desde una perspectiva lateral o frontal para poder sentarnos en ella. Si la silla se encontrase de espaldas no sería lógico poder utilizarla.

Este inconveniente se vuelve más grande aún cuando se piensa en como se usan esos objetos, sino como los agarraría un personaje, o como se llevarían puestos. Hay algunos casos que pueden ser resueltos personalizando el prompt del usuario de tal manera que la posición de los objetos fuese siempre una específica, pero esto no es nada fiable, y menos aún dando al jugador completa libertad sobre el prompt.

Capítulo 5

Evaluación y resultados del proyecto

5.1. Procedimiento de evaluación

Para evaluar el proyecto se ha decidido enviar el prototipo a algunas personas y posteriormente hacer que esas personas completen una encuesta. Esta encuesta se dividía en cuatro secciones principales:

En primer lugar, como se observa en la figura 5.1, se han hecho preguntas relevantes al desempeño en el videojuego. Con estas preguntas se consigue saber si el prototipo creado se puede completar y se puede evaluar su dificultad en base a las respuestas de los usuarios.

Posteriormente, en la figura 5.2 se evalúan cuestiones generales sobre el prototipo, como su valoración general, sus mecánicas o sus puzzles. Al dar respuesta a estas cuestiones generales se obtiene un grado de satisfacción general de los usuarios con el prototipo, dejando claro cuáles si hay alguna carencia en las partes clave.

En la tercera sección se realizan las preguntas de la figura 5.3, que trata un aspecto más específico del prototipo realizado. Se tratan preguntas sobre el nivel de comprensión del juego tras el tutorial, la interfaz de usuario o la valoración del sistema de IA para generar imágenes. Estas preguntas ofrecen una visión de los usuarios más cercana al objetivo que se quiere lograr, que es la integración de la IA generativa en el videojuego.

Por último, como se puede ver en la figura 5.4, se ha dejado un espacio para aportar comentarios. Dejando este espacio se puede recibir un “feedback” muy útil, ya que puede no estar representado en ninguna de las secciones.

Cuestiones sobre desempeño del juego	
Pregunta	Tipo de respuesta
Grado de avance en el juego	Selección
Grado de dificultad del juego	Selección
En caso de no completar nada, escribir la razón	Escribir

Tabla 5.1: Preguntas de la sección 1

Cuestiones generales sobre el juego	
Pregunta	Tipo de respuesta
Valoración general del juego	Puntuación 1-5
Valoración general de las mecánicas	Puntuación 1-5
Valoración de los puzzles del juego	Puntuación 1-5
Valorar la posibilidad de jugar juegos similares	Puntuación 1-5

Tabla 5.2: Preguntas de la sección 2

Cuestiones sobre aspectos particulares del juego	
Pregunta	Tipo de respuesta
Valorar la comprensión del juego tras pasar el tutorial	Puntuación 1-5
Valorar interfaz de usuario	Puntuación 1-5
Valora el sistema de inventario del personaje	Puntuación 1-5
Valorar control del personaje (facilidad de moverse)	Puntuación 1-5
Valoración del sistema de IA para producir las imágenes	Puntuación 1-5
Valoración del sistema para comprender las imágenes	Puntuación 1-5

Tabla 5.3: Preguntas de la sección 4

Observaciones	
Pregunta	Tipo de respuesta
Observaciones y comentarios	Escribir

Tabla 5.4: Preguntas de la sección 5

5.2. Resultados de la evaluación

En la evaluación se han recopilado las respuestas de 12 personas distintas y se irá desglosando cada pregunta con sus correspondientes resultados totales.

En primer lugar, los resultados de la pregunta numero uno que se encuentra en la figura 5.1 permiten observar que todos los jugadores que probaron el prototipo consiguieron completarlo al completo. Esto probablemente se deba a la corta extensión del prototipo y a su dificultad no muy elevada.



Figura 5.1: Resultados de la pregunta 1

Seguidamente, los resultados de la pregunta número dos observables en la figura 5.2 aclaran lo dicho anteriormente, la dificultad del prototipo no es muy elevada y ningún jugador se encontró con alguna complicación especialmente difícil.



Figura 5.2: Resultados de la pregunta 3

Posteriormente se encontraría las respuestas de la pregunta número tres, pero ningún jugador tuvo problemas para completar el prototipo así que no se tiene ninguna respuesta para este apartado.

A continuación se presentan los resultados de la pregunta número cuatro en la figura 5.3, que abre la sección de valoración general. en los que se puede apreciar que el prototipo ha causado una buena impresión general en las personas que lo han probado.

