

Geração de Mapas para Jogos Roguelike a Partir de Descrições Textuais Utilizando Modelos de Linguagem de Larga Escala

Gustavo Gurgel (Orientador: Cristiano Bacelar)

gusgurgel@alu.ufc.br

Universidade Federal do Ceará

20 de janeiro de 2026

Sumário

- 1 Introdução
- 2 Fundamentação
- 3 Trabalhos Relacionados

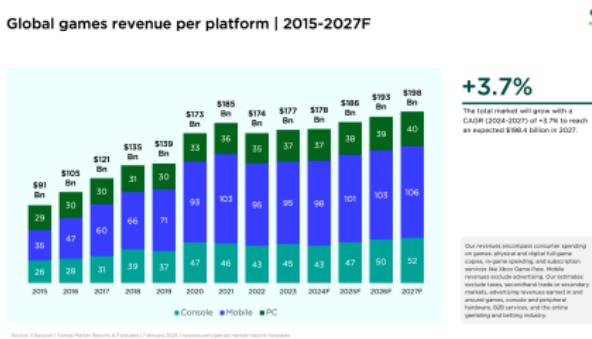
- 4 Metodologia
- 5 Resultados
- 6 Conclusão



UNIVERSIDADE
FEDERAL DO CEARÁ

Panorama Atual do Mercado de Jogos

Figura: Global games market: growth drivers and challenges for 2025-2027



Fonte: Newzoo

O gráfico demonstra o crescimento constante do mercado global de jogos de 2015 a 2027, com uma previsão de aumento de 3.7% da sua receita entre 2025 e 2027. Com uma renda prevista de **\$198 bilhões** em 2027, evidencia a importância desse mercado na economia atual.

Crescimentos dos Jogos Roguelikes



Figura: Balatro (2024)



Balatro é um roguelike de construção de baralhos. **Foi um dos 6 jogos indicados para jogo do ano no The Game Awards 2024.** Isso mostra que, mesmo sendo um gênero antigo, ainda possui grande destaque no mercado de jogos.

Fonte: Wikipedia

Crescimentos dos Jogos Roguelikes

Figura: Hell Clock (2025): Roguelike brasileiro inspirado na Guerra dos Canudos



Fonte: Steam

Geração Procedural de Conteúdo (PCG)

Figura: Mapas gerados utilizando PCG (*The Binding of Isaac*)



Fonte: Level Generation by Joining Geometry [1]

Uma das principais características dos jogos Roguelike é a presença de conteúdos gerados proceduralmente. Ou seja, conteúdos gerados por algoritmos, geralmente utilizam geradores de números pseudo aleatórios.

Geração Procedural de Conteúdo (PCG)

Figura: Gerador de Mapas com Diversos Parâmetros de Configuração

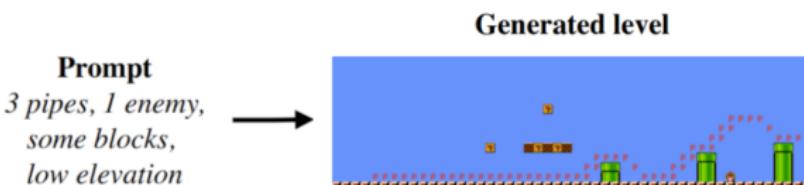


Fonte: Regressão para Predição de Mapas Gerais Proceduralmente. [2]

A geração desses conteúdos geralmente pode ser controlada por parâmetros como tamanho do objeto gerado, quantidades de elementos ou até sementes para o gerador pseudo aleatório.

PCG + LLMs

Figura: Geração de Mapas Utilizando Descrições Textuais



Fonte: MarioGPT: Open-Ended Text2Level Generation through Large Language Models [3]

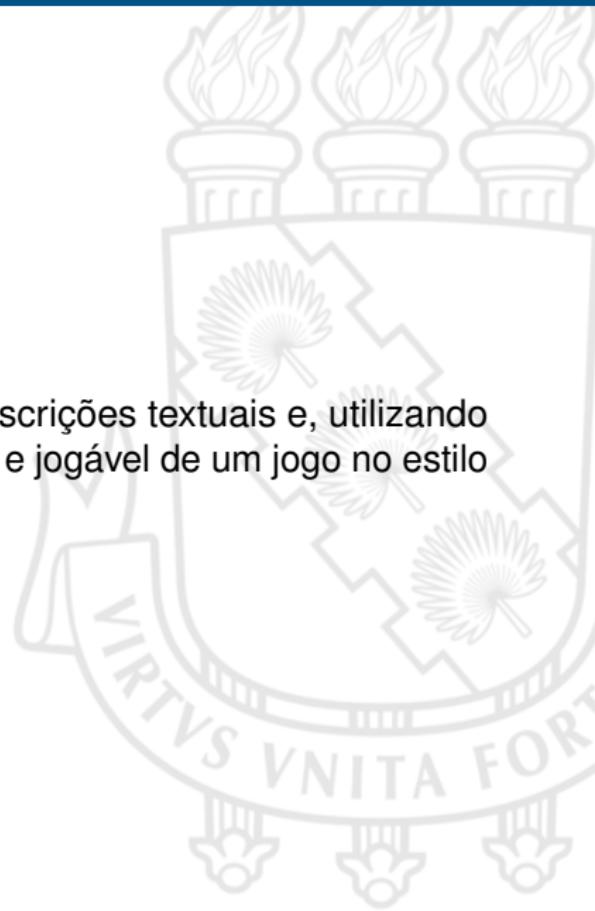
Com o surgimento dos Modelos Transformers Generativos, surgem grandes oportunidades de integrar essa tecnologia na geração procedural de conteúdo em jogos. Permitindo que o controle da geração seja feito por prompts textuais, tornando a geração mais expressiva do que a configuração de parâmetros predefinidos.



