

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAISO - CHILE



“GENERACIÓN DE INTERIORES 3D A PARTIR DE
LENGUAJE NATURAL PARA SU USO Y MEJORA EN
MOTORES GRÁFICOS”

GERSON HERNAN MARIHUAN GONCALVES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Sven von Brand
Profesor Correferente: Pedro Godoy

Agosto - 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: GENERACIÓN DE INTERIORES 3D A PARTIR DE LENGUAJE NATURAL PARA SU USO Y MEJORA EN MOTORES GRÁFICOS

Nombre del candidato(a): Gerson Hernan Marihuan Goncalves

Carrera / Grado: Ingeniería Civil en Informática

Campus: Casa Central ; Departamento: Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Sven von Brand, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 02-08-2025 ; Firma: _____

Estudiante o Candidato(a):

Fecha: 02-08-2025 ; Firma: _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATORIA

A mis padres, por su amor, apoyo incondicional y por enseñarme el valor del esfuerzo. Este logro también es suyo.

AGRADECIMIENTOS

Quiero comenzar agradeciendo profundamente a mis padres, por su amor incondicional, su paciencia infinita y su constante apoyo a lo largo de toda mi formación académica. Gracias por confiar en mí incluso en los momentos más difíciles, por impulsarme a seguir adelante cuando dudaba, y por enseñarme, con su ejemplo diario, el valor del esfuerzo, la responsabilidad y la honestidad. Este logro no habría sido posible sin ustedes.

A mis amigos y compañeros de carrera, quienes estuvieron presentes en las largas jornadas de estudio, trabajos en equipo y desafíos que marcaron estos años. Gracias por su compañía, por las risas compartidas y por estar ahí cuando fue necesario apoyarnos mutuamente. Sin duda, esta etapa fue mucho más llevadera y significativa gracias a ustedes.

A la Universidad Técnica Federico Santa María, por abrirme las puertas a una formación rigurosa y de excelencia, y al Departamento de Informática, por entregar una sólida base académica que me permitió crecer profesionalmente y enfrentar con confianza este proyecto. También extendiendo mi gratitud a todos los profesores que fueron parte de este camino y que, directa o indirectamente, contribuyeron a mi desarrollo.

Finalmente, agradezco a todas las personas que, de una u otra manera, me ofrecieron palabras de ánimo, colaboración técnica o simplemente una mano amiga en los momentos clave. Cada gesto fue importante y quedará guardado con gratitud.

Este trabajo es el reflejo de muchos esfuerzos compartidos. A todos, gracias.

RESUMEN

Resumen—Esta memoria presenta el diseño e implementación de un sistema capaz de generar interiores 3D a partir de descripciones en lenguaje natural, con el objetivo de facilitar la creación de entornos virtuales para su uso en motores gráficos. El sistema propuesto utiliza un modelo de lenguaje de gran escala (LLM) para interpretar las descripciones textuales del usuario y extraer relaciones espaciales entre los objetos mencionados. Posteriormente, se aplica un algoritmo de colocación que organiza estos elementos en un espacio tridimensional, respetando las restricciones inferidas. La visualización final se realiza mediante el motor gráfico Godot. Los resultados muestran que el sistema es capaz de producir escenas coherentes, funcionales y visualmente plausibles a partir de entradas expresadas de forma natural, lo que representa un avance hacia la accesibilidad y automatización del diseño de interiores en aplicaciones interactivas.

Palabras Clave—Lenguaje natural; modelos de lenguaje; interiores 3D; motores gráficos; generación automática.

ABSTRACT

Abstract—This thesis presents the design and implementation of a system capable of generating 3D interior scenes from natural language descriptions, aiming to simplify the creation of virtual environments for use in graphics engines. The proposed system leverages a large language model (LLM) to interpret user-provided text and extract spatial relationships between mentioned objects. An object placement algorithm then organizes these elements within a 3D space, respecting the inferred spatial constraints. The final scene is rendered using the Godot graphics engine. Results demonstrate that the system can produce coherent, functional, and visually plausible scenes from natural language inputs, representing a step forward in the accessibility and automation of interior design in

interactive applications.

Keywords—Natural language; language models; 3D interiors; graphics engines; automated generation.

GLOSARIO

3D: Tres Dimensiones. Representación de objetos o entornos que incluyen profundidad además de alto y ancho.

API (Application Programming Interface): Conjunto de reglas que permiten la interacción entre diferentes sistemas de software. En este trabajo, se utiliza para comunicar el frontend con el backend.

AR: Realidad Aumentada. Tecnología que superpone elementos virtuales sobre el entorno físico en tiempo real.

BERT (Bidirectional Encoder Representations from Transformers): Modelo de lenguaje desarrollado por Google, que permite comprender el contexto de una palabra considerando el texto que la rodea a ambos lados.

CSS (Cascading Style Sheets): Lenguaje utilizado para describir la presentación de un documento HTML, en este caso adaptado como formato estructurado para representar escenas.

DFS (Depth-First Search): Algoritmo de búsqueda en profundidad utilizado para ubicar objetos en un espacio tridimensional manteniendo restricciones espaciales.

DI: Departamento de Informática.

Godot: Motor gráfico de código abierto utilizado en esta memoria para renderizar escenas tridimensionales.

GPT (Generative Pre-trained Transformer): Modelo de lenguaje autoregresivo desarrollado por OpenAI, utilizado para generar texto de manera contextual y coherente.

HTTP (HyperText Transfer Protocol): Protocolo de comunicación utilizado en la web y en APIs REST para enviar y recibir datos entre cliente y servidor.

JSON (JavaScript Object Notation): Formato de texto ligero utilizado para estructurar datos de manera legible para humanos y máquinas.

LLM (Large Language Model): Modelos de lenguaje de gran escala entrenados con grandes volúmenes de texto, como GPT y BERT.

NLP (Natural Language Processing): Procesamiento de lenguaje natural, área de la inteligencia artificial centrada en la comprensión e interpretación del lenguaje humano.

OpenGL / DirectX / Vulkan: APIs gráficas utilizadas por motores gráficos para renderizado visual, aprovechando la capacidad de procesamiento del hardware.

Prompt: Instrucción textual diseñada para guiar el comportamiento de un modelo de lenguaje.

Regex: Expresiones regulares. Secuencias de caracteres que forman un patrón de búsqueda, utilizadas para la validación y limpieza de texto.

Renderizado: Proceso de convertir un modelo digital (2D o 3D) en una imagen o animación,

utilizando software especializado.

REST (Representational State Transfer): Arquitectura para desarrollar servicios web. Se basa en operaciones HTTP estándar como GET, POST, PUT y DELETE.

UTFSM: Universidad Técnica Federico Santa María.

VR: Realidad Virtual. Simulación tridimensional de entornos inmersivos con los que el usuario puede interactuar mediante dispositivos especializados.

Índice

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	VI
ÍNDICE DE FIGURAS	X
ÍNDICE DE TABLAS	XI
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Objetivos	5
1.1.1 Objetivos específicos	5
CAPÍTULO 2: MARCO CONCEPTUAL	6
2.1 Procesamiento de lenguaje natural	6
2.2 Modelos de lenguaje de gran escala	9
2.2.1 Limitaciones y Desafíos de los Modelos de Lenguaje de Gran Escala	10
2.2.2 Técnicas de <i>prompting</i>	11
2.3 Motores gráficos	14
2.4 Generación de Entornos 3D a partir de Lenguaje Natural	15
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	21
3.1 Entrada de texto	22
3.2 Construcción del prompt	24
3.2.1 Limpieza y verificación de la salida del <i>LLM</i>	28
3.3 Algoritmo de colocación de objetos	30
3.4 Renderizado	32
3.4.1 Incorporación de modelos 3D	34
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	35
4.1 Procedimiento de evaluación	35
4.2 Instrumentos de evaluación	36
4.3 Análisis de Resultados	38
4.4 Resultados	38

4.4.1	Análisis cuantitativo	39
4.4.2	Análisis cualitativo	41
4.5	Discusión de los resultados	42
CAPÍTULO 5: CONCLUSIONES		44
5.1	Revisión de los objetos alcanzados	44
5.2	Conclusión general	44
5.3	Aspectos técnicos que condicionaron el desarrollo	47
5.4	Trabajo a futuro	48
ANEXOS		50
5.1	Anexo A – Formulario de Evaluación Usado	50
5.2	Anexo B –Resultados Recopilados	52
5.2.1	Evaluador 1	53
5.2.2	Evaluador 2	54
5.2.3	Evaluador 3	55
5.2.4	Evaluador 4	56
5.2.5	Evaluador 5	57
5.2.6	Evaluador 6	58
5.2.7	Evaluador 7	59
5.3	Anexo C – Capturas de Escenas Evaluadas	59

Índice de figuras

1	Árbol del problema, Fuente: Elaboración propia	4
2	Árbol de dependencias para la oración 'Una silla al frente de un escritorio'. Las etiquetas se abrevian de la siguiente manera: <i>Det</i> (determinante), <i>subj</i> (sujeto), <i>prep</i> (preposición) y <i>obj</i> (objeto).	8
3	Árbol Constituyente para la oración ``Una silla al frente de un escritorio". Las etiquetas se abrevian de la misma manera que 2	8
4	Escena generada por <i>WordsEye</i> . El pajarito está en la jaula. La jaula está en la silla. Fuente: [15]	17
5	Dos imágenes que ilustran la representación visual de descripciones de texto en distintos trabajos	18
6	Las cuatro etapas para la generación de escenas en <i>Learning Spatial Know- ledge for Text to 3D Scene Generation</i> . Fuente: [18]	19
7	Flujo general de la solución. Las flechas negras indican el flujo y las flechas transparentes ejemplos del proceso, Fuente: Elaboración propia.	22
8	Dos habitaciones de 500x500 unidades: a la izquierda, una grilla con una dis- tancia de 20 unidades entre puntos; a la derecha, una grilla con una distancia de 100 unidades.	31
9	¿La escena coincide con la descripción que proporcionaste?, Fuente: Elabo- ración propia	39
10	¿Los objetos están alineados correctamente?, Fuente: Elaboración propia	40
11	¿Los objetos están colocados sin colisiones ni intersecciones visualmente problemáticas?, Fuente: Elaboración propia	40
12	¿Los objetos están posicionados de manera lógica según su uso esperado?, Fuente: Elaboración propia	41
13	In the foot of the bed there is a coffee table, and in the opposite corner from where the bed is positioned, there is a lamp.	60
14	I want a room with a double bed, with a nightstand on the left side of the bed and another nightstand on the right. There should also be a black shelf on the wall opposite the bed, and below that shelf there should be a vanity with its own drawer and a desk chair.	60
15	Create an interior scene using the following furniture items: a chair, coffee table, double bed, drawer, fridge, gaming chair, lamp, nightstand, puff, single bed, sofa, stool, table, and a wooden chair. Arrange them naturally in a cozy order	60
16	A bedroom for twin boys	61

17	A room with a double bed in the center, a drawer on the wall in front of the bed, a table next to the drawer, and a chair placed by the table.	61
18	A bed with a nightstand beside it, a table across the room, and a gamer chair in front of the table.	61

Índice de tablas

1	Formulario de evaluación (Parte 1).	51
2	Formulario de evaluación (Parte 2).	52

INTRODUCCIÓN

La generación automática de entornos tridimensionales ha adquirido una relevancia creciente en diversas industrias como los videojuegos, la animación digital, la arquitectura, la simulación, la robótica y la educación, gracias al impacto significativo que estos entornos tienen en la experiencia del usuario y en la eficacia operativa de múltiples aplicaciones. Históricamente, la creación de interiores 3D ha sido un proceso manual altamente especializado, laborioso y costoso, que requiere de herramientas avanzadas, así como un conocimiento técnico profundo en modelado tridimensional, disposición espacial, iluminación, estética y funcionalidad del espacio generado.

Ante este escenario, la posibilidad de generar automáticamente interiores a partir de descripciones en lenguaje natural emerge como una solución innovadora y de gran potencial práctico. Esta alternativa tiene por objetivo permitir que cualquier usuario, incluso aquellos sin formación específica en áreas técnicas o de diseño, pueda describir escenas complejas de manera sencilla e intuitiva. Un sistema inteligente interpreta estas descripciones textuales y las convierte automáticamente en entornos tridimensionales funcionales, estéticamente atractivos y coherentes con la intención original del usuario. Este avance es posible gracias a recientes desarrollos en el campo del procesamiento de lenguaje natural (NLP), en particular con el surgimiento y la evolución de los Modelos de Lenguaje de Gran Escala (LLMs) como GPT y BERT, capaces de comprender y contextualizar instrucciones complejas.

La presente memoria desarrolla una propuesta integral que implica el diseño e implementación de un sistema modular claramente definido, compuesto principalmente por dos componentes: un backend y un frontend. El backend tiene la tarea crucial de interpretar la descripción textual mediante técnicas avanzadas de prompting, aplicadas a modelos LLM especializados, para extraer y organizar información relevante sobre las relaciones espaciales, atributos y ubicaciones de los objetos mencionados por el usuario. Una vez procesada la información textual, el backend utiliza un algoritmo de colocación optimizado que dispone los elementos sobre una grilla tridimensional, asegurando así un posicionamiento lógico y eficiente que respete todas las restricciones espaciales inferidas.

Por otro lado, el frontend se encarga de la visualización y renderizado en tiempo real de la escena generada. Utilizando el motor gráfico Godot Engine, este módulo convierte el esquema espacial creado por el backend en representaciones visuales tridimensionales interactivas. Este enfoque modular no solo facilita el mantenimiento y expansión del sistema, sino que también permite adaptaciones específicas a diferentes tipos de usuarios y contextos de uso,

potenciando su aplicabilidad práctica en múltiples escenarios.

La estructura del documento está organizada para ofrecer al lector un entendimiento progresivo del proyecto. Inicialmente, se presenta una clara definición del problema abordado, junto con los objetivos generales y específicos que guían el desarrollo del trabajo. Posteriormente, se introduce un marco conceptual detallado que abarca aspectos fundamentales relacionados con el procesamiento de lenguaje natural, modelos de lenguaje, motores gráficos y técnicas avanzadas para la generación automatizada de entornos 3D. Seguidamente, se detalla con precisión la propuesta de solución, explicando los componentes técnicos y las decisiones metodológicas adoptadas durante la implementación. A continuación, se describe la metodología utilizada para validar la solución mediante un estudio cualitativo enfocado en la percepción y satisfacción del usuario. Finalmente, se presentan conclusiones sólidas que destacan los principales logros, contribuciones tecnológicas y limitaciones identificadas en el proceso.

En definitiva, esta memoria busca contribuir significativamente a la democratización del diseño 3D, aprovechando el potencial disruptivo de las tecnologías emergentes para transformar y simplificar procesos tradicionalmente complejos, promoviendo así una mayor accesibilidad en la creación y utilización de entornos virtuales.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

La generación de entornos virtuales tridimensionales se ha convertido en una actividad clave para diversas industrias, incluyendo videojuegos, animación digital, arquitectura, robótica y simulación, debido a su impacto directo en la experiencia del usuario final. Históricamente, crear estos entornos implica procesos complejos que requieren herramientas especializadas, además de demandar tiempo significativo y altos costos operativos. Sin embargo, este ámbito ha evolucionado notablemente desde hitos tempranos como la película Toy Story [1], pionera en la animación completamente tridimensional, pasando por videojuegos como The Sims [2], donde el diseño de interiores juega un rol central en la interacción y entretenimiento del usuario, hasta plataformas modernas como RoboTHOR [3], un entorno de simulación abierto diseñado específicamente para entrenar agentes robóticos mediante aprendizaje por simulación.