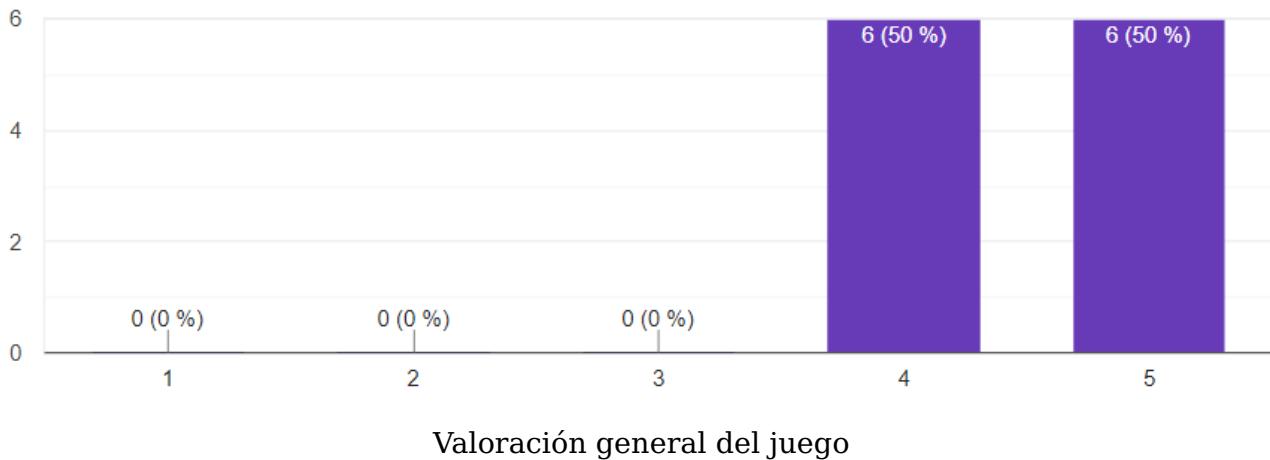


Figura 5.3: Resultados de la pregunta 4

Sucesivamente, se observan los resultados de la pregunta número cinco en la figura 5.4. Estos resultados, al igual que los anteriores, dan a entender que la idea que se ha propuesto del prototipo y sus mecánicas tiene una buena base sobre la que construir un videojuego más completo.

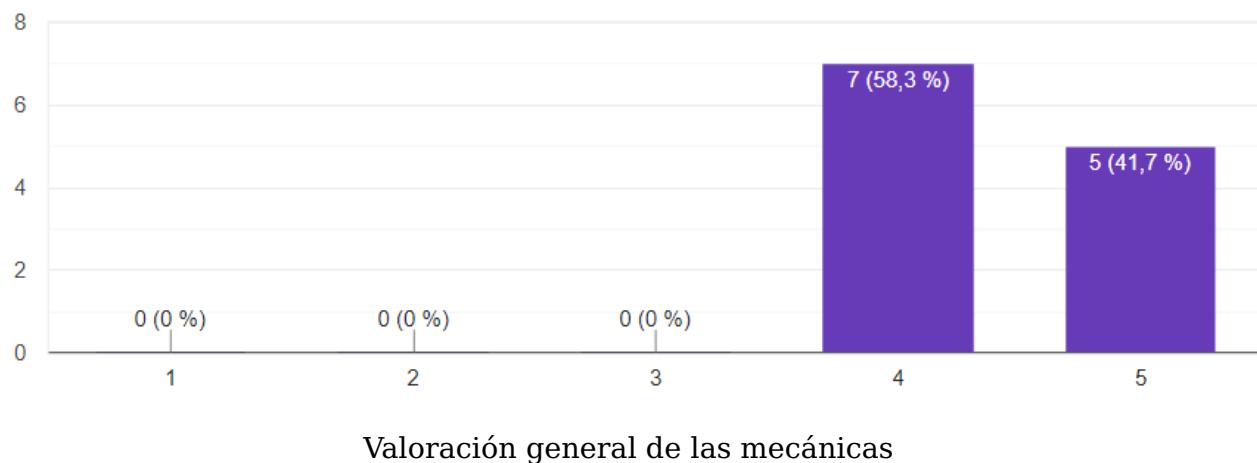


Figura 5.4: Resultados de la pregunta 5

Consecuentemente se hallan las respuestas de la pregunta número 6, visibles en la figura 5.5, en la que se empieza a ver una caída de las respuestas positivas, llegando a casi centrarse en una media de respuestas neutras. Esto se puede relacionar con la facilidad del propio prototipo y sus puzzles, ya que si la experiencia es demasiado sencilla genera aburrimiento en el jugador.