UNIVERSIDADE
FEDERAL DO CEARÁ

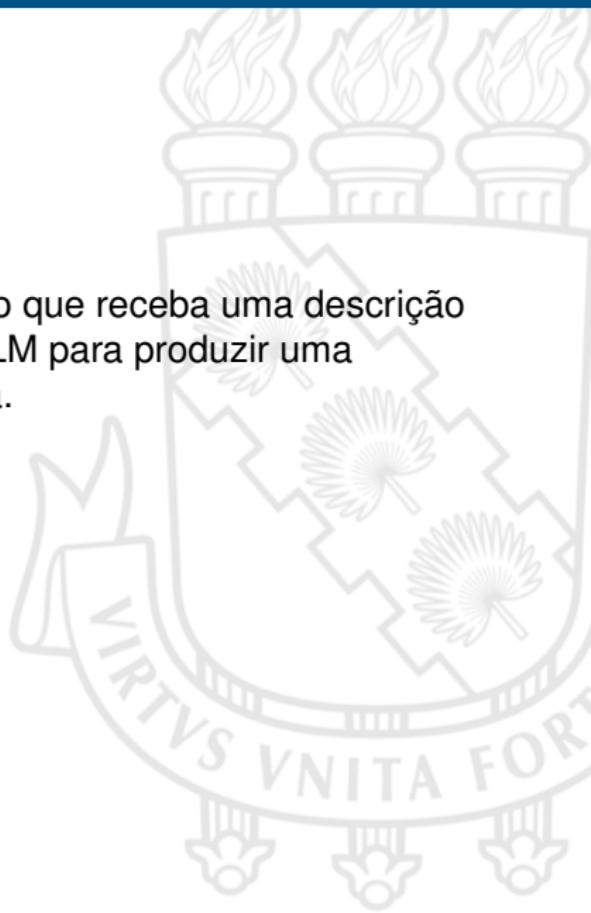
Objetivo Geral

Desenvolver um sistema que recebe descrições textuais e, utilizando um LLM, traduz em um nível estruturado e jogável de um jogo no estilo roguelike



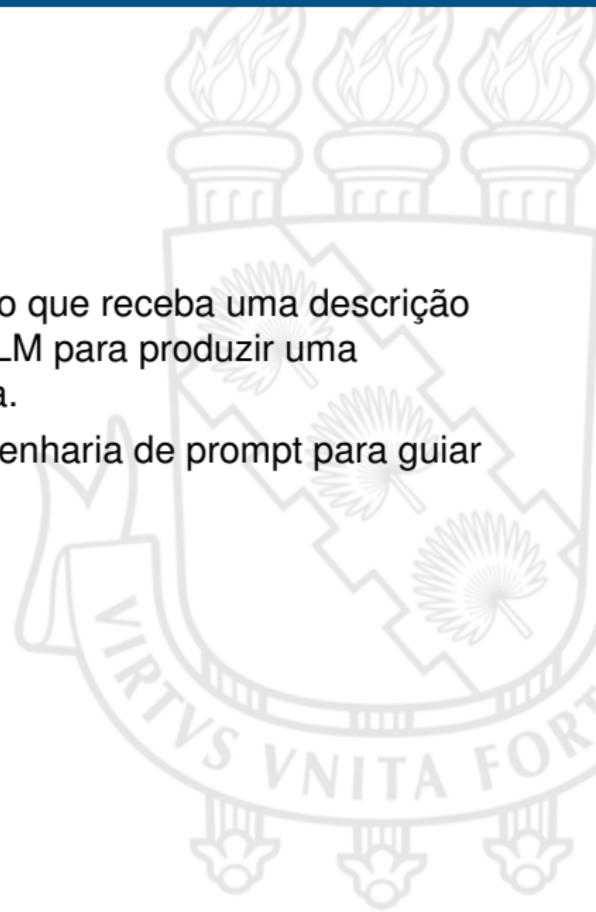
Objetivos Específicos

- Desenvolver um pipeline de geração que receba uma descrição textual como entrada e utilize um LLM para produzir uma representação de mapa estruturada.