Dentro de esta área, la generación de interiores 3D es un desafío particular. A diferencia de los entornos abiertos, como paisajes o ciudades, los interiores requieren una disposición cuidadosa de los elementos en un espacio limitado, asegurando funcionalidad, estética y coherencia estructural. Actualmente, este proceso es manual: los diseñadores deben seleccionar cada objeto, definir materiales, configurar la iluminación y asegurarse de que todo encaje correctamente. Aunque existen software avanzados que facilitan estas tareas, la creación de interiores sigue siendo laboriosa y altamente iterativa.

Una alternativa prometedora para enfrentar esta problemática es utilizar la generación automática a partir del lenguaje natural. En lugar de diseñar manualmente cada detalle del espacio, un diseñador podría simplemente escribir una descripción, como por ejemplo: 'una sala de estar moderna con sofás grises, una mesa central de madera y luz cálida', y un sistema inteligente traduciría automáticamente dicha descripción en un modelo tridimensional coherente y listo para su utilización.

Este enfoque, basado en modelos avanzados de procesamiento de lenguaje natural (como GPT o BERT) y técnicas modernas de generación automatizada (redes generativas adversarias, modelos de difusión, entre otras), simplificaría significativamente el proceso de diseño, reduciendo la dependencia de herramientas complejas como Blender, Autodesk Maya o Unity. De esta manera, el diseño tridimensional se vuelve accesible no solo a profesionales especializados, sino también a usuarios sin formación específica en modelado, incluyendo

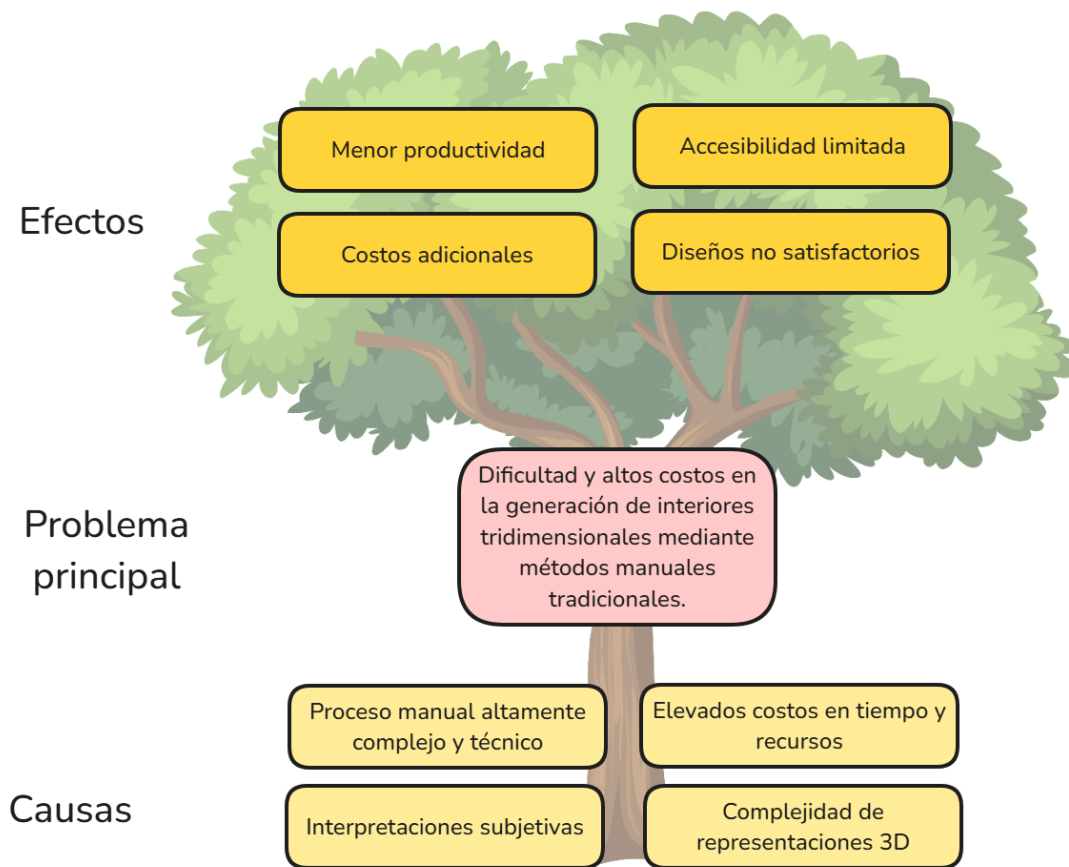


Figura 1: Árbol del problema, Fuente: Elaboración propia

arquitectos, desarrolladores de videojuegos o diseñadores gráficos. Además, este método permitiría la exploración ágil de múltiples alternativas de diseño sin requerir repetidamente el esfuerzo y tiempo asociados al modelado manual.

El problema específico abordado en esta memoria consiste, precisamente, en desarrollar un método para la generación automática de interiores 3D a partir de descripciones en lenguaje natural. Este reto implica desafíos particulares que lo distinguen de otras áreas de generación tridimensional, tales como interpretar con precisión las relaciones espaciales entre objetos, preservar coherencia en el diseño y cumplir restricciones físicas y funcionales inherentes a los espacios de interiores.

1.1. Objetivos

El objetivo general de esta memoria es diseñar e implementar un sistema que permita generar interiores 3D a partir de lenguaje natural para su uso y mejora en motores gráficos.

1.1.1. Objetivos específicos

1. Implementar método para extraer modelos 3D basado en descripciones textuales de escenas.
2. Implementar método para organizar los modelos 3D en una habitación.
3. Evaluar la calidad de las escenas generadas mediante la aplicación de métricas de evaluación.

En la Figura 1 se presenta el árbol del problema que resume lo expuesto anteriormente.

CAPÍTULO 2

MARCO CONCEPTUAL

Para desarrollar un sistema capaz de generar interiores tridimensionales a partir de descripciones en lenguaje natural, es fundamental comprender y analizar una serie de conceptos y tecnologías que constituyen el soporte teórico y técnico de esta memoria. El presente marco conceptual aborda las áreas clave que sustentan la propuesta, proporcionando una visión general de los fundamentos necesarios para comprender el funcionamiento y la relevancia del sistema diseñado.

Se inicia con una revisión del Procesamiento de Lenguaje Natural (*NLP*), explicando su evolución histórica, los principales enfoques utilizados para interpretar el lenguaje humano, y su importancia en la extracción de información estructurada a partir de descripciones textuales. Posteriormente, se presentan los Modelos de Lenguaje de Gran Escala (*LLMs*), detallando sus características, ventajas, limitaciones y técnicas de prompting utilizadas para guiar su comportamiento, aspectos fundamentales para lograr una interpretación adecuada del input del usuario.

A continuación, se introduce el concepto de Motores Gráficos, esenciales para el renderizado y visualización de los interiores generados, explicando sus componentes principales y el rol que cumplen en la materialización de los entornos virtuales. Finalmente, se exploran los métodos existentes para la Generación de Entornos 3D a partir de Lenguaje Natural, destacando los avances y desafíos enfrentados por sistemas anteriores, y justificando así la necesidad de la solución propuesta.

2.1. Procesamiento de lenguaje natural

La historia del procesamiento del lenguaje natural (*NLP*) ha estado marcada por un esfuerzo continuo para enseñar a las máquinas a interpretar, comprender y generar lenguaje humano. Desde sus primeros pasos en la década de 1950, la evolución de esta disciplina ha seguido el desarrollo de la inteligencia artificial y la computación. Uno de los primeros hitos fue el desarrollo de *ELIZA* [4], un programa diseñado para simular una conversación con un terapeuta mediante reglas simples de reconocimiento de patrones. Si bien *ELIZA* no comprendía realmente el lenguaje, su capacidad para mantener una conversación básica sorprendió en su momento y abrió el camino para investigaciones posteriores.

Durante las siguientes décadas, los enfoques se centraron principalmente en el uso de gramáticas formales y reglas lingüísticas explícitas para modelar las estructuras sintácticas del lenguaje. Un ejemplo influyente fueron las gramáticas de Chomsky, que buscaban capturar las relaciones estructurales entre las partes de una oración mediante reglas estrictas. Sin embargo, este enfoque mostró rápidamente sus limitaciones, ya que el lenguaje humano es extremadamente flexible, ambiguo y altamente dependiente del contexto, lo que dificulta su descripción completa mediante reglas fijas.

En este contexto, surgieron herramientas de apoyo como las expresiones regulares (*Regex*), introducidas por Stephen Cole Kleene [5] como una notación matemática para describir lenguajes formales. Posteriormente, en 1968, Ken Thompson [6] implementó las *Regex* en un editor de texto, marcando su primera aplicación práctica en el procesamiento de texto. Las expresiones regulares permiten definir patrones de búsqueda en textos, facilitando tareas como la extracción de información, la limpieza de datos y la validación de estructuras textuales. Aunque simples en apariencia, las *Regex* ofrecen un enorme poder al permitir identificar elementos relevantes en grandes volúmenes de texto de manera rápida y eficiente. En esta memoria, su uso resulta fundamental para la limpieza y validación de la salida generada por los modelos de lenguaje, asegurando un formato estructurado y procesable para las siguientes etapas del sistema.

Algunos trabajos han aplicado reglas gramaticales para capturar relaciones espaciales entre objetos en descripciones textuales. En [7] proponen cuatro algoritmos que analizan estructuras de árboles de dependencia y árboles de constituyentes. Los árboles de dependencia de la Figura 2 se enfocan en identificar las relaciones gramaticales directas entre palabras dentro de una oración, como el vínculo entre un verbo y su sujeto o complemento. Por otro lado, los árboles de constituyentes de la Figura 3 presentan una representación jerárquica de la oración, destacando sus componentes gramaticales, como frases nominales y verbales, lo que permite un análisis más detallado de la estructura global del lenguaje.

A partir de la década de 1990, el campo dio un giro hacia métodos estadísticos gracias a la disponibilidad de grandes cantidades de datos textuales y el aumento en la capacidad computacional. Los algoritmos empezaron a calcular probabilidades para determinar cuál era la palabra o estructura más probable en un contexto dado. Por ejemplo, los modelos *n-grams* [8] eran capaces de predecir la próxima palabra en una oración analizando secuencias de palabras previas. Aunque estos modelos funcionaban razonablemente bien para tareas sencillas, carecían de una verdadera comprensión del lenguaje, ya que solo se basaban en frecuencias estadísticas.

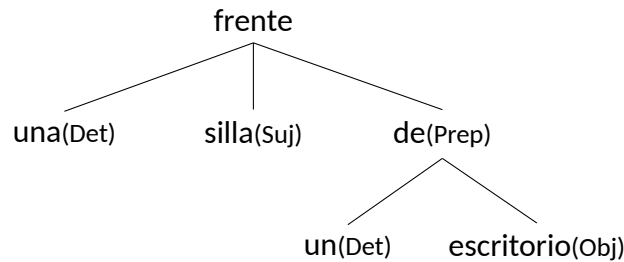


Figura 2: Árbol de dependencias para la oración 'Una silla al frente de un escritorio'. Las etiquetas se abrevian de la siguiente manera: *Det* (determinante), *subj* (sujeto), *prep* (preposición) y *obj* (objeto).

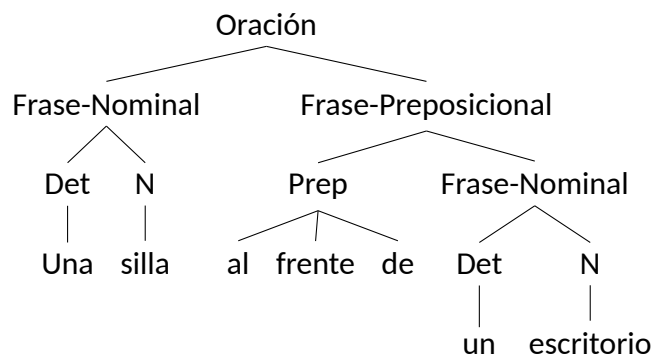


Figura 3: Árbol Constituyente para la oración "Una silla al frente de un escritorio". Las etiquetas se abrevian de la misma manera que 2

El avance más significativo llegó con el surgimiento de las redes neuronales profundas a principios de la década de 2010. Estos modelos podían aprender representaciones matemáticas (vectores) complejas de palabras y frases, permitiendo capturar relaciones semánticas. Por ejemplo, las palabras "rey" y "reina" estarían cerca en un espacio vectorial, mostrando una relación semántica evidente. Modelos como *Word2Vec* [9] revolucionaron el campo al introducir estas técnicas.

Finalmente, la llegada de los *Transformers* [10] marcó una nueva era en el procesamiento del lenguaje natural. Los Transformers permitieron el desarrollo de modelos como *BERT* [11] y *GPT* [12], capaces de comprender el contexto completo de una oración, no solo de izquierda a derecha, sino en ambas direcciones. Esto permitió que las máquinas no solo generaran texto coherente, sino que también respondieran preguntas, resumieran información y comprendieran instrucciones complejas, como "describe una habitación moderna con muebles minimalistas y luz natural".

2.2. Modelos de lenguaje de gran escala

Los Modelos de Lenguaje de Gran Escala (LLMs) son sistemas avanzados basados en arquitecturas de redes neuronales profundas, entrenados con enormes volúmenes de datos textuales provenientes de diversas fuentes como libros, artículos, páginas web y diálogos humanos. Su objetivo principal es comprender, inferir, transformar y generar texto de manera coherente, natural y contextualmente adecuada. Estos modelos han transformado radicalmente el campo del procesamiento de lenguaje natural Subsección 2.1, proporcionando nuevas posibilidades para automatizar, mejorar y enriquecer tareas que anteriormente requerían intervención humana especializada. Gracias a su capacidad para adaptarse a diferentes tareas, han permitido abordar problemas complejos en áreas como la educación, el derecho, la programación y la medicina, entre muchas otras .

Los modelos de lenguaje a gran escala se caracterizan principalmente por estar constituidos por miles de millones de parámetros entrenables. Esta enorme cantidad de parámetros les permite aprender representaciones profundas y capturar patrones lingüísticos de alta complejidad a múltiples niveles: sintáctico (estructura gramatical), semántico (significado) y pragmático (uso contextual del lenguaje). Estas capacidades de representación sofisticada permiten que los LLMs no sólo entiendan matices sutiles de diferentes lenguas y registros de comunicación, sino también que generalicen conocimiento entre dominios dispares, logrando así un desempeño sobresaliente en tareas no vistas durante su entrenamiento inicial.

Entre las características fundamentales de estos modelos destacan:

- **Generalización:** Capacidad para enfrentarse a una variedad amplia de tareas lingüísticas sin necesidad de un entrenamiento específico adicional.
 - *Ejemplo:* Un mismo modelo, como GPT-3.5, puede redactar cartas, escribir poesía, generar código de programación o mantener conversaciones informales de manera competente.
- **Adaptabilidad:** Habilidad para ajustarse rápidamente a diferentes contextos temáticos o estilos comunicativos mediante instrucciones breves, conocidas como “prompts”.
 - *Ejemplo:* Si se le indica al modelo “actúa como un experto en derecho internacional”, podrá modular sus respuestas para reflejar conocimientos técnicos y terminología específica del área.

- **Generación Coherente y Contextualizada:** Competencia para producir textos fluidos y cohesivos que mantienen coherencia temática a lo largo de extensos diálogos o documentos.
 - *Ejemplo:* Al solicitar un ensayo sobre las ventajas y desventajas de la inteligencia artificial, el modelo puede ofrecer una respuesta bien organizada, equilibrada y argumentada.
- **Transferencia de Conocimiento:** Capacidad para aplicar habilidades adquiridas en un contexto a tareas nuevas o similares, reduciendo la necesidad de datos de entrenamiento adicionales.