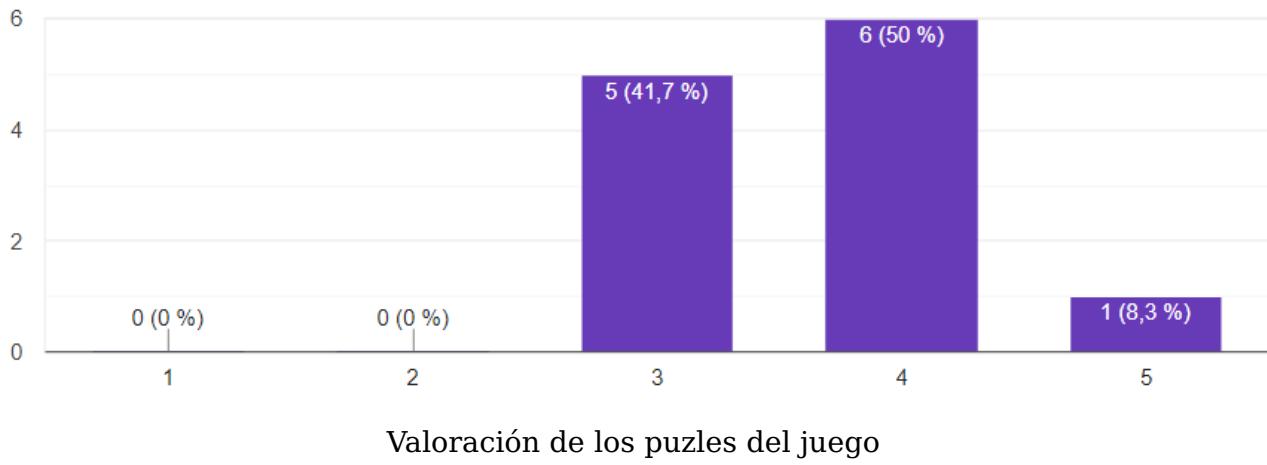


Figura 5.5: Resultados de la pregunta 6

Después, analizando las respuestas de la pregunta 7 observadas en la figura 5.6 se puede intuir que la mayoría de jugadores, al menos los que probaron el prototipo, jugarían a un videojuego de las mismas características en el caso de que este existiese.

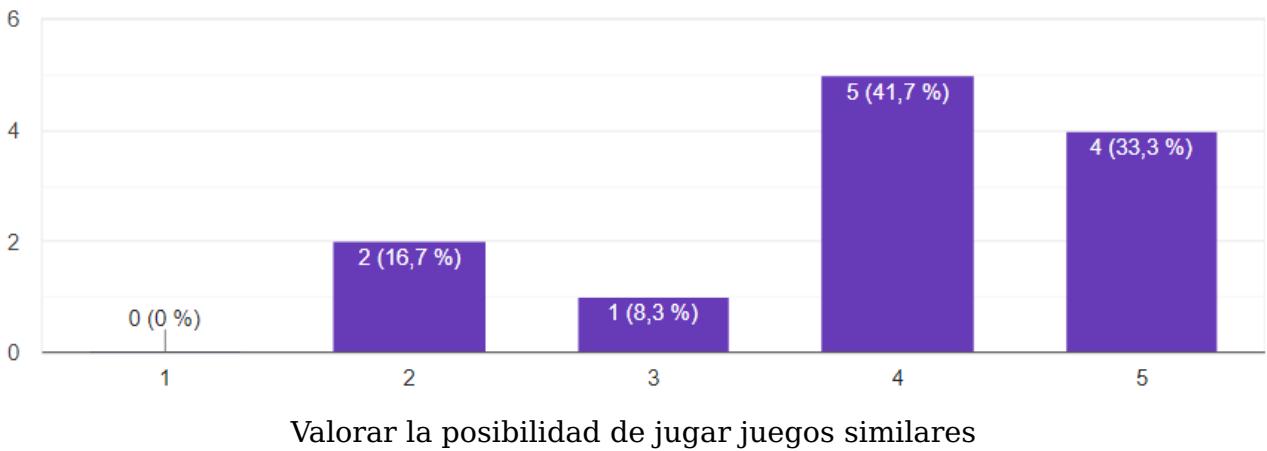


Figura 5.6: Resultados de la pregunta 7

Seguidamente en las respuestas de la pregunta número ocho, con sus respectivas respuestas en la figura 5.7, en la que se comienza con la valoración específica, se observa como el tutorial es de gran ayuda para la comprensión de las mecánicas del juego, ya que casi todos los jugadores lo han valorado positivamente.

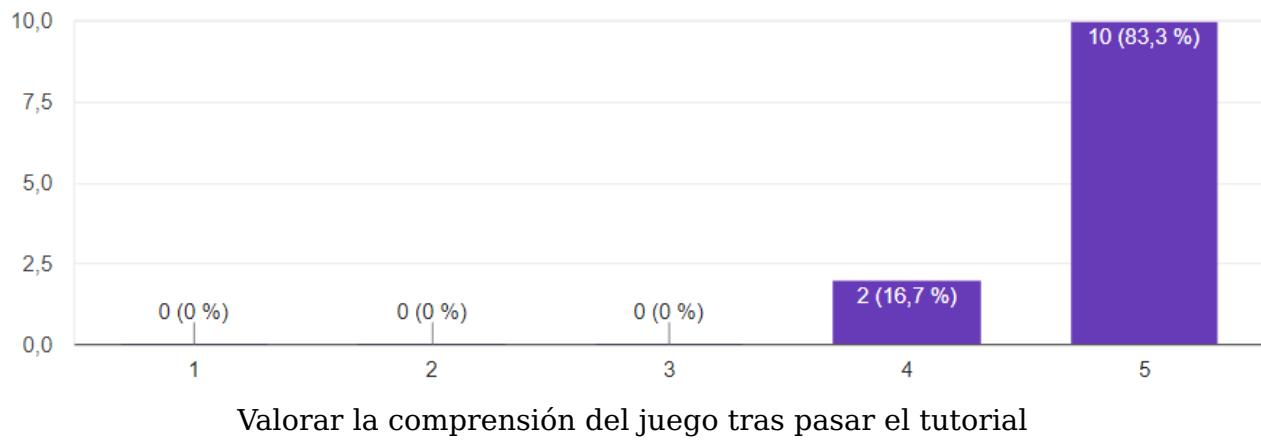


Figura 5.7: Resultados de la pregunta 8

Posteriormente, las respuestas de la pregunta número nueve, visibles en la figura 5.8, sugieren que la interfaz de usuario es decente y que no tiene ningún problema grave. Aún así parece que hay partes que podrían mejorarse para lograr una mayor comodidad al usuario.