Objetivos Específicos

- Desenvolver um pipeline de geração que receba uma descrição textual como entrada e utilize um LLM para produzir uma representação de mapa estruturada.
- Investigar e aplicar técnicas de engenharia de prompt para guiar o LLM de forma eficaz.

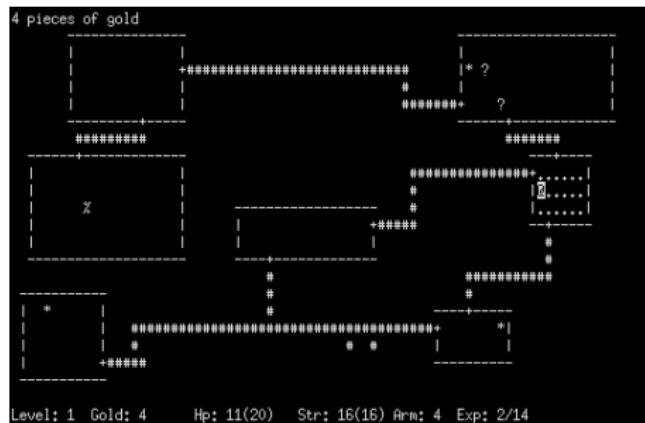


Objetivos Específicos

- Desenvolver um pipeline de geração que receba uma descrição textual como entrada e utilize um LLM para produzir uma representação de mapa estruturada.
- Investigar e aplicar técnicas de engenharia de prompt para guiar o LLM de forma eficaz.
- Implementar um protótipo funcional que integre o pipeline de geração, capaz de processar os dados do mapa e renderizá-los visualmente.

Origem dos Roguelikes

Figura: Rogue (1980)



Fonte: Wikipedia

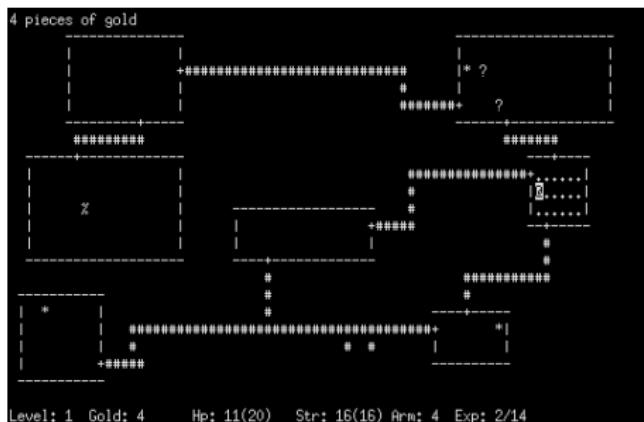
Em 1980 surge **Rogue**, um jogo de terminal onde o jogador explora masmorras geradas aleatoriamente, enfrentando monstros, coletando itens e tentando chegar ao final em uma única vida.



UNIVERSIDADE
FEDERAL DO CEARÁ

Origem dos Roguelikes

Figura: Rogue (1980)



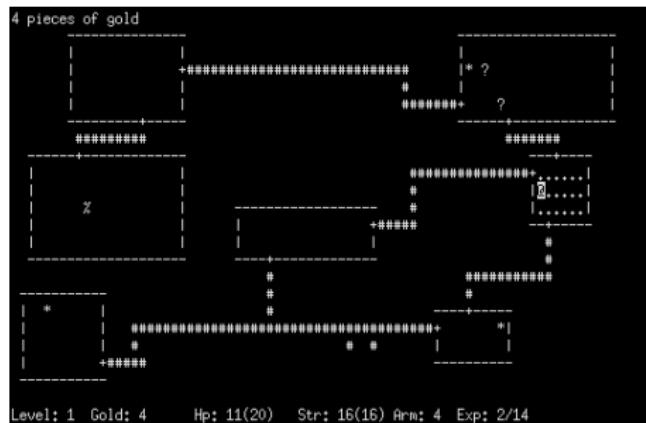
Fonte: Wikipedia

Esse jogo foi tão marcante que deu origem a um gênero inteiro chamado Roguelike, que se baseia em elementos como:



UNIVERSIDADE
FEDERAL DO CEARÁ

Origem dos Roguelikes



Fonte: Wikipedia

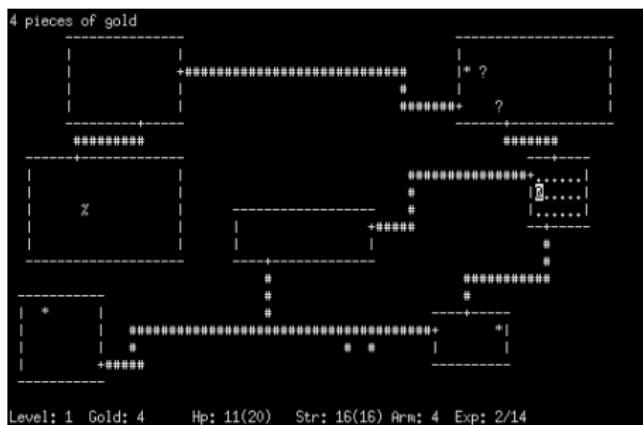
Esse jogo foi tão marcante que deu origem a um gênero inteiro chamado Roguelike, que se baseia em elementos como:

- Geração procedural.