Actualmente, algunos de los modelos más representativos en esta categoría incluyen GPT-3.5 y GPT-4 de OpenAI, PaLM de Google, LLaMA de Meta, y Claude desarrollado por Anthropic, todos ellos marcando hitos significativos en la evolución de las capacidades de procesamiento del lenguaje [13].

2.2.1. Limitaciones y Desafíos de los Modelos de Lenguaje de Gran Escala

A pesar de sus impresionantes habilidades, los Modelos de Lenguaje de Gran Escala enfrentan diversas limitaciones y desafíos que es imprescindible considerar para su implementación responsable:

- **Sesgos en los datos:** Al ser entrenados en datos de internet y otras fuentes humanas, los LLMs absorben sesgos culturales, de género, raciales y políticos existentes, lo que puede llevarlos a reproducir o amplificar estereotipos y prejuicios en sus respuestas.
- **Alucinaciones:** Los LLMs tienen la tendencia a inventar hechos o generar respuestas incorrectas con una apariencia muy convincente, fenómeno conocido como “alucinaciones”. Esto representa un riesgo particular en contextos donde la precisión factual es crítica, como en entornos médicos, legales o de investigación científica.
- **Interpretabilidad Limitada:** Dado que los LLMs son cajas negras de altísima complejidad, resulta extremadamente difícil para los investigadores y usuarios finales comprender o justificar por qué un modelo generó una respuesta determinada, lo que afecta la confianza en sus resultados.

- **Dependencia de Recursos Computacionales:** Entrenar, ajustar o incluso desplegar LLMs requiere infraestructuras computacionales avanzadas y costosas, limitando su accesibilidad a instituciones o empresas con altos recursos económicos.
- **Cuestiones Éticas y de Responsabilidad:** El uso de LLMs plantea preocupaciones importantes sobre la privacidad de los datos, la creación de desinformación, la propiedad intelectual de los contenidos generados y la responsabilidad legal por los daños que puedan derivarse de su uso inadecuado.

Reconocer y enfrentar estos desafíos de manera activa es esencial para garantizar que la utilización de los modelos de lenguaje se realice de forma segura, ética y beneficiosa para la sociedad. Adoptar buenas prácticas como la moderación de salidas, la evaluación crítica de resultados y la implementación de políticas de gobernanza tecnológica son pasos fundamentales para maximizar el valor de estos sistemas avanzados mientras se minimizan sus riesgos potenciales [13].

2.2.2. Técnicas de *prompting*

El término *prompting* se refiere al método mediante el cual se condiciona o guía a un modelo de lenguaje para que realice tareas específicas sin necesidad de modificar sus pesos o parámetros internos. En esencia, se trata de un proceso de ingeniería de instrucciones en el que se construyen entradas cuidadosamente diseñadas para que el modelo interprete la tarea de manera adecuada. Dado que los LLMs interpretan y responden en función del contexto que reciben en el *prompt*, su diseño apropiado influye de manera significativa en la calidad, precisión, coherencia, creatividad y relevancia de las respuestas generadas. Dominar técnicas de *prompting* resulta fundamental, ya que permite aprovechar todo el potencial de los LLMs sin necesidad de recurrir a reentrenamientos complejos o costosos procesos de ajuste fino (*fine-tuning*), facilitando así la implementación de soluciones ágiles, escalables y altamente eficientes [12].

Entre las técnicas más utilizadas destacan:

1. **Role Prompting (Prompt basado en roles)**

Consiste en asignar explícitamente un “rol” o identidad específica al modelo, condicionando así sus respuestas hacia una perspectiva o estilo determinado. Esta técnica

permite adaptar el tono, el vocabulario, la profundidad y el enfoque de las respuestas según la necesidad del usuario.

Ejemplo:

Prompt: “Eres un crítico gastronómico experto. Describe brevemente el plato de una manera sofisticada: pizza margarita.”

Respuesta esperada: “La pizza margarita, sencilla en su composición pero sublime en sabor, combina la frescura aromática de la albahaca, la delicada acidez del tomate San Marzano, y la cremosidad ligera de la mozzarella fresca sobre una base de masa fina y crujiente, horneada hasta alcanzar un equilibrio perfecto.”

2. **Zero-shot Prompting (Prompt sin ejemplos)**

En esta modalidad, se presenta una tarea específica sin proporcionar ejemplos previos, confiando en la capacidad de generalización y comprensión del modelo. Es ideal para tareas simples o cuando se desea evaluar la comprensión natural del modelo sobre una instrucción.

Ejemplo:

Prompt: “Enumera tres beneficios ambientales de reciclar plástico.”

Respuesta esperada:

- a) Reducción del volumen de residuos en vertederos y océanos.
- b) Disminución del consumo energético en comparación con la producción de plástico nuevo.
- c) Reducción en la emisión de gases de efecto invernadero.

3. **Few-shot Prompting (Prompt con pocos ejemplos)**

Implica incluir uno o varios ejemplos de preguntas y respuestas antes de plantear la tarea solicitada. Esta técnica sirve para orientar al modelo mediante patrones explícitos, aumentando la precisión y pertinencia de las respuestas.

Ejemplo:

Prompt:

P: ¿Cuál es la capital de Francia? R: París

P: ¿Cuál es la capital de Italia? R: Roma

P: ¿Cuál es la capital de Japón? R:

Respuesta esperada: Tokio

4. **Structured Prompting**(Prompt estructurado)

Esta técnica consiste en solicitar respuestas bajo un formato específico y predefinido, como tablas, listas, o estructuras tipo *JSON* o *XML*, facilitando así su procesamiento automático posterior.

Ejemplo:

Prompt: “Extrae la siguiente información estructurada del texto: 'Ana vive en Valparaíso y tiene 28 años. Es ingeniera informática.' Utiliza formato *JSON*.”

Respuesta esperada:

```
{
  "nombre": "Ana",
  "ciudad": "Valparaíso",
  "edad": 28,
  "profesion": "ingeniera informática"
}
```

5. **Chain-of-Thought Prompting** (Cadena de Pensamiento)

Esta técnica solicita al modelo que razone paso a paso antes de dar una respuesta final, fomentando un pensamiento más estructurado y reduciendo errores lógicos o conceptuales.

Ejemplo:

Prompt: “Si Juan tiene tres manzanas y compra dos más, ¿cuántas manzanas tendrá? Razona paso a paso.”

Respuesta esperada: “Juan comienza con 3 manzanas. Compra 2 manzanas adicionales. En total, tiene $3 + 2 = 5$ manzanas.”

6. **Self-Consistency Prompting** (Consistencia Interna)

Implica generar múltiples cadenas de razonamiento y luego seleccionar la respuesta más frecuente o lógica entre ellas, mejorando la exactitud general de la salida al minimizar respuestas aleatorias o inconsistentes.

Ejemplo:

Prompt: “¿Cuál es la suma de 48 y 27? Explica tu razonamiento.”

Respuesta esperada:

- "48 + 20 = 68; 68 + 7 = 75."
- "Sumando primero las decenas: 40 + 20 = 60; luego las unidades: 8 + 7 = 15; sumando 60 + 15 = 75."
- "48 más 2 es 50; 50 más 25 es 75."

Al compararlas, se elegiría "75" como la respuesta final más consistente.

7. Multimodal Prompting (Prompting multimodal)

Cuando los LLMs están integrados con capacidades multimodales, es posible incorporar imágenes, tablas u otros formatos no textuales dentro del *prompt*, enriqueciendo las entradas y permitiendo respuestas aún más completas.

Ejemplo:

Prompt: "Observa esta gráfica de temperaturas y describe la tendencia principal."

Respuesta esperada: "La gráfica muestra un aumento gradual de la temperatura media mensual de enero a julio, seguido de una disminución progresiva hasta diciembre."

Estas técnicas permiten maximizar la efectividad de los modelos de lenguaje de gran escala, logrando respuestas más precisas, relevantes, consistentes y adaptadas a diversas necesidades. Desde aplicaciones sencillas como asistentes virtuales, hasta sistemas complejos de análisis de datos o generación de conocimiento, implementar estrategias adecuadas de prompting no solo incrementa la calidad de las respuestas obtenidas, sino que también potencia significativamente las capacidades prácticas de los LLMs en contextos reales de uso. Asimismo, el aprendizaje y dominio progresivo de estas técnicas abre la puerta a nuevos niveles de innovación en el diseño de soluciones basadas en inteligencia artificial [12].

2.3. Motores gráficos

Un motor gráfico es un software especializado que proporciona las herramientas y funciones necesarias para la creación, manipulación y renderizado de entornos y objetos tridimensionales o bidimensionales en aplicaciones interactivas. Su función principal es facilitar el desarrollo eficiente de contenido visual para videojuegos, simuladores, aplicaciones de realidad virtual (VR), realidad aumentada (AR) y otras experiencias digitales.

Los motores gráficos permiten a los desarrolladores centrarse en el diseño y la lógica de sus aplicaciones sin tener que programar desde cero aspectos complejos como el renderizado de gráficos, la simulación de física o la gestión de animaciones.

Un motor gráfico suele estar compuesto por varios módulos clave que trabajan de manera conjunta para generar el contenido visual y físico de una aplicación:

- **Renderizado:** Es responsable de convertir las descripciones matemáticas de objetos tridimensionales en imágenes 2D que se muestran en pantalla. Utiliza técnicas como la rasterización o el trazado de rayos (ray tracing).
- **Motor de Física:** Simula el comportamiento de los objetos en el espacio tridimensional, incluyendo colisiones, gravedad, y dinámicas de cuerpos rígidos.
- **Gestión de Escenas:** Organiza y administra la disposición de los objetos en el espacio tridimensional, permitiendo que los desarrolladores estructuren el contenido visual de manera jerárquica.
- **Animaciones:** Permiten dar vida a los objetos y personajes mediante la interpolación de movimientos y transformaciones a lo largo del tiempo.
- **Sistema de audio:** Facilita la integración de efectos de sonido y música, muchas veces con capacidades de audio espacial para experiencias inmersivas.
- **Herramientas de desarrollo:** Interfaces gráficas y entornos de desarrollo visual que simplifican el trabajo de diseño y programación.

El correcto entendimiento de estos módulos permite optimizar el desarrollo del proyecto a elaborar, potenciando la eficiencia del trabajo creativo y técnico de los equipos de desarrollo.

Además, los motores gráficos suelen utilizar APIs gráficas para interactuar con el hardware y renderizar las imágenes. Algunas de las más comunes incluyen DirectX, OpenGL y Vulkan [14].

Algunos ejemplos destacados de motores gráficos son Unreal Engine, Unity y Godot, cada uno con sus propias características y ventajas.

2.4. Generación de Entornos 3D a partir de Lenguaje Natural

La generación de entornos 3D a partir de descripciones en lenguaje natural ha cobrado relevancia en los últimos años debido al avance en inteligencia artificial, aprendizaje profundo y motores gráficos. Esta tecnología tiene aplicaciones en diversos campos, como videojuegos,

simulaciones, arquitectura y realidad virtual, donde la capacidad de traducir descripciones textuales en representaciones tridimensionales automatizadas mejora significativamente la eficiencia y accesibilidad del diseño de escenas.

Uno de los sistemas pioneros en esta área fue *WordsEye* [15], desarrollado a principios de la década del 2000. Este sistema permitía la conversión automática de texto en escenas 3D utilizando una base de datos extensa de modelos tridimensionales y técnicas de procesamiento del lenguaje natural. Gracias a este enfoque, *WordsEye* facilitó a usuarios sin experiencia en modelado 3D la creación de escenas complejas mediante simples descripciones textuales, tal como se ilustra en 4.

El proceso de generación de escenas en *WordsEye* se basaba en los siguientes componentes:

- **Análisis del Lenguaje Natural:** El sistema descomponía las descripciones textuales en estructuras sintácticas y semánticas, identificando objetos, acciones y relaciones espaciales entre ellos.
- **Base de Datos de Modelos 3D:** Contaba con un amplio repertorio de modelos tridimensionales predefinidos, que incluían información sobre su geometría, poses posibles y propiedades funcionales.
- **Motor de Representación:** Una vez interpretado el texto, el sistema seleccionaba los modelos adecuados y los ubicaba en la escena de acuerdo con las relaciones espaciales y acciones especificadas en la descripción.

A pesar de su innovación, *WordsEye* enfrentó varios desafíos técnicos y conceptuales. Por ejemplo, uno de sus principales problemas era en la interpretación del lenguaje natural. Traducir descripciones en lenguaje natural a entornos tridimensionales implica superar varios obstáculos inherentes al procesamiento del lenguaje natural:

- **Ambigüedad y Polisemia:** Muchas palabras tienen significados múltiples según el contexto. Por ejemplo, 'banco' puede referirse a un asiento o a una institución financiera. Resolver estas ambigüedades requiere un análisis contextual profundo.
- **Variabilidad en la Expresión:** Una misma escena puede describirse de diversas formas. El sistema debe reconocer y comprender esta diversidad para evitar interpretaciones erróneas. Por ejemplo, 'Cuatro sillas alrededor de una mesa' y 'Una mesa con dos sillas en frente y dos sillas detrás'

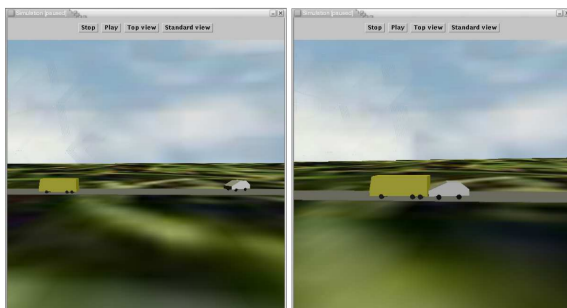


Figura 4: Escena generada por WordsEye. El pajarito está en la jaula. La jaula está en la silla.
Fuente: [15]

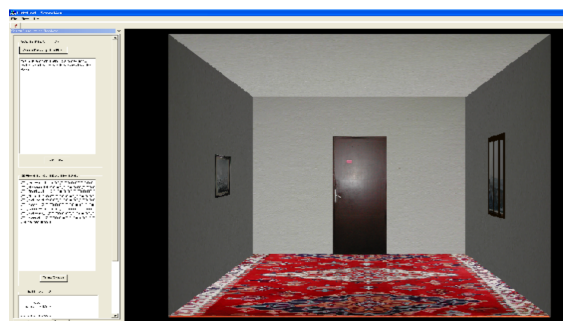
- **Implicaturas y Contexto:** Los hablantes suelen omitir detalles que se consideran obvios. Frases como 'Coloca la taza en la mesa' suponen que la mesa tiene espacio disponible, pero esto no está explicitado en el texto.
- **Complejidad de las Relaciones Espaciales:** Términos como 'a la izquierda de' o 'detrás de' dependen del punto de vista. Sin un modelo preciso de perspectiva, el sistema podría interpretar incorrectamente la ubicación de los objetos.

Posteriormente, se desarrollaron sistemas con enfoques más específicos en lugar de crear escenas de uso general. Un ejemplo destacado es el *CarSim System* [16], diseñado para generar simulaciones 3D de accidentes de tráfico a partir de informes escritos en lenguaje natural, como se observa en Figura 5a. Este sistema procesa reportes de accidentes, extrae la información relevante y la convierte en una descripción formal que luego se utiliza para crear y animar la escena correspondiente en 3D

Otro caso es el trabajo de [17], que se centra en la reconstrucción de escenas del crimen a partir de descripciones textuales breves. Este enfoque permite indexar y consultar fotografías de escenas del crimen de manera más eficiente, facilitando la reconstrucción visual basada en la información proporcionada en los informes, también ilustrado en la Figura 5b.