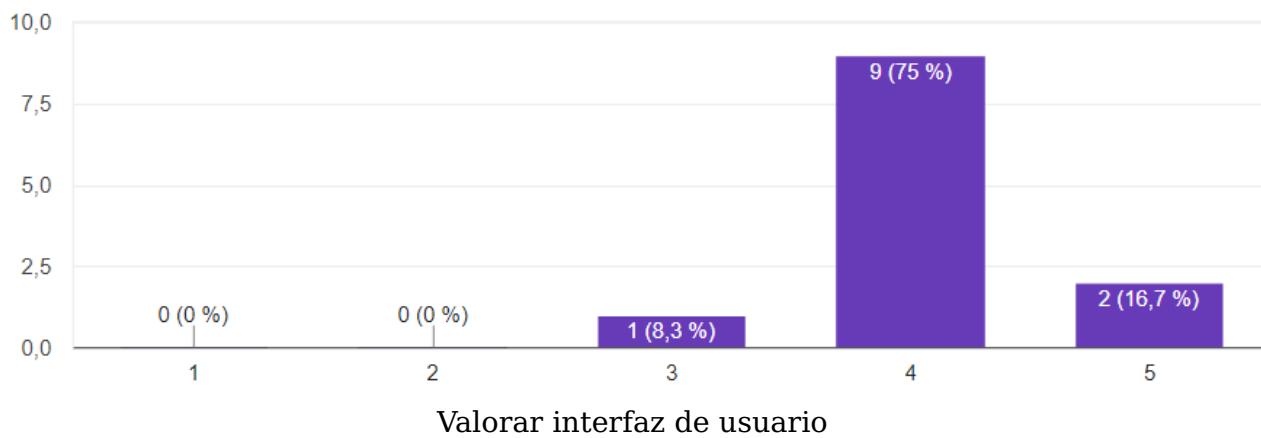


Figura 5.8: Resultados de la pregunta 9

Acto seguido, en las respuestas de la figura 5.9, que pertenecen a la pregunta número diez, dan a entender que el sistema de inventario cumple su función, aunque también existe margen de mejora en el mismo.

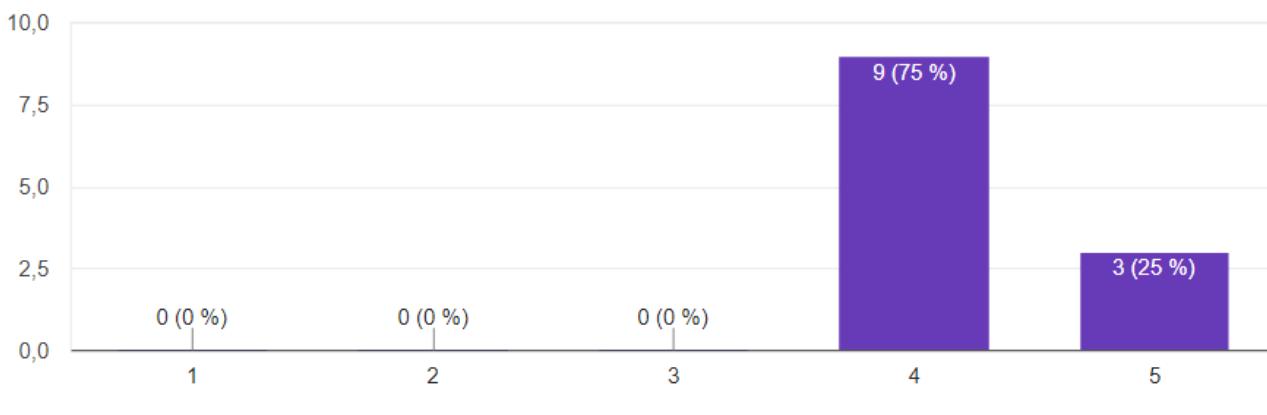


Figura 5.9: Resultados de la pregunta 10

A continuación se muestran en la figura 5.10 los resultados de la pregunta número once, que indican una alta satisfacción con el control del personaje. Teniendo en cuenta esta información, es considerable la reutilización del controlador en otros videojuegos 2D.