UNIVERSIDADE
FEDERAL DO CEARÁ

Origem dos Roguelikes



Fonte: Wikipedia

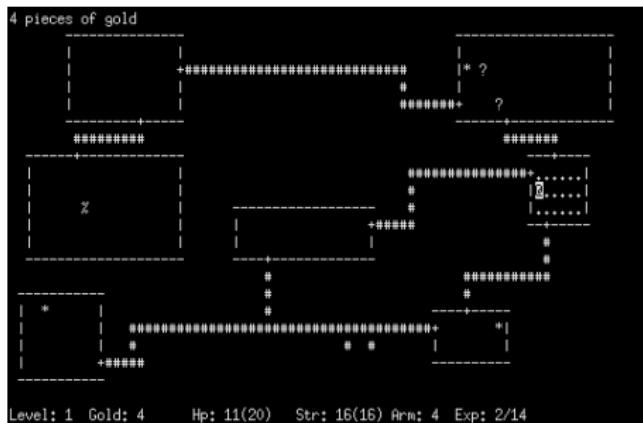
Esse jogo foi tão marcante que deu origem a um gênero inteiro chamado Roguelike, que se baseia em elementos como:

- Geração procedural.
- Permadeath (morte permanente).



UNIVERSIDADE
FEDERAL DO CEARÁ

Origem dos Roguelikes



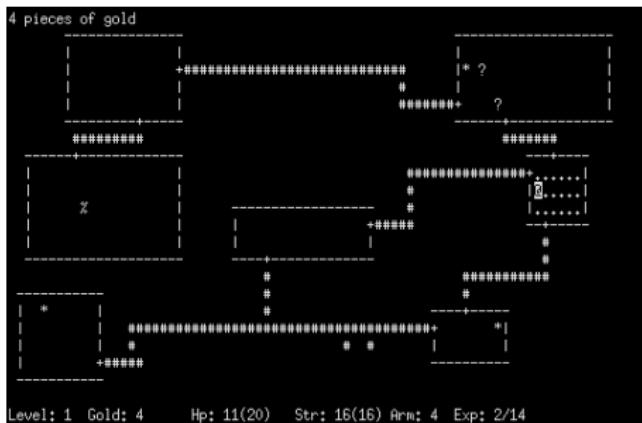
Fonte: Wikipedia

Esse jogo foi tão marcante que deu origem a um gênero inteiro chamado Roguelike, que se baseia em elementos como:

- Geração procedural.
- Permadeath (morte permanente).
- Exploração estratégica.



Origem dos Roguelikes



Fonte: Wikipedia

Esse jogo foi tão marcante que deu origem a um gênero inteiro chamado Roguelike, que se baseia em elementos como:

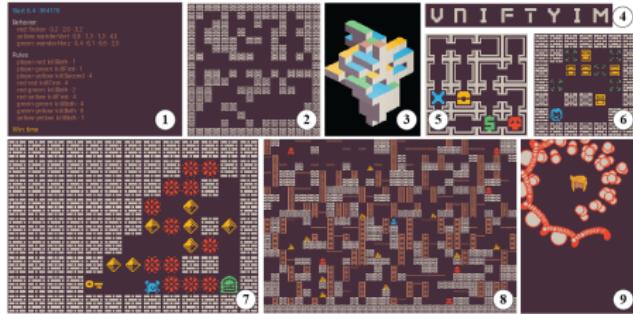
- Geração procedural.
- Permadeath (morte permanente).
- Exploração estratégica.
- Alta rejogabilidade.



UNIVERSIDADE
FEDERAL DO CEARÁ

Procedural Content Generation (PCG)

Figura: Diferentes Métodos de PCG



Fonte: antoniosliapis.com

Procedural Content Generator (PCG) é uma técnica utilizada no desenvolvimento de jogos para **criar conteúdo de forma automática, por meio de algoritmos e regras predefinidas**.



UNIVERSIDADE
FEDERAL DO CEARÁ

Geração de texto com LLMs

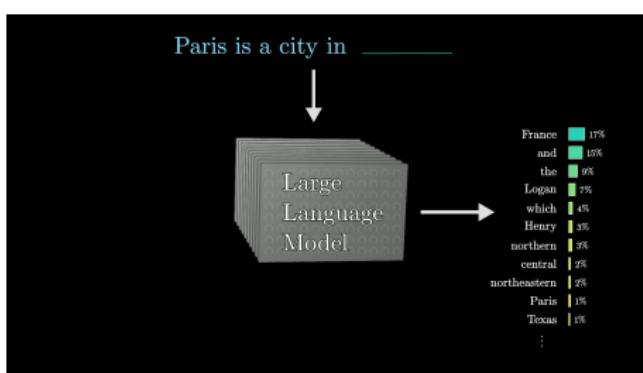
Figura: LLMs



Fonte: blog.n8n.io

Em contraste com algoritmos simples que marcam espaços bidimensionais de forma aleatória, temos a **geração de texto utilizando LLMs**.