(a) Accidente automovilístico representado en el trabajo de [16].



(b) Escena del crimen renderizada en [17]: “There is a carpet on the floor and a door on the front wall. A window is on the right wall. There is a painting on the left wall.”

Figura 5: Dos imágenes que ilustran la representación visual de descripciones de texto en distintos trabajos

En el ámbito del diseño de interiores, la inteligencia artificial incorporada (*embodied AI*) ha impulsado el desarrollo de sistemas especializados en la generación de entornos interiores. Un ejemplo destacado es el trabajo de [18], que implementa un enfoque innovador al emplear técnicas de *Machine Learning* para aprender de una base de datos de escenas y utilizar ese conocimiento para inferir las posiciones más adecuadas de los objetos. Este sistema consta de cuatro etapas secuenciales (ver Figura 6):

1. **Procesamiento de texto:** El proceso comienza con una descripción textual proporcionada por el usuario, que detalla los objetos y sus relaciones dentro de la escena.
2. **Plantilla de escena:** A partir del texto de entrada, se crea una representación estructurada llamada “plantilla de escena”. Esta plantilla identifica los objetos mencionados y las relaciones espaciales explícitas entre ellos, sirviendo como un esquema inicial de la escena.
3. **Escena geométrica:** Utilizando conocimientos espaciales aprendidos de una base de datos de escenas 3D existentes, el sistema infiere las posiciones y orientaciones más probables de los objetos. Se aplican restricciones geométricas y espaciales para transformar la plantilla de escena en una disposición geométrica coherente.
4. **Escena 3D:** Finalmente, se seleccionan modelos 3D específicos para cada objeto y se colocan según la disposición determinada en la etapa anterior, generando una escena 3D completa y visualmente coherente que refleja la descripción textual original.

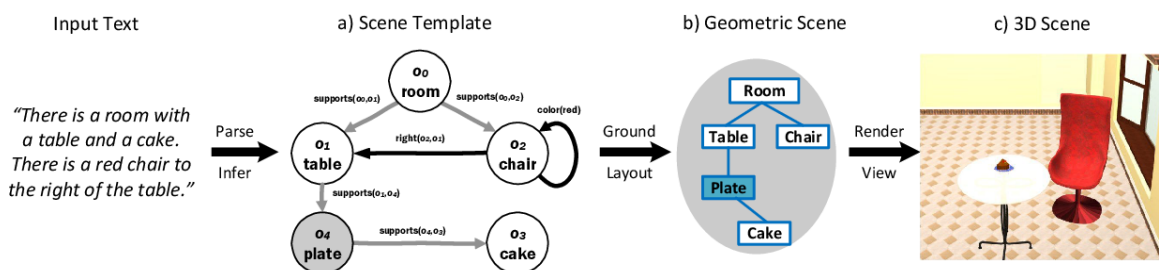


Figura 6: Las cuatro etapas para la generación de escenas en *Learning Spatial Knowledge for Text to 3D Scene Generation*. Fuente: [18]

Aunque este enfoque ofrece ventajas significativas respecto a trabajos anteriores, aún presenta desafíos. La implementación del sistema es compleja, y la etapa de procesamiento de texto se basa en reglas predefinidas, lo que limita la capacidad de generalización y comprensión profunda del lenguaje. Los sistemas de generación de lenguaje natural basados en reglas suelen carecer de creatividad y flexibilidad, produciendo resultados predecibles y enfrentando dificultades para manejar la ambigüedad y comprender el contexto.

Con la llegada de los Modelos de Lenguaje a Gran Escala (LLMs), se han desarrollado varios enfoques para la generación de interiores en 3D, aprovechando su capacidad para procesar y razonar sobre relaciones espaciales. Entre estos, destacan dos trabajos: LayoutGPT [19] y LLplace [20], ambos utilizando LLMs como componentes centrales en la generación de diseños, aunque con diferencias significativas en su metodología y ejecución.

LayoutGPT sigue un proceso en dos etapas para generar diseños de habitaciones en 3D a partir de descripciones textuales:

- **Generación de diseño:** A partir de una entrada en lenguaje natural, LayoutGPT genera un diseño estructurado en formato CSS, donde cada objeto recibe un tamaño y una posición específica. Este método explota la capacidad de los LLMs para procesar datos estructurados, tratando la generación de diseño como una tarea de codificación.
- **Generación de la escena final:** El diseño generado se utiliza como entrada para modelos de síntesis visual, garantizando la corrección espacial y mejorando la precisión.

El uso de un formato similar a CSS permite a *LayoutGPT* codificar la información espacial de manera estructurada, facilitando su interpretación por los LLMs.

Por otro lado, *LLplace* adopta un enfoque distinto al ajustar finamente un LLM de código abierto (*Llama3*) en lugar de depender del aprendizaje en contexto. Mientras que LayoutGPT

se basa en ejemplos previos (*Few-Shot*) para garantizar la precisión del diseño, *LLplace* elimina esta necesidad haciéndolo más adaptable a diferentes configuraciones de habitaciones.

Si bien tanto *LayoutGPT* como *LLplace* demuestran el potencial de los LLMs en la generación de escenas 3D, presentan diferencias clave en su enfoque:

- **LayoutGPT** utiliza el aprendizaje en contexto y un formato basado en CSS, lo que lo hace modular y fácilmente integrable con modelos generativos existentes.
- **LLplace** emplea ajuste fino, lo que le permite autocorregir errores y modificar diseños en tiempo real mediante instrucciones en lenguaje natural.

Otro enfoque relevante en la generación de entornos 3D es *Holodeck* [21], un sistema basado en modelos de lenguaje que permite la creación automatizada de escenas completas a partir de descripciones en lenguaje natural. A diferencia de *LayoutGPT* y *LLplace*, que utilizan LLM para generar todo el esquema de la habitación *Holodeck* utiliza el LLM para inferir sobre las relaciones espaciales de los objetos y luego utiliza un algoritmo *greedy* para cumplir estas condiciones.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

Para abordar el problema planteado, se propone una arquitectura modular inspirada en enfoques comunes del desarrollo web, compuesta por dos componentes principales e independientes llamadas backend y frontend.

El **backend** tiene como responsabilidad principal el procesamiento del lenguaje natural y la generación del esquema lógico de la escena. Su función es interpretar las descripciones textuales proporcionadas por el usuario, identificar los objetos mencionados, extraer sus características y relaciones espaciales, y convertir esta información en un formato estructurado. Este esquema servirá como base para la disposición de los objetos en la escena final.

El **frontend**, por otro lado, se encarga de la representación gráfica del esquema generado por el backend. Utilizando el motor gráfico Godot, este módulo renderiza la habitación en tiempo real, permitiendo visualizar la escena generada a partir del texto.

Ambos módulos están diseñados para integrarse de manera fluida, lo que permite una transición eficiente entre el análisis del texto y la representación visual. Esta arquitectura modular no solo facilita el desarrollo y mantenimiento del sistema, sino que también garantiza su escalabilidad y adaptabilidad ante futuras mejoras o expansiones.

Un esquema del flujo de la solución propuesta se presenta en la figura 7 y se describe a continuación:

1. Entrada del usuario:

El proceso comienza con una descripción textual de la escena, por ejemplo, una habitación con una cama y una silla de escritorio. Esta información se envía al backend a través de una API REST.

2. Construcción del prompt:

El backend genera un prompt con las instrucciones y la descripción de la habitación.

3. Inferencia de relaciones espaciales:

El modelo de lenguaje procesa el Prompt y devuelve una salida estructurada. Esta salida contiene las condiciones espaciales que deben cumplir los objetos para reproducir la escena según lo descrito.

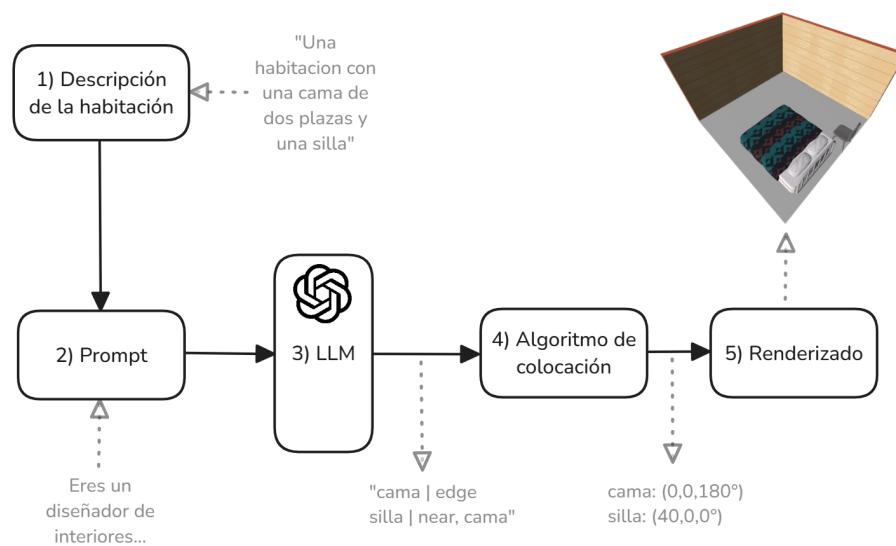


Figura 7: Flujo general de la solución. Las flechas negras indican el flujo y las flechas transparentes ejemplos del proceso, Fuente: Elaboración propia.

4. **Algoritmo de colocación de objetos:** Posteriormente, el sistema aplica un algoritmo de búsqueda en profundidad (*Depth-First Search*) para ubicar cada objeto dentro del espacio, asegurando que se cumplan las relaciones espaciales inferidas en el paso anterior. Al finalizar, se genera una solución completa con las posiciones y orientaciones de cada objeto.
5. **Renderizado de la escena:** Finalmente, la solución generada es enviada al módulo frontend, donde se instancia cada objeto en la escena 3D según los atributos definidos (posición, orientación, tipo).

A continuación se detallara cada modulo y como realizan el procesamiento.

A continuación se detallara cada etapa del proceso.

3.1. Entrada de texto

El sistema inicia en la aplicación de visualización, donde el usuario introduce una descripción textual que detalla los objetos que componen la habitación, así como sus posibles relaciones espaciales. Esta entrada puede variar en complejidad, desde configuraciones mínimas hasta descripciones con múltiples elementos y restricciones espaciales explícitas. Por ejemplo:

"Una cama con dos mesas de noche a los lados, un escritorio frente a la cama y una silla frente al escritorio".

Una vez enviada la descripción, la aplicación genera una solicitud HTTP conforme al estándar REST, dirigida al endpoint correspondiente del backend encargado del procesamiento del texto. Esta solicitud incluye un cuerpo en formato JSON que encapsula los datos necesarios para generar el esquema espacial. A continuación, se presenta la estructura general de dicha solicitud:

La aplicación envía una solicitud HTTP al servidor backend utilizando el método POST. A continuación se muestra la estructura general de esta solicitud:

```
1 POST /get_scene_template HTTP/1.1
2 Host: 127.0.0.1:5000
3 Content-Type: application/json
4
5 {
6   "description": "Una cama con dos mesas de noche a los lados, un
7     escritorio frente a la cama y una silla frente al escritorio"
8 }
```

Listing 1: Ejemplo de solicitud HTTP con cuerpo JSON

Este ejemplo representa una estructura mínima y clara, donde el campo *description* contiene la entrada del usuario. En versiones futuras del sistema, este esquema podría complementarse con campos como *language*, *session_id* o *user_id*, los cuales permitirían una personalización más precisa del procesamiento, así como una trazabilidad más robusta de las solicitudes.

Desde el punto de vista funcional, esta etapa implica la construcción y envío de una petición POST con las cabeceras adecuadas, especificando el tipo de contenido.

Cabe destacar que en esta etapa no se realiza validación semántica ni corrección gramatical del texto recibido. El diseño del sistema delega completamente esta responsabilidad al modelo de lenguaje, lo que permite una mayor flexibilidad expresiva por parte del usuario. Sin embargo, también implica desafíos significativos, como el manejo de ambigüedades o descripciones incompletas, que deberán ser gestionados adecuadamente en las etapas posteriores del flujo.

3.2. Construcción del prompt

El lenguaje natural permite a los usuarios describir espacios y relaciones espaciales de una forma intuitiva, rica y flexible. Sin embargo, esta flexibilidad también introduce ambigüedad y variabilidad, lo que representa un reto considerable al intentar traducir esas descripciones en instrucciones estructuradas para un sistema computacional. Para abordar este desafío, se emplea un modelo de lenguaje de gran escala (LLM), el cual es capaz de interpretar estas descripciones textuales y convertirlas en un conjunto de restricciones espaciales comprensibles para el sistema. Estas restricciones se utilizan luego para construir un esquema lógico que sirve como base para la generación y visualización de las escenas.

La clave para que el modelo produzca resultados útiles y coherentes reside en la elaboración de un prompt cuidadosamente diseñado. Este prompt actúa como interfaz entre el usuario y el modelo, proporcionando las instrucciones necesarias para guiar la generación de la salida. En el sistema propuesto, el prompt se organiza en cinco componentes fundamentales basado en el trabajo de [21]:

1. **Asignación de rol al modelo:** Se le indica explícitamente al modelo que adopte el rol de un diseñador de interiores experimentado. Bajo esta premisa, su tarea principal consiste en organizar los objetos descritos por el usuario dentro de un espacio definido, utilizando una serie de restricciones espaciales. Este enfoque corresponde a una técnica conocida como *role prompting*, una forma de *zero-shot prompting* en la cual no se proporcionan ejemplos concretos, pero se guía al modelo mediante instrucciones detalladas para que genere una respuesta especializada.
2. **Definición de restricciones:** El prompt proporciona un conjunto claro y detallado de restricciones que el modelo debe utilizar para describir las relaciones espaciales entre los objetos. Estas restricciones se agrupan en dos grandes categorías:
 - **Restricciones Globales:** Determinan la ubicación general del objeto en relación con la habitación:
 - **edge** : El objeto se ubica cerca de una pared o en el borde del espacio.
 - **middle** : El objeto está alejado de los bordes, más hacia el centro de la habitación.
 - **Restricciones Locales:** Describen cómo se relaciona un objeto con otros elementos en la escena:
 - **Distancia:** **near** (cerca, entre 50 cm y 150 cm), **far** (lejos, más de 150 cm).

- **Posición:**
 - `left of`: El objeto se ubica a la izquierda de otro.
 - `right of`: El objeto se ubica a la derecha de otro.
 - `in front of`: Posicionado frente al otro objeto, según su orientación.
 - `side of`: A un lado (izquierda o derecha) del otro objeto, sin especificar cuál.
 - `around`: Alrededor del objeto, como sillas en torno a una mesa.
- **Alineación:** `center aligned` (alineación central con respecto a otro objeto).
- **Orientación:** `face to` (orientado hacia el centro de otro objeto).

3. **Formato de salida:** Para garantizar una salida uniforme y fácil de procesar, se exige que el modelo siga un formato de salida rígido. Cada línea representa un objeto y debe seguir la estructura, ejemplo:

```
object_name | global constraint | [local constraint 1] | ...  
sofa-0 | edge  
coffee table-0 | middle | near, sofa-0 | in front of, sofa-0
```

Este formato facilita el análisis automático de los resultados, ya que cada restricción está claramente delimitada y asociada con un objeto. La generación se realiza sin ejemplos previos, por lo tanto, también puede considerarse una aplicación de *structured zero-shot prompting*, en la que el modelo sigue un formato predefinido basado únicamente en las instrucciones.