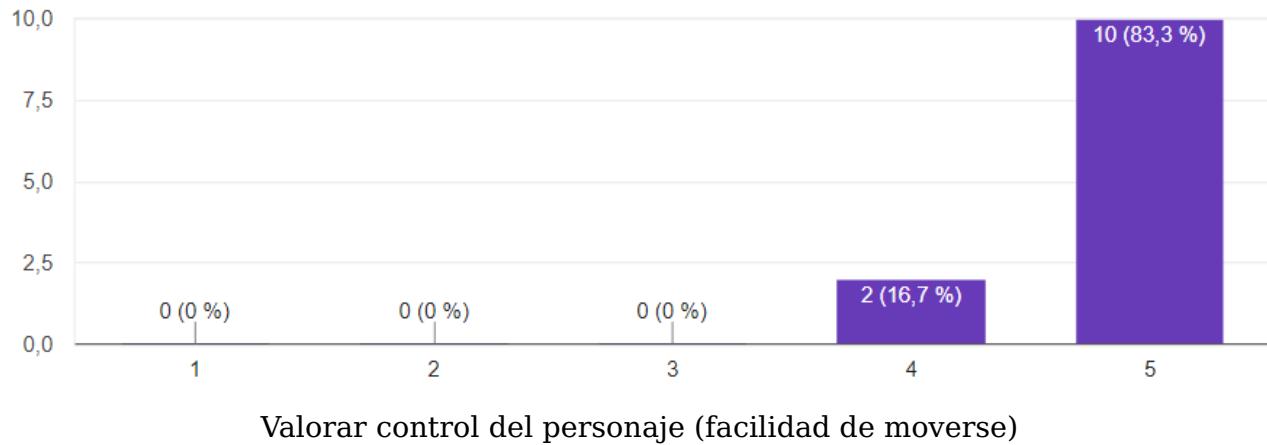


Figura 5.10: Resultados de la pregunta 11

Más tarde, en la figura 5.11 se analizan las respuestas de la pregunta número doce, en las que se tiene en cuenta el rendimiento del sistema de IA. Estas respuestas tienden a ser neutras probablemente por el tiempo de carga de la generación de objetos, ya que, a pesar de no ser excesivo, tampoco es relativamente corto y es algo que puede resultar desagradable a algunas personas.

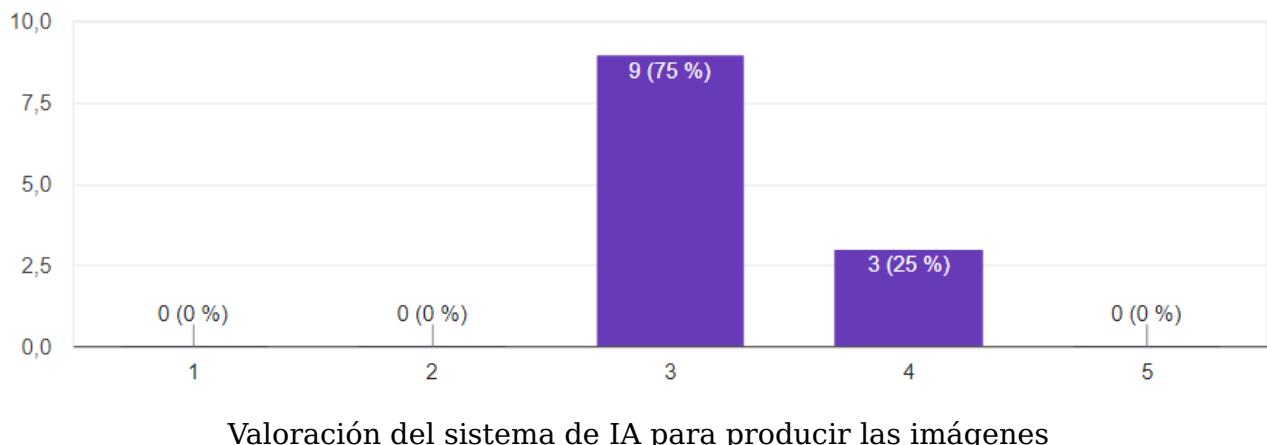


Figura 5.11: Resultados de la pregunta 12

Para dar fin a esta sección se observa la figura 5.12 que contiene los resultados de la pregunta número trece. Esta pregunta permite entrever que el rendimiento del clasificador utilizado ha sido decente y no ha ocasionado problemas.