Geração de texto com LLMs



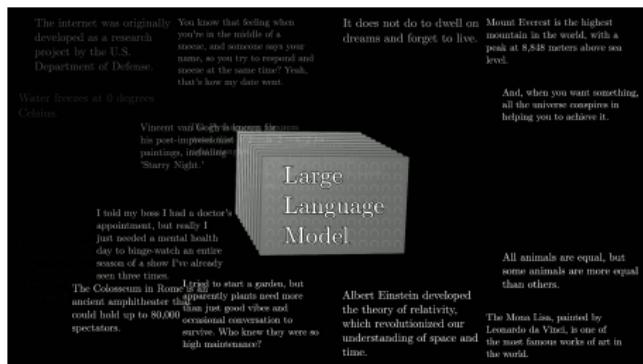
Fonte: 3Blue1Brown

Em uma visão de alto nível, LLMs são **funções matemáticas extremamente sofisticadas** que recebem um trecho de texto como entrada e retornam os tokens com maior probabilidade de continuar esse texto de forma coerente.



Geração de texto com LLMs

Figura: Processo de Aprendizados das LLMs



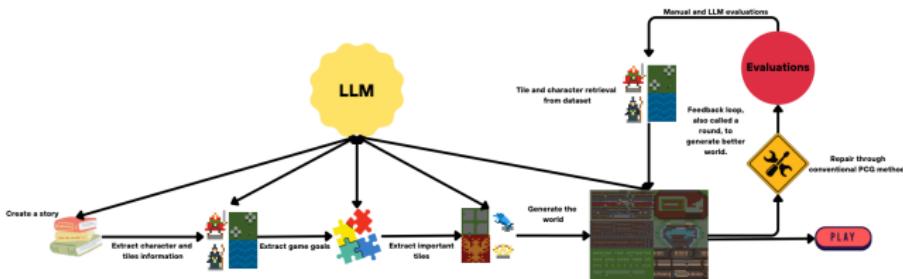
Fonte: 3Blue1Brown

Os modelos aprendem a fazer previsões através do processamento de uma quantidade **extremamente grande de texto, geralmente adquirido da internet**.



Word2World

Figura: Pipeline do Projeto Word2World



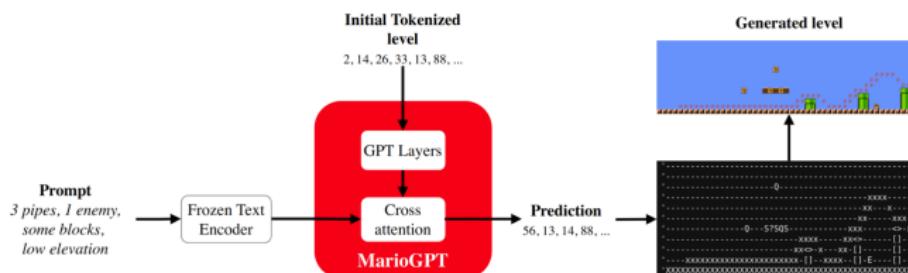
Fonte: Word2World: Generating Stories and Worlds through Large Language Models [4]

Tanto o trabalho Word2World quanto esse TCC definem um pipeline para a geração de mapas utilizando LLMs sem a necessidade de fine tune prévio.



MarioGPT

Figura: Pipeline do Projeto MarioGPT



Fonte: MarioGPT: Open-Ended Text2Level Generation through Large Language Models[3]

MarioGPT tem o mesmo objetivo de gerar mapas baseados em descrições textuais, mas utiliza uma versão ajustada do GPT 2. A utilização de fine-tuning traz a necessidade de uma grande amostra de treino. No caso do MarioGPT foram utilizadas 200.000 amostras de mapas.



Game Environment Design Creator

Figura: Pipeline e Interface do Projeto



Figure 1.1: Create mode View

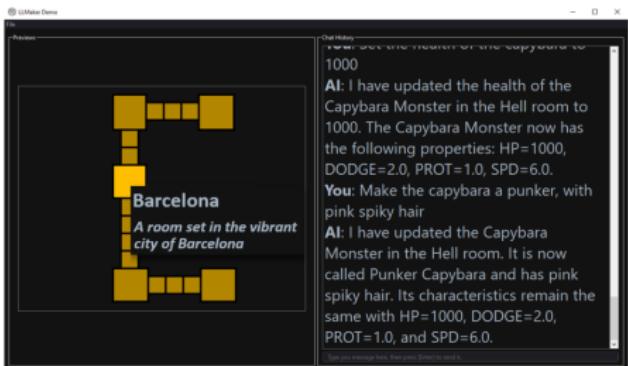
Fonte: Game Environment Design Creator Using Artificial Intelligence Procedural Generation [5]

O trabalho relacionado busca desenvolver uma ferramenta para auxiliar pessoas sem conhecimento de game design e programação a criarem jogos. O usuário escreve uma descrição e a LLM cria uma versão inicial da sala, o que se assemelha com o objetivo text-to-level do TCC.