4. **Guías de construcción:** Para aumentar la coherencia y utilidad de la salida generada por el modelo, se introducen una serie de reglas adicionales:

- Iniciar con un objeto ancla.
- Colocar primero los objetos más grandes.
- Asegurar que los objetos nuevos dependan únicamente de los ya colocados previamente.
- Mantener alineación entre objetos del mismo tipo.
- Favorecer ubicaciones en los bordes siempre que sea posible, para aprovechar mejor el espacio y mejorar la estética visual.
- Usar `around` y `face to` para ubicar sillas frente a escritorios o mesas.

5. **Instrucción final:** En la última parte del prompt, se incorpora la descripción original del usuario utilizado en el campo *description* mencionado en la sección 3.1. Se proporciona además una advertencia clara al modelo para que respete estrictamente el formato especificado, sin añadir texto introductorio ni conclusiones adicionales.

Una vez completado el prompt, este se envía a un modelo *LLM* de OpenAI mediante una solicitud API. Los parámetros utilizados para la generación son los siguientes:

- **Modelo:** gpt-3.5-turbo-instruct
- **Temperatura:** 0.7. Este valor ofrece un equilibrio entre coherencia y creatividad en las respuestas generadas.
- **Máximo de tokens:** 512. Suficiente para describir escenas con múltiples objetos y restricciones espaciales.
- **Penalización por frecuencia:** 0. Se permite la repetición de términos comunes, como nombres de objetos similares (por ejemplo, sofa-0, sofa-1).

Este enfoque produce una salida estructurada, precisa y fácilmente procesable, que luego se utilizará como base para la colocación espacial de los modelos 3D. A continuación, se presenta un ejemplo representativo:

Entrada del usuario:

A bed with two nightstands between.

Salida generada por el modelo:

```
bed-0 | edge
nightstand-0 | middle | right of, bed-0 | face to, bed-0
nightstand-1 | middle | left of, bed-0 | face to, bed-0
```

En este ejemplo, el modelo infiere que la cama debe colocarse cerca del borde de la habitación, mientras que las mesas de noche se ubican a los costados y orientadas hacia la cama.

A continuación se detalla el *prompt* completo:

```
1 You are an experienced room designer.
2 Please help me arrange objects in the room by assigning constraints
  to each object.
3 Here are the constraints and their definitions:
4 1. global constraint:
5     1) edge: at the edge of the room, close to the wall, most of
      the objects are placed here.
6     2) middle: not close to the edge of the room.
7
8 2. distance constraint:
9     1) near, object: near to the other object, but with some
      distance,  $50\text{cm} < \text{distance} < 150\text{cm}$ .
10    2) far, object: far away from the other object,  $\text{distance} \geq 150$ 
      cm.
11
12 3. position constraint:
13    1) in front of, object: in front of another object.
14    2) around, object: around another object, usually used for
      chairs.
15    3) side of, object: on the side (left or right) of another
      object.
16    4) left of, object: to the left of another object.
17    5) right of, object: to the right of another object.
18
19 4. alignment constraint:
20    1) center aligned, object: align the center of the object with
      the center of another object.
21
22 5. Rotation constraint:
23    1) face to, object: face to the center of another object.
24
25    For each object, you must have one global constraint and
      you can select various numbers of constraints and any
      combinations of them and the output format must be:
26 object | global constraint | constraint 1 | constraint 2 | ...
27 For example:
28 sofa-0 | edge
29 coffee table-0 | middle | near, sofa-0 | in front of, sofa-0 |
```

```
center aligned, sofa-0 | face to, sofa-0
30 tv stand-0 | edge | far, coffee table-0 | in front of, coffee table
   -0 | center aligned, coffee table-0 | face to, coffee table-0
31 desk-0 | edge | far, tv stand-0
32 chair-0 | middle | in front of, desk-0 | near, desk-0 | center
   aligned, desk-0 | face to, desk-0
33 float lamp-0 | middle | near, chair-0 | side of, chair-0
34
35     Here are some guidelines for you:
36 1. I will use your guideline to arrange the objects *iteratively*,
   so please start with an anchor object which doesn't depend on
   the other objects (with only one global constraint).
37 2. Place the larger objects first.
38 3. The latter objects could only depend on the former objects.
39 4. The objects of the *same type* are usually *aligned*.
40 5. I prefer objects to be placed at the edge (the most important
   constraint) of the room if possible which makes the room look
   more spacious.
41 6. When handling chairs, you should use the around position
   constraint. Chairs must be placed near to the table/desk and
   face to the table/desk.
42
43     Now I want you to design a bedroom and the room size is 5m x 5m
   .
44 Here are the room that I want you to make:
45 {description}
46 Please follow the desired format *strictly* (do not add any
   additional text at the beginning or end) to provide the
   constraints for each object.
```

3.2.1. Limpieza y verificación de la salida del LLM

Aunque los modelos de lenguaje de gran escala *LLM* pueden generar estructuras complejas a partir de texto libre, su salida no siempre es perfecta ni directamente utilizable. Por esta razón, es necesario aplicar un proceso de limpieza y verificación previo al uso de las restricciones en el algoritmo de colocación. Este proceso permite asegurar la validez sintáctica y semántica de las restricciones generadas, además de adaptarlas a las capacidades del siste-

ma.

A continuación, se describen los principales errores detectados en la salida del modelo, así como las estrategias implementadas para corregirlos:

1. **Formato incorrecto:** Con una temperatura alta (por ejemplo, 2), el modelo tiende a generar texto con formato incorrecto. Aunque una temperatura más baja, como 0.7, mejora la consistencia, no garantiza que los resultados sean siempre adecuados. A continuación, se presenta un ejemplo de este problema:

Entrada:

A bed with two nightstands between.

Salida generada:

Here is the bedroom with the desired format:

```
bed-0 | edge
nightstand-0 | middle | in front of, bed-0 | face to, bed-0
nightstand-1 | middle | in front of, bed-0 | face to, bed-0
```

El encabezado "*Here is the bedroom with the desired format:*" no cumple con el formato especificado y debe ser removido. Para evitar este tipo de errores, el sistema aplica una validación por expresiones regulares (*Regex*) que filtra líneas no conformes.

2. **Restricciones del mismo tipo:** A veces el modelo asigna múltiples restricciones del mismo tipo a un objeto, lo cual puede generar conflictos lógicos. Por ejemplo:

```
bed-0 | edge
nightstand-0 | middle | in front of, bed-0 | side of, bed-0
```

Aquí, las restricciones *in front of* y *side of* son incompatibles para una misma instancia, ya que implican ubicaciones espaciales distintas. En estos casos, se prioriza una la primera restricción.

3. **Restricciones redundantes:** También pueden surgir relaciones inválidas donde un objeto hace referencia a sí mismo:

```
nightstand-0 | middle | in front of, nightstand-0
```

Estas restricciones se eliminan automáticamente, ya que son lógicamente imposibles y no aportan información útil.

Una vez verificada la consistencia estructural de la salida, se realiza un filtrado adicional para eliminar cualquier objeto que no tenga una representación 3D disponible en el sistema. Por ejemplo:

Por ejemplo, si la estructura de la habitación es:

```
bed-0          | edge
nightstand-0   | middle | in front of, bed-0 | side of, bed-0
```

y el sistema no cuenta con un modelo 3D de un *nightstand*, entonces se remueve de la estructura toda referencia a ese objeto. Resultando en:

```
bed-0 | edge
```

Para llevar a cabo este proceso, se itera sobre cada objeto en la estructura y se compara con una lista de los nombres de los objetos 3D disponibles. Esta verificación se realiza mediante el cálculo de la similitud de coseno, lo que permite identificar nombres similares y garantizar que solo se mantengan aquellos objetos con representación válida en el sistema.

Una vez limpiada la estructura se da a paso a lo que es el algoritmo de colocación 3.3

3.3. Algoritmo de colocación de objetos

Una vez que se cuenta con una estructura espacial limpia y validada, el siguiente paso consiste en traducir las restricciones generadas por el modelo en posiciones físicas dentro del entorno tridimensional. Para ello, se implementa un algoritmo de colocación que utiliza una grilla que discretiza el suelo de la habitación.

Representación del espacio: La habitación se modela como una grilla bidimensional compuesta por puntos equiespaciados que cubren la superficie disponible. Cada punto representa una posible ubicación candidata para un objeto. La densidad de esta grilla (definida

por la distancia entre puntos) puede ajustarse en función del nivel de detalle requerido o del tamaño promedio de los objetos. A mayor densidad, más opciones de ubicación están disponibles, esto se puede visualizar en 8.

Es importante destacar que los objetos no siempre se colocan exactamente sobre los puntos de la grilla. En muchos casos, se aplican transformaciones menores (por ejemplo, un desplazamiento de pocos centímetros) para asegurar el cumplimiento estricto de restricciones espaciales como el contacto con una pared (*edge*).

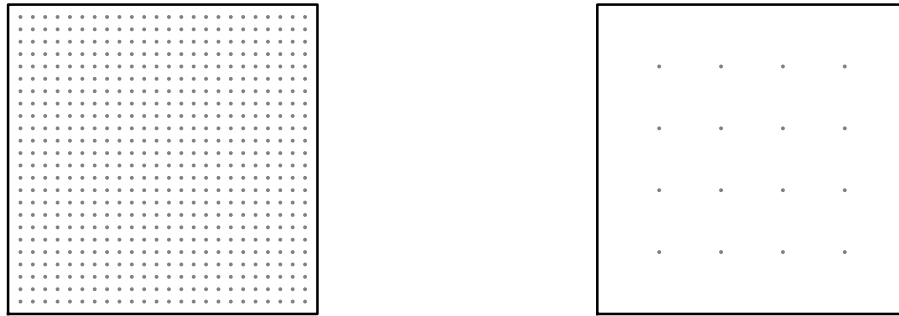


Figura 8: Dos habitaciones de 500x500 unidades: a la izquierda, una grilla con una distancia de 20 unidades entre puntos; a la derecha, una grilla con una distancia de 100 unidades.

Proceso de colocación: El algoritmo de colocación sigue una estrategia jerárquica. Primero se filtran los puntos candidatos según la restricción global del objeto (*edge* o *middle*). Posteriormente, se aplican de forma secuencial las restricciones locales: distancia, posición relativa, alineación y rotación. El proceso general del algoritmo presentado en 4 se describe a continuación:

1. Iterar sobre la lista de objetos, ordenada de forma jerárquica (primero el objeto ancla, seguido de los dependientes).
2. Para cada objeto:
 - a) Filtrar los puntos disponibles de acuerdo con su restricción global.
 - b) Aplicar los filtros sucesivos correspondientes a las restricciones locales (si existen).

- c) Si se encuentra una posición válida, colocar el objeto y marcar las celdas correspondientes como ocupadas.
- d) Si no se encuentra una ubicación válida, el objeto se omite del escenario final.

```
1 def colocar_objetos(objetos, restricciones, grilla):
2     escenario = {}
3
4     for obj in objetos:
5         posiciones = generar_candidatos(grilla)
6
7         # Filtrar según restricción global
8         posiciones = filtrar_por_restriccion_global(obj, posiciones
9             )
10
11        # Filtrar restricciones locales secuenciales
12        for restriccion in restricciones[obj]:
13            posiciones = filtrar_restriccion_local(restriccion,
14                posiciones, escenario)
15
16        # Seleccionar primera posición válida
17        if posiciones:
18            mejor_pos = posiciones[0]
19            escenario[obj] = mejor_pos
20            actualizar_grilla(grilla, mejor_pos)
21        else:
22            continue # No se coloca el objeto
23
24    return escenario
```

Listing 2: Pseudocódigo del algoritmo de colocación

3.4. Renderizado

Una vez completado el algoritmo de colocación, el escenario resultante —que contiene todos los objetos posicionados y orientados— se transmite al frontend mediante una solicitud HTTP. Esta comunicación se realiza a través de la API REST definida previamente, lo que da inicio al proceso de renderizado de la escena en el motor gráfico.

Esta visualización ocurre en el mismo programa que se utiliza para ingresar la descripción inicial. Una vez recibido el esquema, se procede a su interpretación y renderizado.

El esquema enviado consiste en una lista de objetos, donde cada objeto contiene los siguientes atributos:

```
1 {  
2   "nombre": "sofa-0",  
3   "modelo_3d": "res://modelos/sofa.tscn",  
4   "x": 2.5,  
5   "z": 4.0,  
6   "orientacion": 90  
7 }
```

Listing 3: Ejemplo de esquema

Cada objeto posee:

- **Nombre:** Identificador del objeto.
- **Modelo 3d:** Ruta hacia el modelo 3D correspondiente en Godot.
- **X y Z:** Coordenadas de posición sobre el plano de la habitación.
- **Orientación:** Rotación en grados alrededor del eje Y en grados.

El algoritmo de visualización recorre el esquema y, para cada objeto, instancia su modelo tridimensional y lo posiciona en la escena utilizando los atributos definidos. A continuación, se presenta un pseudocódigo que resume este proceso:

```
1 def visualizar_escena(esquema):  
2     for objeto in esquema:  
3         nodo = instanciar_modelo(objeto.modelo_3d)  
4         nodo.position.x = objeto.x  
5         nodo.position.z = objeto.z  
6         nodo.rotation.y = objeto.orientacion  
7         agregar_a_escena(nodo)
```

Listing 4: Pseudocódigo del algoritmo de visualización

- **Instanciación del modelo:** Se crea una instancia del modelo 3D especificado para el objeto, utilizando la ruta proporcionada en el esquema.
- **Asignación de posición:** Se asignan únicamente las coordenadas x y z porque, en un entorno de interiores, los objetos se distribuyen sobre un plano horizontal que representa el suelo. El eje y corresponde a la altura, y se asume inicialmente como cero por los alcances de este trabajo.
- **Rotación en Y:** Se aplica una rotación únicamente alrededor del eje Y, ya que este ajuste es suficiente para modificar la orientación horizontal de los objetos sin afectar su inclinación vertical, adecuada para elementos como muebles y decoraciones.

La función `instanciar_modelo` corresponde a una función nativa de Godot 4 que permite cargar dinámicamente modelos 3D como nodos, los cuales luego se integran al árbol de escenas del motor gráfico utilizando la función nativa `agregar_a_escena`.

3.4.1. Incorporación de modelos 3D

Cada objeto representado en la escena 3D está asociado a un modelo previamente almacenado en el frontend. Estos modelos fueron preparados manualmente para asegurar su compatibilidad espacial y visual dentro del entorno virtual. Las siguientes consideraciones fueron aplicadas:

- **Escalado:** Los modelos fueron ajustados tomando como referencia que una unidad en Godot equivale a un metro en el mundo real. Esta relación facilita el posicionamiento preciso y garantiza proporciones coherentes entre los distintos elementos.
- **Alineación:** Todos los modelos están colocados sobre el plano XY, y su orientación frontal apunta hacia el eje positivo del eje Z.
- **Centro geométrico:** El punto de origen de cada modelo coincide con su centro geométrico, lo que permite una colocación más precisa en el espacio.

Estas convenciones aseguran que los modelos puedan integrarse de forma uniforme en la escena, manteniendo la coherencia visual y espacial esperada a partir del esquema y la inferencia del LLM.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

Para evaluar la calidad del ordenamiento de los objetos en las escenas generadas, se diseñará un estudio de enfoque cualitativo en el que participarán entre 5 y 10 evaluadores. Este rango se alinea con las recomendaciones de Jakob Nielsen [22], quien sostiene que probar con cinco usuarios es suficiente para identificar la mayoría de los problemas de usabilidad, basándose en modelos matemáticos que respaldan esta afirmación. Esta decisión también considera las limitaciones de tiempo y recursos del presente trabajo.