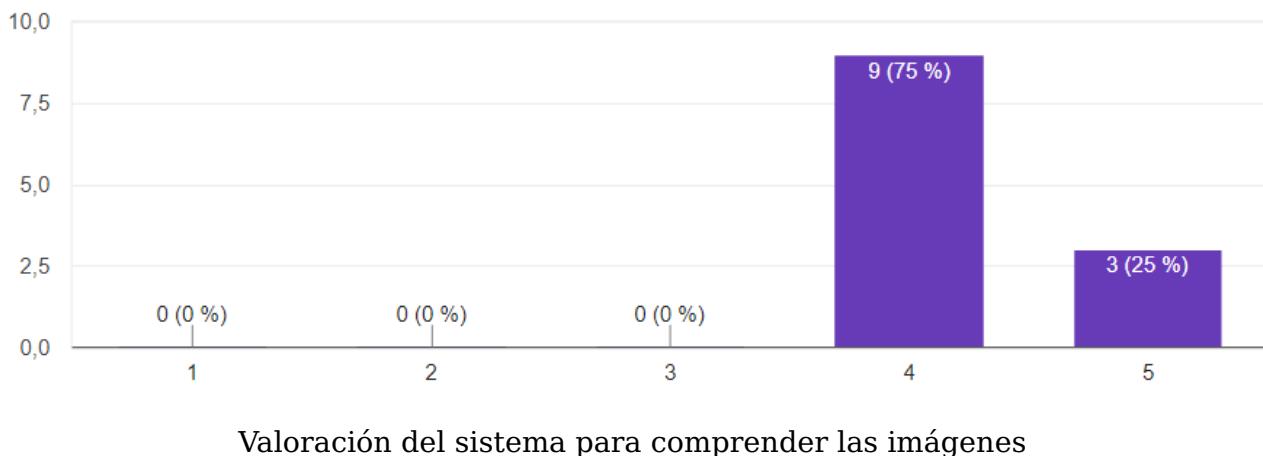


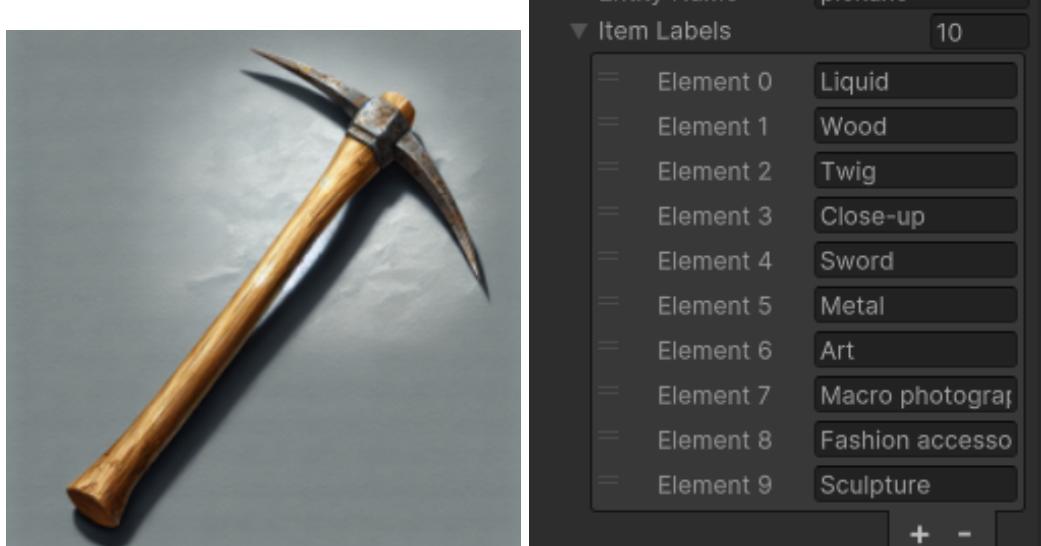
Figura 5.12: Resultados de la pregunta 13

Por último se tienen las respuestas de las observaciones, que en este caso han sido muy pocas y breves, valorando positivamente el proyecto, pero sin incluir ninguna queja ni comentario destacable más allá de lo tratado en otras preguntas.

5.3. Problemas encontrados

El mayor problema que se ha encontrado con el planteamiento del proyecto son los clasificadores. A pesar de que desempeñaron bien en las pruebas que se realizaron, una vez dentro del entorno del juego se vio claramente que no etiquetaban correctamente algunas imágenes.

Para ver este error mejor se puede observar la figura 5.13, en la que a la izquierda se tiene la imagen generada con el prompt “Pico” o “Pickaxe” y a la derecha las etiquetas que ha devuelto Google Cloud Vision. Claramente hay algún problema con este tipo de imágenes, puesto que herramientas como un martillo o un hacha se etiquetaban de manera muy parecida, pero el pico destacaba como el peor etiquetado. Este fallo hacía que el juego fuese frustrante ya se estaba intentando completar un puzzle muy simple y una solución tan clara no funcionaba por un fallo en las etiquetas del objeto.



(a) Imagen generada con prompt: "Pic-
co"
(b) Etiquetas devueltas por Google
Cloud Vision

Figura 5.13: Error en la clasificación

También se intentó cambiar de clasificador sobre la marcha, pero todos daban resultados similares a los que se ven en la figura 5.13.

Estos clasificadores estaban fallando en la única tarea para la que fueron inventados, y ese error estaba rompiendo completamente el proyecto, así que se optó por utilizar la alternativa que se mencionó en el capítulo 3.

El modelo de lenguaje que se utilizó para sustituir al clasificador fue Gemini 1.5 (18). Esto se decidió principalmente por dos razones:

- Para no utilizar GPT puesto que sería utilizar dos APIs de la misma empresa y podría dar un resultado poco realista, ya que la generación y clasificación podrían compartir conjuntos de datos.
- La generosa prueba gratuita de Gemini 1.5 hace que los gastos del prototipo no sean mayores que si se usara otra API de pago como podría ser la de GPT.

Para la petición se utilizó el siguiente mensaje:

1 "Return a list, and only the list, of labels that tells what's the object in the image.
For example, if it is a sun, you must response with the following list:sun,
sunlight, light, star, fire... and so on. Just write the labels separated by commas."

De esta manera no se tuvo que cambiar la implementación del reconocedor demasiado y se pudo arreglar fácilmente el error de etiquetado.

Capítulo 6

Conclusiones y líneas futuras

Los avances en la inteligencia artificial generativa han abierto muchas posibilidades en el mundo de los videojuegos, como la creación de paisajes o la generación de diálogos naturales y adaptativos. Esta capacidad para producir contenido de alta calidad en tiempo real no solo reducen los costes y tiempos de desarrollo, sino que también otorga un grado de libertad creativa a desarrolladores menos experimentados. Los desarrolladores ahora pueden enfocarse en aspectos más creativos, dejando las tareas más repetitivas y laboriosas a la inteligencia artificial.