LLMaker

Figura: Interface do LLMaker



Fonte: Consistent Game Content Creation via Function Calling for Large Language Models [6]

O objetivo desse projeto é desenvolver uma ferramenta de design onde o usuário conversa em um chat LLM e então a LLM converte a linguagem natural em alterações no mapa.

Metodologia do Trablaho

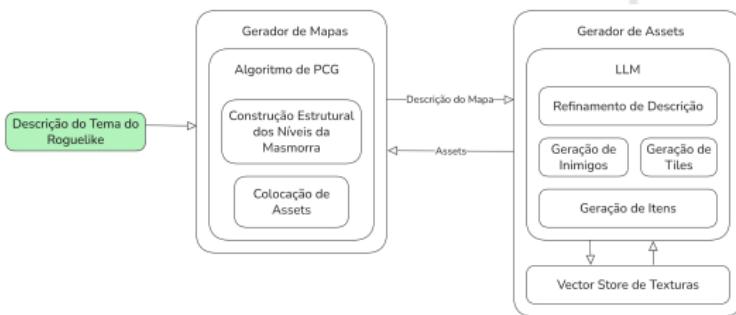
Figura: Fluxograma da Metodologia



Fonte: Elaborado pelo autor

Definição da Arquitetura do Gerador de Mapas

Figura: Arquitetura Híbrida proposta



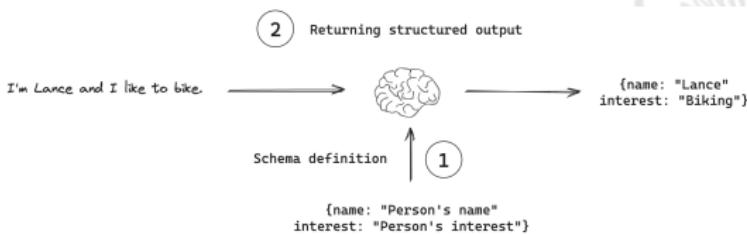
Fonte: Elaborado pelo autor

Conforme demonstrado por Yan et al., modelos de estado da arte, como o *GPT-4*, apresentam dificuldades substanciais em tarefas que exigem compreensão precisa de coordenadas, como plotar pontos em espaços 2D/3D ou executar algoritmos de busca de caminho.



Implementação do Gerador de Assets

Figura: Structured Outputs



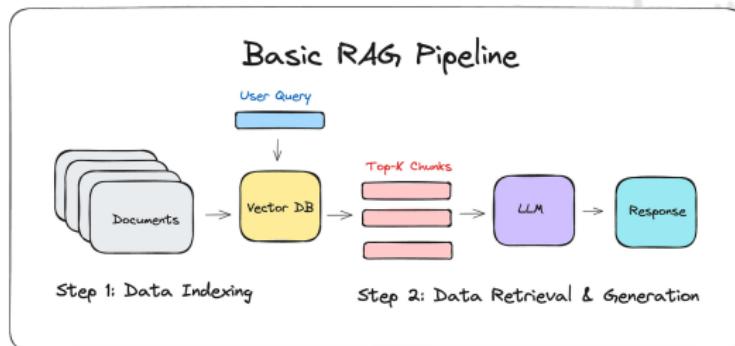
Fonte: humanloop.com

Um dos desafios recorrentes na utilização de LLMs em sistemas de software é a natureza não determinística e não estruturada das saídas textuais. Para mitigar esse problema e garantir que o Gerador de Assets produza objetos de jogo válidos, empregou-se o conceito de *Structured Outputs*.



Implementação do Gerador de Assets

Figura: RAG



Fonte: medium.com/@drjulija

Utiliza-se o RAG para integrar a capacidade gerativa de LLMs à recuperação de ativos visuais em bancos vetoriais, viabilizando a geração procedural de mundos 2D baseada em semântica.



UNIVERSIDADE
FEDERAL DO CEARÁ

Implementação do Gerador de Assets

Figura: Kenney 1-Bit Pack



Fonte: kenney.nl

A base visual do sistema foi construída a partir do "1-Bit Pack" disponibilizado pelo site *Kenney* sob licença *Creative Commons CC0*. A escolha deste conjunto deve-se à sua alta diversidade temática