El objetivo principal de la validación es determinar si la disposición de los objetos dentro de las habitaciones generadas es lógica, funcional y visualmente coherente, sin enfocarse en aspectos como la calidad visual o el realismo gráfico. Cada evaluador analizará únicamente una escena con el fin de mantener el tiempo de participación en torno a los 30 minutos, lo que permite evitar la fatiga cognitiva y asegurar una valoración más detallada. Esta decisión también responde a la disponibilidad limitada de los participantes y evita posibles sesgos derivados de la comparación entre múltiples escenas.

4.1. Procedimiento de evaluación

Para la evaluación del sistema se contactó a un total de 7 personas pertenecientes a carreras universitarias de Ingeniería y Arquitectura, dado su conocimiento técnico y visual, considerado relevante para los objetivos del proyecto. La recolección de participantes se realizó de forma mixta, combinando interacciones presenciales y remotas, según la disponibilidad de los evaluadores.

En la modalidad presencial, se abordó directamente a los participantes, explicándoles el objetivo del proyecto y presentándoles el programa en funcionamiento. Los evaluadores redactaron su propia descripción de una habitación (en inglés), ejecutaron la generación de la escena y observaron el resultado en tiempo real. Luego, completaron un formulario para evaluar la escena generada, y se registró una captura de pantalla junto con la descripción original para fines de documentación.

En la modalidad remota, el contacto se realizó mediante *Discord*. Se compartió pantalla para mostrar el funcionamiento del programa y, tras recibir la descripción escrita del evaluador,

se ingresó en el sistema. Posteriormente, el evaluador observó la generación de la escena en tiempo real y completó el formulario mediante un enlace. También se capturó una imagen de la escena correspondiente.

Las respuestas obtenidas fueron utilizadas como instrumento de validación del sistema. Se aplicó un cuestionario compuesto por preguntas cerradas (con escala ordinal de 1 a 4) y preguntas abiertas. Los datos cuantitativos fueron procesados estadísticamente para calcular promedios y detectar tendencias generales, mientras que los datos cualitativos fueron analizados para identificar observaciones relevantes sobre errores, aciertos y sugerencias de mejora.

De manera explícita, el proceso de evaluación constó de las siguientes etapas:

1. **Redacción de la descripción:** cada evaluador recibió una lista de objetos disponibles y redactó una breve descripción textual de una habitación en la que se incluyeran al menos cinco de dichos objetos. La redacción se realizó en inglés, dado que el sistema de generación opera con este idioma.
2. **Generación de la escena:** el sistema procesó la descripción como entrada, generó automáticamente una escena 3D y se capturó una imagen de la misma. Esta imagen sirvió como base para la evaluación posterior.
3. **Evaluación de la escena:** el evaluador analizó la imagen y completó un cuestionario con cuatro criterios de evaluación. Cada uno de ellos fue valorado mediante una escala ordinal, similar a una escala de Likert, con opciones categóricas que facilitaban la comprensión. También se incluyó un espacio para justificaciones abiertas.
4. **Recopilación de datos:** las respuestas fueron recolectadas de forma anónima y almacenadas para su posterior análisis, tanto cuantitativo como cualitativo.

4.2. Instrumentos de evaluación

Para evaluar la calidad de las escenas generadas, se aplicó un cuestionario estructurado compuesto por cuatro preguntas cerradas, cada una acompañada de una justificación abierta. Las preguntas abordan aspectos clave de la distribución espacial y la correspondencia semántica entre la descripción textual y la escena producida.

Cada pregunta cerrada ofrece cuatro opciones de respuesta ordenadas de mayor a menor grado de cumplimiento del criterio, permitiendo así una evaluación ordinal. Las justificaciones abiertas permiten captar matices y observaciones específicas por parte de los evaluadores.

Los criterios evaluados son:

1. **Correspondencia semántica:** ¿La escena coincide con la descripción proporcionada?
 - a) Sí, la escena representa perfectamente la descripción.
 - b) En general, coincide, pero hay algunos detalles distintos.
 - c) Hay varios elementos que no coinciden con la descripción.
 - d) La escena no representa en absoluto la descripción dada.
2. **Orientación de objetos:** ¿Los objetos están alineados correctamente? (Ejemplo: sillas frente a una mesa, camas orientadas hacia un área despejada).
 - a) Sí, todos los objetos están correctamente alineados.
 - b) En general bien alineados, pero hay algunos con orientación dudosa.
 - c) Varios objetos están orientados de forma incorrecta.
 - d) La mayoría de los objetos tienen una orientación incorrecta.
3. **Ausencia de colisiones:** ¿Los objetos están colocados sin colisiones ni intersecciones visualmente problemáticas?
 - a) No hay solapamientos, todos los objetos están bien posicionados.
 - b) Pequeños solapamientos que no afectan significativamente la disposición.
 - c) Múltiples solapamientos visibles, algunos afectan la comprensión de la escena.
 - d) Grave solapamiento de objetos, la disposición no es funcional.
4. **Ubicación funcional:** ¿Los objetos están posicionados de manera lógica según su uso esperado?
 - a) Sí, todos los objetos tienen una ubicación adecuada.
 - b) Mayormente bien ubicados, pero hay algunos elementos cuestionables.
 - c) Algunos objetos están en posiciones poco funcionales.
 - d) La ubicación de los objetos no es lógica ni práctica.

Luego, para capturar observaciones subjetivas que se escapan del alcance de las preguntas anteriores, se le pedirá al evaluador una justificación por cada pregunta.

4.3. Análisis de Resultados

Una vez obtenidas las respuestas de los evaluadores, se analizarán los datos de la siguiente manera:

- **Análisis cuantitativo:** Las respuestas a las preguntas cerradas serán codificadas numéricamente en una escala de 1 a 4, donde 4 representa el mayor grado de cumplimiento del criterio evaluado. Se calcularán estadísticas descriptivas como media, mediana y desviación estándar para cada ítem.
- **Análisis cualitativo:** Las justificaciones abiertas serán leídas y revisadas manualmente para identificar comentarios repetidos, observaciones relevantes y sugerencias concretas. A partir de esto, se agruparán las respuestas en temas comunes, como problemas de ubicación, errores en la interpretación de la descripción o recomendaciones de mejora. Este análisis permitirá complementar los datos numéricos con interpretaciones más detalladas sobre cómo los evaluadores percibieron las escenas generadas.
- **Discusión de resultados:** Luego se compararán los resultados numéricos con los comentarios escritos de los evaluadores, para ver en qué aspectos coinciden y en cuáles hay diferencias. Esta comparación ayudará a entender mejor cómo perciben los usuarios la calidad de las escenas generadas, no solo en números, sino también en sus propias palabras.

4.4. Resultados

A continuación se mostrarán los resultados obtenidos a partir de la evaluación realizada por los participantes, así como el análisis correspondiente de dichos datos. Esta sub-sección se divide en dos partes: primero, se presentan los resultados cuantitativos derivados de las preguntas cerradas del cuestionario, los cuales permiten obtener una visión general del desempeño del sistema en relación con los criterios definidos. Luego, se analiza la información cualitativa proveniente de las justificaciones abiertas entregadas por los evaluadores, la cual

ofrece una perspectiva más detallada sobre aspectos específicos que influyeron en sus respuestas.

Los resultados individuales, incluyendo las capturas de pantalla de las escenas evaluadas, las descripciones ingresadas por los usuarios y los datos recopilados, se encuentran disponibles en los Anexos (ver Subsección 5.2 y Subsección 5.3).

4.4.1. Análisis cuantitativo

El análisis cuantitativo de las evaluaciones realizadas por los participantes muestra resultados positivos en la mayoría de los criterios observados, tanto en términos de percepción como en consistencia estadística.

1) ¿La escena coincide con la descripción que proporcionaste?

7 respuestas



Figura 9: ¿La escena coincide con la descripción que proporcionaste?, Fuente: Elaboración propia

En cuanto a la **correspondencia semántica**, el 71,4 % de los participantes indicó que la escena “en general coincidía, pero con algunos detalles distintos”, mientras que un 14,3 % consideró que la escena representaba perfectamente la descripción, y otro 14,3 % señaló que “había varios elementos que no coincidían”. La media fue de 2,83, la mediana de 3, y la desviación estándar de 0,98, lo que refleja una mayor variabilidad en la percepción de este criterio.

2) ¿Los objetos están alineados correctamente? (Ejemplo: sillas frente a una mesa, camas orientadas hacia un área despejada).

7 respuestas

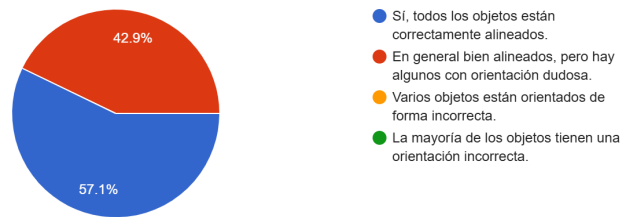


Figura 10: ¿Los objetos están alineados correctamente?, Fuente: Elaboración propia

En el criterio de **orientación de los objetos**, el 57,1 % de los evaluadores respondió que “todos los objetos estaban correctamente alineados”, y el 42,9 % indicó que estaban “en general bien alineados, pero con algunos con orientación dudosa”. Estos resultados se reflejan en una media de 3,67, mediana de 4 y desviación estándar de 0,47, lo que muestra un alto nivel de acuerdo sobre la correcta orientación espacial.

3) ¿Los objetos están colocados sin colisiones ni intersecciones visualmente problemáticas?

7 respuestas

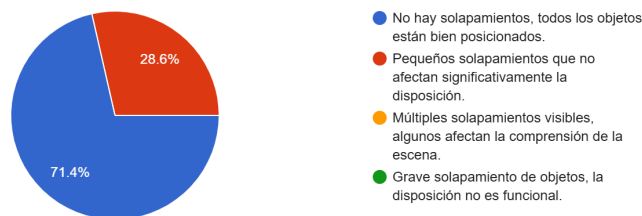


Figura 11: ¿Los objetos están colocados sin colisiones ni intersecciones visualmente problemáticas?, Fuente: Elaboración propia

Respecto a la **ausencia de colisiones**, el 71,4 % consideró que “no hay solapamientos, todos los objetos están bien posicionados”, mientras que el 28,6 % reportó “pequeños solapamientos que no afectan significativamente la disposición”. La media en este criterio también fue 3,67, la mediana de 4 y la desviación estándar de 0,47.

4) ¿Los objetos están posicionados de manera lógica según su uso esperado?

7 respuestas



Figura 12: ¿Los objetos están posicionados de manera lógica según su uso esperado?, Fuente: Elaboración propia

Finalmente, en la evaluación de la **ubicación funcional**, el 100 % de los evaluadores eligió la opción “mayormente bien ubicados, pero hay algunos elementos cuestionables”. Este consenso absoluto se refleja en una media de 3,00, mediana de 3 y desviación estándar igual a 0.

En conjunto, estos resultados respaldan que el sistema cumple adecuadamente con los aspectos básicos de disposición espacial, como orientación y posicionamiento sin colisiones, pero presenta oportunidades de mejora en la interpretación semántica de las descripciones y en la lógica funcional más precisa del mobiliario dentro del entorno simulado.

4.4.2. Análisis cualitativo

A partir de las justificaciones abiertas proporcionadas por los evaluadores, se identificaron varios patrones comunes que ayudan a interpretar con mayor profundidad las valoraciones numéricas:

1. **Interpretación general correcta, pero con errores puntuales:** Muchos evaluadores reconocen que la escena generada logra representar adecuadamente su descripción, pero mencionan discrepancias específicas, como la posición incorrecta de un mueble o la omisión de un detalle. Ejemplo: *“La imagen coincide con la descripción pero las camas son para 4 personas en vez de 2”*.
2. **Problemas con la orientación de objetos secundarios:** Se mencionaron dificultades en la orientación de elementos como sillas o veladores, que si bien estaban presen-

tes, no se ubicaban mirando hacia donde se esperaba. Esto refuerza que la **rotación y el sentido funcional** de algunos objetos no siempre es comprendido por el sistema. Ejemplo: *“El velador del lado derecho de la cama está mirando hacia un lado distinto”*.

3. **Solapamientos leves, pero perceptibles:** Aunque cuantitativamente el criterio de colisiones obtuvo una alta puntuación, varios comentarios hacen referencia a **pequeños solapamientos entre muebles**, sobre todo con veladores y camas. Ejemplo: *“Veladores solapados con camas”* o *“Entre el velador y la cama existe un pequeño solapamiento”*.
4. **Ambigüedad en el significado de “lógico” o “funcional”:** Algunos evaluadores interpretan la lógica funcional de formas distintas: algunos se refieren a distancias entre objetos, otros a su facilidad de uso. Esto sugiere que el criterio de ubicación funcional puede beneficiarse de una definición más precisa en futuras versiones del instrumento. Ejemplo: *“Los veladores no se alcanzan desde la cama... pero tiene sentido espacialmente”*.

En conjunto, estos hallazgos cualitativos respaldan y enriquecen los resultados cuantitativos. Si bien la orientación y el posicionamiento general son bien evaluados, aún hay margen de mejora en la precisión semántica y en el entendimiento funcional del contexto por parte del sistema.

4.5. Discusión de los resultados

Los resultados obtenidos en la evaluación permiten extraer conclusiones relevantes sobre el desempeño del sistema de generación de escenas a partir de descripciones textuales. En términos generales, los criterios de orientación y colisiones recibieron las valoraciones más altas por parte de los evaluadores, lo que indica una adecuada capacidad del sistema para alinear correctamente los objetos y evitar superposiciones evidentes. Esto sugiere que la lógica geométrica implementada para la colocación espacial cumple de manera satisfactoria su función.

Sin embargo, la valoración promedio más baja se observó en el criterio de correspondencia entre la escena generada y la descripción original. Esta discrepancia evidencia ciertas limitaciones del sistema en la interpretación semántica del texto, particularmente en descripciones que contienen relaciones espaciales complejas o referencias implícitas. Además, el análisis cualitativo refuerza esta observación, con varios evaluadores señalando diferencias

entre lo solicitado y lo representado, aunque en muchos casos se trató de detalles específicos más que errores estructurales.

Por otro lado, el criterio de ubicación funcional mostró una evaluación uniforme, lo cual puede ser interpretado de dos formas: por un lado, indica que la mayoría percibe una distribución razonable; por otro, la falta de variación podría reflejar que la consigna fue interpretada de forma ambigua o que los evaluadores no percibieron elementos particularmente destacables en este aspecto. Algunos comentarios señalaron, por ejemplo, que ciertos objetos resultaban inaccesibles o colocados en posiciones poco prácticas, aunque desde una perspectiva espacial eran coherentes.

Estos hallazgos permiten concluir que el sistema es competente en la disposición espacial básica, pero que requiere mejoras en su capacidad de comprender la lógica funcional y semántica de las descripciones. Se sugiere incorporar mecanismos de verificación post-generación o ajustes basados en la relación entre objetos. Asimismo, para futuras evaluaciones se recomienda redefinir ciertos criterios con mayor precisión y explorar métricas adicionales.