En el marco de este proyecto, se ha desarrollado una videojuego que utiliza inteligencia artificial generativa como una de sus mecánicas principales, uso de la IA generativa poco habitual en el uso de los videojuegos. El prototipo ha demostrado ser adecuado como base de lo que podría ser un proyecto mayor. Sin embargo, este proyecto también ha descubierto ciertos problemas y carencias de las tecnologías tratadas como lo pueden ser la coherencia y el contexto del contenido generado. A pesar de esto, el proyecto ha cumplido los objetivos iniciales, poniendo a prueba el uso de la IA generativa en videojuegos y haciendo posible examinar las ventajas y desventajas de este tipo de implementación.

Además, aunque la tecnología ha avanzado considerablemente, aún existe una brecha en la capacidad de la IA para comprender y replicar la profundidad de la creatividad humana. A esto se suma la problemática moral del uso de contenido digital público para el entrenamiento de estos modelos sin el previo permiso de los artistas que han creado dicho contenido.

A pesar de todos estos inconvenientes, el resultado del proyecto es positivo teniendo en cuenta los resultados de la encuesta realizada. Esto no quiere decir que sea una implementación perfecta, pero sí que deja entrever que se está yendo por el buen camino.

De cara al futuro, lo más sensato es esperar a ver como evolucionan con el tiempo estos modelos. En primer lugar se espera que en un futuro se encuentre una mejora clara en la capacidad de la IA para generar contenido coherente y contextual en escenarios más complejos y dinámicos. Y a su vez se espera también que estos modelos sean mucho más rápidos o que la capacidad de cómputo sea mayor para tener una interacción más dinámica y cómoda para el jugador.

En cuanto al proyecto desarrollado, lo primero que se debería hacer si se quiere mejorar sería ubicar el alcance de la mecánica principal. Los puzzles con la tecnología y forma de resolverlos actual son limitados. Por lo que se podría reconsiderar el sistema actual o ampliarlo de manera que dé más libertad a la hora de hacer niveles. Una vez se haya tenido en cuenta esa posible expansión, se deberían tener en cuenta los aspectos convencionales de un videojuego, como pueden ser los sonidos, el arte o la música.

Capítulo 7

Conclusions and future work

Advances in generative artificial intelligence have generated many possibilities in the world of video games, such as the creation of landscapes or the generation of natural and adaptive dialogues. This ability to produce high-quality content in real time not only reduces development costs and times, but also gives a degree of creative freedom to less experienced developers. Developers can now focus on more creative aspects, leaving the more repetitive and laborious tasks to artificial intelligence.

As a part of this project, a video game has been developed that uses generative AI as one of its main mechanics, a use of generative AI that is unusual in video games. The prototype has proven to be suitable as a basis for what could be a larger project. However, this project has also uncovered problems and shortcomings of the technologies involved, such as the consistency and context of the generated content. Despite this, the project has met the initial objectives, testing the use of generative AI in video games and making it possible to examine the advantages and disadvantages of this type of implementation.

In addition, although technology has advanced considerably, there is still a gap in the ability of AI to understand and replicate the depth of human creativity. Added to this is the moral issue of using public digital content to train these models without the prior permission of the artists who created that content.

Despite all these drawbacks, the outcome of the project is positive considering the results of the survey conducted. This does not mean that it is a perfect implementation, but it does suggest that we are on the right track.

For the future, it makes sense to wait and see how these models evolve over time. First of all, it is expected that in the future we will find a clear improvement in the ability of AI to generate coherent and contextual content in more complex and dynamic scenarios. In turn, it is also expected that these models will be much faster, or that the computational capacity will be greater to have a more dynamic and comfortable interaction for the player.

As for the developed project, the first thing that should be done if improvements are to be made would be to locate the scope of the main mechanics. The puzzles with the current technology and way of solving them are limited. So the current system could be reconsidered or extended to give more freedom when making levels. Once this possible expansion has been taken into account, the conventional aspects of a videogame, such as sounds, art or music, should be also taken care of.

Capítulo 8

Presupuesto

8.1. Costes del desarrollo

Para el desarrollo del prototipo, como se ha visto en el principio del capítulo 3, se han utilizado herramientas de licencia gratuita y de uso libre, por lo que no hay un coste en esta parte del proyecto, más allá de los recursos necesarios como puede ser el ordenador donde se desarrolla o el internet necesario para las llamadas a las APIs.

En cuanto al coste respecto al tiempo, el sueldo medio al año de un ingeniero informático es de 39000€ según Glassdoor (19). Esto serían 3250€ al mes, lo que en una jornada laboral de 40 horas semanales se convierte a 20,312€/hora.

Un desglose de las fases del proyecto se encuentra en la figura 8.1. En esta tabla tienen en cuenta las horas que se han tomado para cada fase, y en base a esas horas se tiene su coste monetario correspondiente. Todos los costes totales han sido truncados para una mayor claridad.