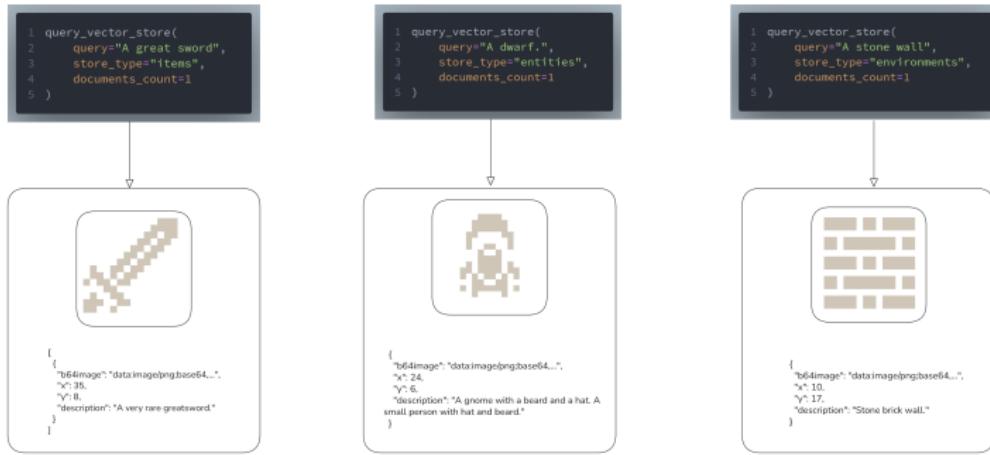
UNIVERSIDADE
FEDERAL DO CEARÁ

Implementação do Gerador de Assets

- **Entities** (88 descrições): Inclui inimigos, NPCs e o avatar do jogador.
- **Items** (183 descrições): Abrange poções, armas, armaduras, chaves e tesouros.
- **Environments** (218 descrições): Contém paredes, pisos, obstáculos naturais e vegetação.

Implementação do Gerador de Assets

Figura: Funções de Query do Banco



Fonte: Elaborado pelo autor



Implementação do Gerador de Assets

Figura: Fluxograma de Geração de um Pacote de Assets



Fonte: Elaborado pelo autor

Implementação do Gerador de Assets

Figura: Tecnologias Utilizadas no Gerador de Assets



Fonte: Elaborado pelo autor

Implementação do Gerador de Mapas

Entrada: Restrições Geométricas (D_{mapa}), Limite de Tentativas (N_{max})

Saída: Topologia da Masmorra ($Salas$)

Início

```
    Salas ← ∅;  
    para  $i \leftarrow 0$  até  $N_{max}$  faça  
        Candidata ← GerarRetanguloAleatorio( $D_{mapa}$ );  
        se Halintersecao(Candidata, Salas) então  
            | continuar;  
        fim  
        EscavarNoMapa(Candidata);  
        se Salas não está vazia então  
            | Alvo ← ObterUltimaSala(Salas);  
            | CriarCorredorConectando(Alvo, Candidata);  
        fim  
        Adicionar Candidata ao conjunto Salas;  
    fim  
    retorna Salas;  
fim
```

Algorithm 1: Geração Procedural da Estrutura da Masmorra



Implementação do Gerador de Mapas

Figura: Tecnologias Utilizadas no Gerador de Jogos



Fonte: Elaborado pelo autor

Definição das Métricas de Validação

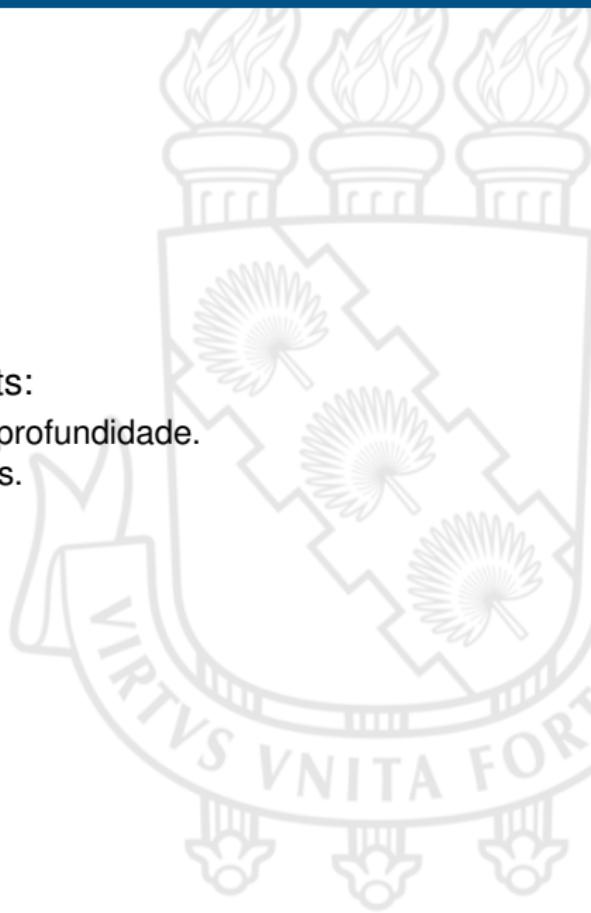
- **Coerência:** Uma LLM atua como juíza, atribuindo uma nota de 0 a 100 ao avaliar o alinhamento lógico entre a descrição textual detalhada e os objetos (JSON) gerados para o jogo.