CAPÍTULO 5

CONCLUSIONES

5.1. Revisión de los objetos alcanzados

Los objetivos planteados al inicio de este trabajo fueron abordados satisfactoriamente a lo largo del desarrollo del sistema:

1. **Implementar un método para extraer modelos 3D basado en descripciones textuales de escenas:** Este objetivo se cumplió mediante el uso de modelos de lenguaje de gran escala y técnicas de *prompting*, lo cual permitió interpretar textos libres y convertirlos en esquemas espaciales estructurados. A partir de dichos esquemas, se identificaron los objetos mencionados y se enlazaron con sus correspondientes modelos 3D.
2. **Implementar un método para organizar los modelos 3D en una habitación:** Se implementó un algoritmo de colocación basado en una grilla discreta y búsqueda por profundidad, que permitió ubicar los objetos respetando restricciones espaciales. La integración con Godot Engine facilitó la visualización directa de los resultados generados.
3. **Evaluar la calidad de las escenas generadas mediante métricas de evaluación:** Se realizó una validación cualitativa con usuarios, aplicando criterios como orientación de objetos, ausencia de colisiones y coherencia espacial. Los resultados obtenidos permitieron medir la efectividad del sistema y detectar áreas de mejora tanto técnica como semántica.

5.2. Conclusión general

El presente trabajo ha abordado el desafío de generar automáticamente interiores tridimensionales a partir de descripciones en lenguaje natural, proponiendo e implementando una solución basada en técnicas modernas de procesamiento de lenguaje natural (NLP), aprendizaje automático y visualización en tiempo real mediante motores gráficos. A través de una arquitectura modular compuesta por un “*backend*” y un “*frontend*”, se logró desarrollar un sistema capaz de interpretar texto libre, extraer relaciones espaciales entre objetos y visualizar estos elementos en entornos virtuales utilizando Godot Engine. Este enfoque, además de

técnico, pone un énfasis significativo en la experiencia del usuario, apuntando a democratizar el diseño 3D al permitir que usuarios sin formación especializada puedan crear entornos complejos mediante instrucciones simples.

Desde el punto de vista técnico, uno de los principales logros de esta memoria fue demostrar con evidencia empírica que los modelos de lenguaje de gran escala pueden ser utilizados eficazmente para inferir información estructurada a partir de texto libre, siempre y cuando se diseñen estrategias de *prompting* adecuadas y se realice una validación de su salida. La implementación de *prompts* estructurados y el uso de restricciones espaciales permitieron generar esquemas lógicos que, si bien aún presentan ciertos errores en la interpretación o disposición de elementos, representan un avance hacia una interfaz de diseño 3D más accesible, automática y adaptable a distintos contextos de uso. Asimismo, se exploró la forma en que técnicas como el *role prompting* y *structured prompting* inciden en la precisión semántica de las salidas generadas, mostrando resultados prometedores en términos de coherencia sintáctica, estructuración espacial y reducción de ambigüedades típicas del lenguaje natural.

El algoritmo de colocación desarrollado, basado en una grilla discreta y en una búsqueda por profundidad, demostró ser efectivo para ubicar objetos en una habitación respetando tanto restricciones globales como locales. Su integración con un sistema de visualización permitió comprobar que es posible generar representaciones funcionales y lógicamente coherentes de interiores sin intervención manual directa, abriendo así posibilidades reales de democratizar el diseño espacial y reducir la dependencia de software profesional. Además, la limpieza automática de la salida del modelo y el control de colisiones resultaron fundamentales para mejorar la coherencia geométrica de las escenas generadas. Se implementaron mecanismos de validación con expresiones regulares y filtros de redundancia semántica para asegurar la consistencia del esquema espacial, lo cual redujo errores y conflictos lógicos que podrían dificultar la interpretación visual del entorno.

En cuanto a la validación de la solución, los resultados obtenidos en la evaluación cualitativa con usuarios mostraron un desempeño satisfactorio, particularmente en los criterios de orientación de objetos y ausencia de colisiones. Las puntuaciones promedio obtenidas sugieren una percepción positiva general del sistema por parte de los evaluadores, lo que confirma que el sistema, incluso en su etapa inicial, puede generar escenas que satisfacen expectativas básicas de coherencia y funcionalidad. No obstante, también se observaron discrepancias puntuales entre las descripciones textuales ingresadas por los usuarios y las escenas generadas, lo que resalta la necesidad de seguir perfeccionando el sistema. Estos hallazgos indican que, si bien la solución actual es robusta en cuanto a geometría y renderizado, aún debe mejorar en la precisión semántica, en la identificación de relaciones espaciales complejas y en

el tratamiento de ambigüedades inherentes al lenguaje humano, aspectos esenciales para una experiencia fluida.

Los comentarios cualitativos obtenidos ofrecieron una visión amplia sobre el funcionamiento del sistema. Se destacaron aciertos como la alineación espacial coherente, el cumplimiento de restricciones funcionales básicas y la ausencia de solapamientos graves, mientras que las principales críticas se centraron en detalles como la orientación incorrecta de objetos secundarios como sillas o veladores, la simetría inconsistente y la falta de precisión en ciertos posicionamientos, especialmente cuando las descripciones eran ambiguas o contenían múltiples interpretaciones posibles. También se notaron confusiones sobre qué tan funcional es una escena, es decir, si realmente tiene sentido para usarla en la vida real. Por eso, en versiones futuras del sistema, sería bueno incluir reglas más claras sobre cómo deben organizarse los objetos para que sean cómodos, fáciles de alcanzar y que dejen suficiente espacio para moverse. Además, ayudaría que el sistema revise automáticamente si hay errores en la distribución, usando criterios simples sobre cómo las personas entienden y se sienten en los espacios, para evitar problemas de claridad, comodidad o uso antes de mostrar la escena final.

El sistema que se desarrolló puede usarse en muchos tipos de situaciones. Por ejemplo, sirve tanto para programadores que quieren crear prototipos más rápido, como para personas sin experiencia técnica que quieren diseñar espacios interiores. También puede ser útil para profesores, estudiantes y diseñadores que necesiten una herramienta rápida para mostrar sus ideas. Además, como el sistema usa tecnologías abiertas como el motor gráfico Godot y servicios accesibles de inteligencia artificial, cualquier persona puede usarlo, modificarlo o mejorarlo sin restricciones. Esto es una gran ventaja frente a otros programas cerrados y costosos.

Sin embargo, el sistema también tiene algunas limitaciones importantes. Por ejemplo, solo puede usar los modelos 3D que ya tiene disponibles, y todavía no incluye funciones avanzadas como aplicar texturas automáticamente, usar luces realistas o permitir que los objetos se muevan e interactúen como en la vida real. Estas mejoras ayudarían a que las escenas se vean más reales y sirvan mejor para proyectos de arquitectura o simulaciones. Además, el sistema solo funciona en espacios planos y simples, sin pisos múltiples, estructuras más complejas o restricciones reales de construcción. Otra limitación es que necesita conexión a internet para usar el modelo de lenguaje de gran escala, lo cual puede ser un problema en lugares donde no hay buena conexión. Para solucionar esto, sería útil explorar formas de usar el sistema sin internet, como modelos que funcionen de forma local o técnicas que permitan guardar y reutilizar información para hacerlo más rápido y eficiente.

5.3. Aspectos técnicos que condicionaron el desarrollo

Durante el desarrollo del sistema se enfrentaron diversas limitaciones técnicas que influyeron directamente en las decisiones de diseño, herramientas utilizadas y alcance funcional de la solución. Si bien el objetivo general se logró, fue necesario realizar ajustes y replanteamientos en función de las restricciones impuestas por el entorno tecnológico y la velocidad con que evoluciona el estado del arte en esta área.

Una de las primeras decisiones técnicas fue la adopción de Flask como framework para construir el *backend*. Esta elección se basó en su simplicidad y facilidad de uso, lo que permitió implementar rápidamente una API REST funcional durante las primeras etapas del desarrollo. Sin embargo, Flask presentó ciertas limitaciones prácticas durante el proceso de construcción y prueba del sistema. En particular, los tiempos de inicio y reinicio del servidor resultaron elevados, lo que ralentizó significativamente las fases de depuración. Estas demoras afectaron directamente la productividad. Por ello, para una futura versión del sistema se recomienda evaluar otras opciones de desarrollo backend, como aquellas basadas en JavaScript (por ejemplo, Express o frameworks asíncronos como Next.js API routes), que ofrecen tiempos de arranque más rápidos.

Otro aspecto técnico crítico fue la búsqueda y selección de modelos 3D compatibles con el motor gráfico Godot. A lo largo del proceso se evidenció que muchos repositorios de modelos ofrecen archivos en formatos no directamente compatibles (como .max o .fbx), o bien contienen mallas demasiado pesadas para renderizado en tiempo real sin hardware especializado. Esta restricción obligó a filtrar y convertir modelos manualmente, priorizando aquellos en formato .glb/.gltf por su eficiencia y soporte nativo en Godot. Además, fue necesario equilibrar fidelidad visual y rendimiento, descartando modelos con alta carga poligonal en favor de versiones más ligeras pero funcionales.

También se identificó como condicionante el gran avance en el campo de los modelos de lenguaje de gran escala (LLMs). Durante el desarrollo del sistema, algunas versiones que inicialmente se consideraron estables, como GPT-3.5, comenzaron a quedar obsoletas frente a nuevos lanzamientos. Esto generó la necesidad de reevaluar decisiones previamente tomadas, ya que ciertas técnicas de *prompting* o flujos de trabajo dejaron de ser compatibles.

Finalmente, el crecimiento acelerado del estado del arte en generación automática de escenas 3D —tanto desde lenguaje natural como desde representaciones visuales— obligó a revisar de forma continua las metodologías adoptadas. Durante la ejecución del proyecto surgieron publicaciones relevantes que presentaban arquitecturas distintas o métricas más

especializadas, lo cual implicó un esfuerzo adicional por mantener la propuesta actualizada y técnicamente coherente dentro del marco temporal de una memoria de titulación.

Estos aspectos técnicos no solo representaron desafíos a superar, sino que también proporcionaron valiosas oportunidades de aprendizaje para futuros trabajos.

5.4. Trabajo a futuro

Frente a las limitaciones descritas anteriormente, se proponen varias líneas de trabajo futuro:

1. **Incorporación de feedback interactivo del usuario:** Dejar que el usuario pueda hacer cambios en la escena usando nuevas instrucciones escritas con lenguaje natural. Esto haría que el diseño se pueda ir mejorando paso a paso, y que el usuario participe activamente en la creación, como si fuera un coautor de la escena.
2. **Expansión de la base de modelos 3D:** Integrar una biblioteca más amplia y diversa de objetos tridimensionales, junto con mecanismos de búsqueda para su selección automática. Se podrían incorporar modelos generados por IA, o incluso permitir que los usuarios suban sus propios modelos.
3. **Mejoras en la comprensión semántica:** Probar combinaciones entre modelos de lenguaje y otros sistemas que entiendan mejor cómo se organizan los objetos en un espacio, como mapas de relaciones o redes que aprenden sobre ubicaciones. Esto ayudaría al sistema a entender mejor descripciones más complicadas y a evitar confusiones típicas del lenguaje humano.
4. **Evaluación en entornos reales de uso:** Realizar pruebas con arquitectos, diseñadores o desarrolladores de videojuegos para evaluar la usabilidad del sistema en contextos profesionales, midiendo tiempos de diseño, satisfacción del usuario y calidad del producto final.
5. **Integración de retroalimentación post-renderizado:** Agregar funciones que permitan al sistema aprender de los cambios que hacen los usuarios, para que pueda mejorar con el tiempo y cometer menos errores en el futuro.

En resumen, este trabajo muestra con ejemplos reales que sí es posible convertir descripciones escritas en escenas 3D útiles, utilizando modelos de lenguaje y motores gráficos. Aunque

todavía hay cosas que se pueden mejorar, como entender mejor el significado del texto o hacer escenas más realistas, el sistema desarrollado ya representa un gran paso hacia una herramienta de diseño más fácil de usar, rápida, personalizable y pensada para personas que no son expertas. Además, como está hecho de forma modular y con tecnologías abiertas, puede seguir creciendo y adaptándose a nuevos usos en el futuro.

ANEXOS

5.1. Anexo A – Formulario de Evaluación Usado

Se utilizó un formulario de Google para evaluar la percepción de los usuarios sobre las escenas generadas por el sistema. El cuestionario contenía preguntas cerradas y abiertas, agrupadas en los siguientes aspectos:

- Correspondencia semántica.
- Orientación de objetos.
- Ausencia de colisiones.
- Ubicación funcional de los objetos.
- Justificación abierta de cada respuesta.

En las tablas de continuación se muestra el formulario aplicado:

Pregunta	Tipo de respuesta	Opciones disponibles
¿La escena coincide con la descripción que proporcionaste?	Selección única (obligatoria)	Sí, la escena representa perfectamente la descripción. En general, coincide, pero hay algunos detalles distintos. Hay varios elementos que no coinciden con la descripción. La escena no representa en absoluto la descripción dada.
Justificación (1)	Respuesta abierta	Texto libre
¿Los objetos están alineados correctamente?	Selección única (obligatoria)	Sí, todos los objetos están correctamente alineados. En general bien alineados, pero hay algunos con orientación dudosa. Varios objetos están orientados de forma incorrecta. La mayoría de los objetos tienen una orientación incorrecta.
Justificación (2)	Respuesta abierta	Texto libre

Tabla 1: Formulario de evaluación (Parte 1).

Pregunta	Tipo de respuesta	Opciones disponibles
¿Los objetos están colocados sin colisiones ni intersecciones visualmente problemáticas?	Selección única (obligatoria)	No hay solapamientos, todos están bien posicionados. Pequeños solapamientos que no afectan significativamente la disposición. Múltiples solapamientos visibles, algunos afectan la comprensión de la escena. Grave solapamiento de objetos, la disposición no es funcional.
Justificación (3)	Respuesta abierta	Texto libre
¿Los objetos están posicionados de manera lógica según su uso esperado?	Selección única (obligatoria)	Sí, todos los objetos tienen una ubicación adecuada. Mayormente bien ubicados, pero hay algunos elementos cuestionables. Algunos objetos están en posiciones poco funcionales. La disposición no es lógica ni práctica.
Justificación (4)	Respuesta abierta	Texto libre
Descripción de la habitación	Respuesta abierta (obligatoria)	Texto libre proporcionado por el usuario
Foto de la habitación	Carga de archivo	Imagen subida por el usuario

Tabla 2: Formulario de evaluación (Parte 2).

5.2. Anexo B –Resultados Recopilados

Las respuestas fueron recopiladas de manera automática en una hoja de cálculo de *Google Sheets*. Los resultados incluyen tanto respuestas cuantitativas como cualitativas:

5.2.1. Evaluador 1

Descripción de la habitación	a bed with a lamp and a coffe table on the foot of the bed
1. ¿La escena coincide con la descripción?	En general, coincide, pero hay algunos detalles distintos.
Justificación 1	El posicionamiento de la cama y la lámpara estaban bien, puestas a esquinas opuestas, pero la mesa de café no estaba a los pies de la cama.
2. ¿Los objetos están alineados correctamente?	Sí, todos los objetos están correctamente alineados.
Justificación 2	Estaban bien puestas, la cama tenía el respaldo en la pared; la mesa no estaba bien posicionada, pero no mal alineada.
3. ¿Hay colisiones?	No hay solapamientos, todos los objetos están bien posicionados.
Justificación 3	La cama se ubicó en una esquina, dejando un pequeño espacio, pero no colisiona con la pared ni con otros muebles.
4. ¿Ubicación lógica?	Mayormente bien ubicados, pero hay algunos elementos cuestionables.
Justificación 4	La posición de la mesa de café no parece lógica, aunque cabe en el espacio dado.