Fase	Tiempo	Coste
Reuniones con el tutor	5 horas	101,56€
Investigación	40 horas	812,5€
Pruebas y selección	30 horas	609,37€
Diseño de assets	25 horas	507,81€
Diseño del prototipo	55 horas	1117,18€
Desarrollo del software	80 horas	1625€
Evaluación del prototipo	15 horas	304,68€
Redacción de la memoria	35 horas	710,93€
Total	285 horas	5789€

Tabla 8.1: Resumen de costos

8.2. Costes de generación de imágenes

En este proyecto se utilizó DALL-E 3 como modelo generador de imágenes. DALL-E 3 es un modelo de inteligencia artificial generativa desarrollado por OpenAI. Su API permite generar imágenes 1024x1024 y otras resoluciones mayores, pero en este proyecto se utilizaron las imágenes más pequeñas. Esta API es completamente de pago y su coste por imagen de 1024x1024 generada es de 0.04€.

8.3. Costes de clasificación de imágenes

Para la clasificación de las imágenes generadas, se terminó utilizando Gemini 1.5. Este modelo se eligió como alternativa a Google Cloud Vision por los problemas vistos en el capítulo 5. A pesar de que la prueba gratuita de Google Cloud Vision se compartía con la de Gemini 1.5, se está utilizando la versión “Flash” de Gemini 1.5, que rinde perfectamente y no tiene ningún coste.

8.4. Resumen de costes

A modo de resumen, se ha dejado una tabla en la figura 8.2 que representa todos los gastos de este proyecto.

Tipo de costo	Descripción	Costo unitario	Coste total
Desarrollo	Herramientas utilizadas	5789€	5789€
Generación de imágenes	API DALL-E 3	0.04€	X€
Clasificación de imágenes	API Gemini 1.5	0€	0€

Tabla 8.2: Resumen de costos

Si el proyecto se quisiese mantener en el tiempo y escalarlo el costo aumentaría de manera considerable, puesto que se requerirían muchos más recursos y llamadas a las APIs mencionadas. Ambas APIs tienen sus limitaciones dependiendo de la cantidad de dinero que se aporte a las mismas, así que si el proyecto se hiciese de esta manera y se tuviese un gran número de jugadores, la cantidad de llamadas permitidas según los costes actuales no serían suficientes para funcionar sin errores.

Bibliografía

- [1] The Generative AI Revolution in Games. (2022, November 17). Retrieved from <https://a16z.com/the-generative-ai-revolution-in-games/>
- [2] Repositorio donde está alojado el proyecto. <https://github.com/DiegoHerrera2/TFG-GenerativeAIGame>
- [3] jtsg_. (2023). ChatGPT vs Google Bard—Comparing usage [Post]. Retrieved from https://www.reddit.com/r/ChatGPT/comments/14kavni/chatgpt_vs_google_bard_comparing_usage/
- [4] Rodriguez, M. (s. f.). Research: Quantifying GitHub Copilot's impact on code quality. Retrieved from <https://github.blog/2023-10-10-research-quantifying-github-copilots-impact-on-code-quality/>
- [5] Nguyen, L. X., Sone Aung, P., Le, H. Q., Park, S.-B., & Hong, C. S. (2023). A New Chapter for Medical Image Generation: The Stable Diffusion Method. *2023 International Conference on Information Networking (ICOIN)*, 483-486. <https://doi.org/10.1109/ICOIN56518.2023.10049010>
- [6] NVIDIA DLSS. Retrieved from [https://developer.nvidia.com/rtx/dlss#:~:text=NVIDIA%20DLSS%20\(Deep%20Learning%20Super,in-class%20image%20quality%20and](https://developer.nvidia.com/rtx/dlss#:~:text=NVIDIA%20DLSS%20(Deep%20Learning%20Super,in-class%20image%20quality%20and)
- [7] Página de Steam del videojuego “Scribblenauts”. https://store.steampowered.com/app/218680/Scribblenauts_Unlimited/?l=spanish
- [8] Guodong (Troy) Zhao. (2023, April 5). How Stable Diffusion works, explained for non-technical people. Retrieved from <https://bootcamp.uxdesign.cc/how-stable-diffusion-works-explained-for-non-technical-people-be6aa674fa1d>
- [9] Merritt, R. (2022, April 19). ¿Qué es un Modelo Transformer? Retrieved from <https://la.blogs.nvidia.com/blog/que-es-un-modelo-transformer/>
- [10] Página principal de Unity. <https://unity.com/es>
- [11] Página principal de Stability AI. <https://stability.ai>
- [12] Página principal de Gemini 1.5. <https://www.thinkdiffusion.com>
- [13] ThinkDiffusionXL is the premier Stable Diffusion model. Retrieved from <https://learn.thinkdiffusion.com/thinkdiffusionxl-model-draft/>
- [14] Página principal de DALL-E 3. <https://openai.com/index/dall-e-3/>

- [15] Página principal de Amazon Rekognition. <https://aws.amazon.com/es/rekognition/>
- [16] Página principal de Google Cloud Vision. <https://cloud.google.com/vision/overview/docs?hl=es-419>
- [17] 2D Platformer Controller (Unity 5). Retrieved from https://www.youtube.com/playlist?list=PLFt_AvWsXl0f0hqURlhYoAabKPgRsqjz
- [18] Página principal de Gemini 1.5. <https://deepmind.google/technologies/gemini/>
- [19] Sueldos para el puesto de Ingeniero De Software en España. Retrieved from https://www.glassdoor.es/Sueldos/ingeniero-de-software-sueldo-SRCH_K00,21.htm