5.2.2. Evaluador 2

Descripción de la habitación	In the foot of the bed there is a coffee table, and in the opposite corner from where the bed is positioned, there is a lamp.
1. ¿La escena coincide con la descripción?	En general, coincide, pero hay algunos detalles distintos.
Justificación 1	Me entiende cuando le pido que los objetos estén uno del lado del otro y también cuando le especifico que los quiero mirándose (por ejemplo, la silla hacia la mesa). Sin embargo, falla cuando le digo que quiero que algo suceda en el centro de la habitación.
2. ¿Los objetos están alineados correctamente?	Sí, todos los objetos están correctamente alineados.
Justificación 2	Hay que especificarlo con claridad, pero sí, están bien alineados.
3. ¿Hay colisiones?	No hay solapamientos, todos los objetos están bien posicionados.
Justificación 3	— *(Sin comentarios adicionales)* —
4. ¿Ubicación lógica?	Mayormente bien ubicados, pero hay algunos elementos cuestionables.
Justificación 4	Se basa en lo que uno le pide, por lo que si le escribes bien, logra algo lógico.

5.2.3. Evaluador 3

Descripción de la habitación	A bed with a nightstand beside it, a table across the room, and a gamer chair in front of the table.
1. ¿La escena coincide con la descripción?	La escena no representa en absoluto la descripción dada.
Justificación 1	Considero que en general entiende bien la descripción, por ejemplo, la orientación de la silla gamer respecto a la mesa. Pero el velador, aunque está al lado de la cama, no está en el lado correcto (cabecera). Lo positivo es que entendió la rotación respecto a la pared.
2. ¿Los objetos están alineados correctamente?	En general bien alineados, pero hay algunos con orientación dudosa.
Justificación 2	La mesa y la silla están bien ubicadas y orientadas. El velador está mal ubicado, aunque su rotación fue bien interpretada.
3. ¿Hay colisiones?	Pequeños solapamientos que no afectan significativamente la disposición.
Justificación 3	Hay un pequeño solapamiento entre el velador y la cama; el resto de los objetos no presenta solapamientos.
4. ¿Ubicación lógica?	Mayormente bien ubicados, pero hay algunos elementos cuestionables.
Justificación 4	El velador no tiene una ubicación lógica respecto a la cama.

5.2.4. Evaluador 4

Descripción de la habitación	A room with a double bed in the center, a drawer to the left of the bed, and a nightstand to the right.
1. ¿La escena coincide con la descripción?	Sí, la escena representa perfectamente la descripción.
Justificación 1	Las ubicaciones coinciden exactamente con las solicitadas, aunque la cama no está completamente centrada.
2. ¿Los objetos están alineados correctamente?	Sí, todos los objetos están correctamente alineados.
Justificación 2	Sería correcto decir que todos están bien alineados, con su orientación adecuada.
3. ¿Hay colisiones?	No hay solapamientos, todos los objetos están bien posicionados.
Justificación 3	Ningún objeto se superpone entre sí; todos están distribuidos respetando el espacio.
4. ¿Ubicación lógica?	Mayormente bien ubicados, pero hay algunos elementos cuestionables.
Justificación 4	Todos están de manera lógica y correcta a como uno esperaría encontrarlos.

5.2.5. Evaluador 5

Descripción de la habitación	A bedroom for twin boys
1. ¿La escena coincide con la descripción?	En general, coincide, pero hay algunos detalles distintos.
Justificación 1	Las camas son para cuatro personas y hay un mueble en vez de una mesa.
2. ¿Los objetos están alineados correctamente?	En general bien alineados, pero hay algunos con orientación dudosa.
Justificación 2	Algunos muebles están mal orientados, pero las camas están alineadas correctamente.
3. ¿Hay colisiones?	Múltiples solapamientos visibles, algunos afectan la comprensión de la escena.
Justificación 3	Las camas colisionan con la pared del fondo.
4. ¿Ubicación lógica?	Algunos objetos están en posiciones poco funcionales.
Justificación 4	Hay muebles bloqueando el paso entre camas.

5.2.6. Evaluador 6

Descripción de la habitación	I want a room with a double bed, a desk with a laptop on it, and a chair in front of the desk. Also, a nightstand on the left side of the bed and a dresser in the corner.
1. ¿La escena coincide con la descripción?	En general, coincide, pero hay algunos detalles distintos.
Justificación 1	Hay muchos más objetos de los solicitados, y las posiciones no siempre se condicen con lo que se pidió.
2. ¿Los objetos están alineados correctamente?	En general bien alineados, pero hay algunos con orientación dudosa.
Justificación 2	Algunos muebles están girados hacia lugares extraños, como el velador que mira hacia la pared.
3. ¿Hay colisiones?	Múltiples solapamientos visibles, algunos afectan la comprensión de la escena.
Justificación 3	La silla atraviesa parte del escritorio y la cama está parcialmente dentro de la pared.
4. ¿Ubicación lógica?	Algunos objetos están en posiciones poco funcionales.
Justificación 4	El escritorio está demasiado cerca de la cama y hay objetos ubicados en esquinas inaccesibles.

5.2.7. Evaluador 7

Descripción de la habitación	Create an interior scene using all possible space. Include a bed, desk, chair, nightstand, and a lamp in a functional layout.
1. ¿La escena coincide con la descripción?	En general, coincide, pero hay algunos detalles distintos.
Justificación 1	Algunos objetos solicitados no están presentes y otros parecen fuera de lugar, pero en general respeta el pedido.
2. ¿Los objetos están alineados correctamente?	En general bien alineados, pero hay algunos con orientación dudosa.
Justificación 2	La mayoría está alineada de forma funcional, pero el escritorio mira hacia la pared sin mucho sentido.
3. ¿Hay colisiones?	Pequeños solapamientos que no afectan significativamente la disposición.
Justificación 3	Hay un leve solapamiento entre la silla y el escritorio, pero nada crítico.
4. ¿Ubicación lógica?	Mayormente bien ubicados, pero hay algunos elementos cuestionables.
Justificación 4	La distribución parece funcional en general, aunque se podría aprovechar mejor el espacio libre.

5.3. Anexo C – Capturas de Escenas Evaluadas

En esta sección se incluyen algunas de las escenas generadas por el sistema, junto con las descripciones textuales ingresadas por los usuarios. Para la primera evaluación se perdió la captura de la escena debido a un problema técnico en la captura de datos, sin embargo, esto no fue impedimento para el análisis de las respuestas.

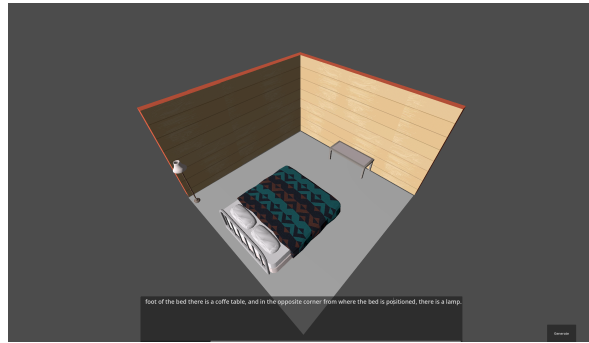


Figura 13: In the foot of the bed there is a coffee table, and in the opposite corner from where the bed is positioned, there is a lamp.

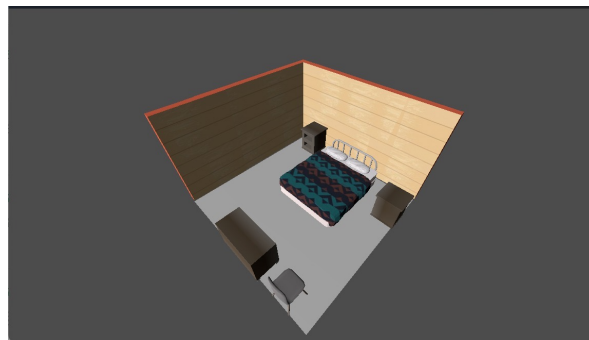


Figura 14: I want a room with a double bed, with a nightstand on the left side of the bed and another nightstand on the right. There should also be a black shelf on the wall opposite the bed, and below that shelf there should be a vanity with its own drawer and a desk chair.

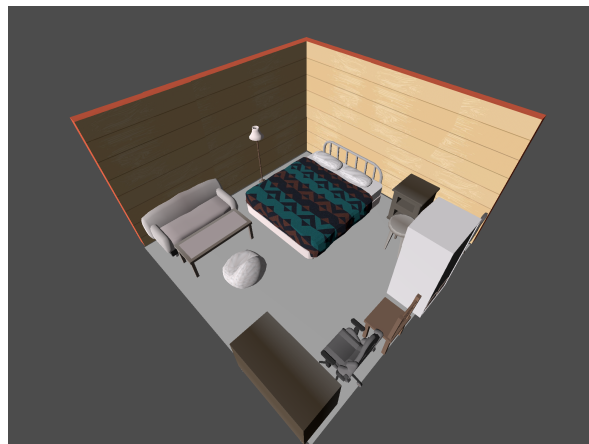


Figura 15: Create a interior scene using the following furniture items: a chair, coffee table, double bed, drawer, fridge, gamer chair, lamp, nightstand, puff, single bed, sofa, stool, table, and a wooden chair. Arrange them naturally in a cozy order

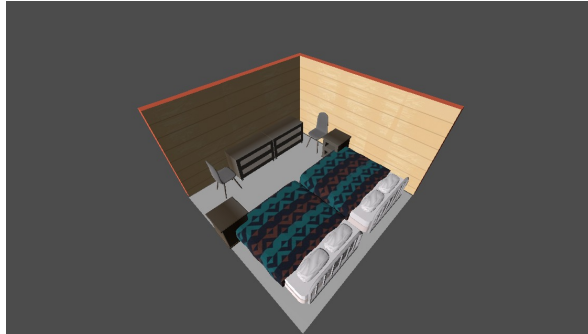


Figura 16: A bedroom for twin boys

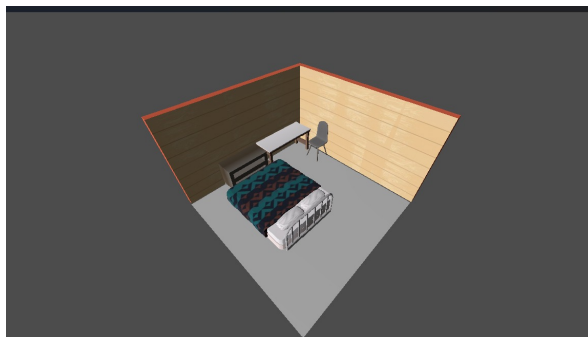


Figura 17: A room with a double bed in the center, a drawer on the wall in front of the bed, a table next to the drawer, and a chair placed by the table.

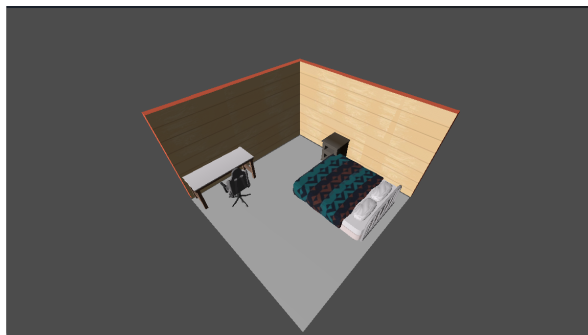


Figura 18: A bed with a nightstand beside it, a table across the room, and a gamer chair in front of the table.

REFERENCIAS BIBLIOGRÁFICAS

- [1] Mark Henne et al. "The making of Toy Story [computer animation]". En: mar. de 1996, págs. 463-468. isbn: 0-8186-7414-8. doi: 10.1109/CMPCON.1996.501812.
- [2] Electronic Arts. *The Sims Video Games - Official EA Site*. <https://www.ea.com/games/the-sims>. [Accedido: 21-Febrero-2025]. 2025.
- [3] Matt Deitke et al. *RoboTHOR: An Open Simulation-to-Real Embodied AI Platform*. 2020. arXiv: 2004.06799 [cs.CV]. url: <https://arxiv.org/abs/2004.06799>.
- [4] Joseph Weizenbaum. "ELIZA—a computer program for the study of natural language communication between man and machine". En: *Commun. ACM* 9.1 (ene. de 1966), págs. 36-45. issn: 0001-0782. doi: 10.1145/365153.365168. url: <https://doi.org/10.1145/365153.365168>.
- [5] Stephen C. Kleene. *Representation of Events in Nerve Nets and Finite Automata*. Princeton University Press, 1951.
- [6] Ken Thompson. "Regular Expression Search Algorithm". En: *Communications of the ACM* 11.6 (1968), págs. 419-422. doi: 10.1145/363347.363387.
- [7] Neha Baranwal, Avinash Singh y Suna Bensh. "Extracting Primary Objects and Spatial Relations from Sentences". En: : 2019. isbn: 978-989-758-350-6. doi: 10.5220/0007570002520259. url: <http://insticc.org/node/TechnicalProgram/icaart/presentationDetails/75700>.
- [8] Peter F. Brown et al. "Class-Based n -gram Models of Natural Language". En: *Computational Linguistics* 18.4 (1992), págs. 467-480. url: <https://aclanthology.org/J92-4003/>.
- [9] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". En: *arXiv preprint arXiv:1301.3781* (2013).
- [10] Ashish Vaswani et al. "Attention is All You Need". En: *Advances in Neural Information Processing Systems*. 2017, págs. 5998-6008.
- [11] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". En: *arXiv preprint arXiv:1810.04805* (2019).
- [12] Tom B Brown et al. "Language Models are Few-Shot Learners". En: *arXiv preprint arXiv:2005.14165* (2020).
- [13] Shervin Minaee et al. *Large Language Models: A Survey*. 2025. arXiv: 2402.06196 [cs.CL]. url: <https://arxiv.org/abs/2402.06196>.

- [14] ¿como se diseña un motor de videojuegos? — *campus.uoc.edu*. https://campus.uoc.edu/annotation/29a976745592d06c6df731cc1d4444f5/756548/PID_00246080/PID_00246080.html. [Accessed 04-02-2025].
- [15] Bob Coyne y Richard Sproat. “WordsEye: an automatic text-to-scene conversion system”. En: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, págs. 487-496. isbn: 158113374X. doi: 10.1145/383259.383316. url: <https://doi.org/10.1145/383259.383316>.
- [16] Richard Johansson, David Williams y Pierre Nugues. “Converting Texts of Road Accidents into 3D Scenes”. En: (jun. de 2004).
- [17] Funda Durupinar, Umut Kahramankaptan e Ilyas Cicekli. “Intelligent Indexing, Querying and Reconstruction of Crime Scene Photographs”. En: (ene. de 2004).
- [18] Angel X. Chang, Manolis Savva y Christopher D. Manning. “Learning Spatial Knowledge for Text to 3D Scene Generation”. En: *Conference on Empirical Methods in Natural Language Processing*. 2014. url: <https://api.semanticscholar.org/CorpusID:3029023>.
- [19] Weixi Feng et al. “LayoutGPT: Compositional Visual Planning and Generation with Large Language Models”. En: *ArXiv abs/2305.15393* (2023). url: <https://api.semanticscholar.org/CorpusID:258865950>.
- [20] Yixuan Yang et al. *LLplace: The 3D Indoor Scene Layout Generation and Editing via Large Language Model*. 2024. arXiv: 2406.03866 [cs.CV]. url: <https://arxiv.org/abs/2406.03866>.
- [21] Yue Yang et al. *Holodeck: Language Guided Generation of 3D Embodied AI Environments*. 2024. arXiv: 2312.09067 [cs.CV]. url: <https://arxiv.org/abs/2312.09067>.
- [22] Jakob Nielsen y Thomas K. Landauer. “A mathematical model of the finding of usability problems”. En: *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. CHI '93. Amsterdam, The Netherlands: Association for Computing Machinery, 1993, págs. 206-213. isbn: 0897915755. doi: 10.1145/169059.169166. url: <https://doi.org/10.1145/169059.169166>.