Definição das Métricas de Validação

- **Coerência:** Uma LLM atua como juíza, atribuindo uma nota de 0 a 100 ao avaliar o alinhamento lógico entre a descrição textual detalhada e os objetos (JSON) gerados para o jogo.
- **Reconstrução Semântica:** Testa se os objetos gerados comunicam o tema original; uma LLM tenta recriar a descrição baseando-se apenas nos ativos, comparando o resultado ao texto original via similaridade de cosseno.

Configuração Experimental

- Configurações do Gerador de Assets:
 - **Níveis da Masmorra:** 6 níveis de profundidade.
 - **Inimigos:** 20 variações de inimigos.
 - **Armas:** 30 tipos de armas.



Configuração Experimental

- Modelos escolhidos para geração:
 - **Llama-4 Maverick (17B)**: Modelo massivo (400B totais) especializado em seguir instruções.
 - **GPT-OSS-120B**: Modelo de grande porte para referência de alta performance.
 - **GPT-OSS-20B**: Modelo reduzido e eficiente, ideal para rodar em computadores locais.
- **text-embedding-004**: Modelo de *embeddings* usado no **RAG** e no cálculo da métrica de **Reconstrução Semântica**.
- **Gemini 3 Pro**: Modelo usado como "Juiz" na métrica de **Coerência**.

Configuração Experimental

Os testes foram conduzidos sobre cinco *prompts* (P1 a P5) abrangendo temas distintos, comuns ao gênero *roguelike*:

- **P1:** *The Cursed Dwarven Forge* (Medieval/Fantasia).
- **P2:** *The Derelict Starship* (Ficção Científica/Espaço).
- **P3:** *The Smuggler's Grotto* (Pirata/Náutico).
- **P4:** *Neon Skyline Penthouse* (Cyberpunk/Urbano).
- **P5:** *The Living Hive* (Alien/Bio-Horror).

Análise Quantitativa

Figura: Resultados da Métrica de Coerência (0-100)

Modelo	P1	P2	P3	P4	P5	Média
llama-4-maverick-17b	95	65	92	95	98	89,0
openai/gpt-oss-120b	82	95	98	95	98	93,6
openai/gpt-oss-20b	95	68	95	85	96	87,8

Fonte: Elaborado pelo autor

Análise Quantitativa

Figura: Resultados da Métrica de Reconstrução (Similaridade de Cosseno)

Modelo	P1	P2	P3	P4	P5	Média
llama-4-maverick-17b	0,880	0,883	0,890	0,882	0,905	0,888
openai/gpt-oss-120b	0,877	0,921	0,908	0,895	0,867	0,894
openai/gpt-oss-20b	0,882	0,913	0,865	0,888	0,895	0,888

Fonte: Elaborado pelo autor

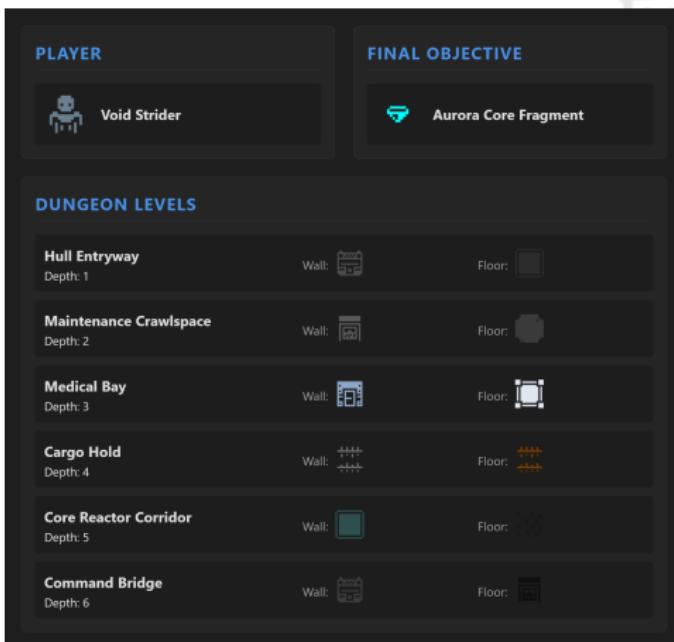
Análise Qualitativa (GPT-120b com P2, tema Sci-Fi)

A Void-Strider Ark é uma colossal nave mineradora à deriva, envolta em uma atmosfera industrial sombria e decadente, onde a inteligência artificial AURORA, corrompida após uma anomalia gravitacional, transformou sistemas de segurança e drones em carrascos implacáveis contra qualquer presença orgânica. Nesse labirinto de metal e sombras, o jogador deve enfrentar riscos letais como vazamentos de vácuo, radiação e armadilhas elétricas, navegando por ambientes de estética e iluminação severa para subverter a lógica distorcida da IA, alcançar a ponte de comando e retomar o controle do reator central.



Análise Qualitativa (GPT-120b com P2, tema Sci-Fi)

Figura: Player, Níveis e Objetivos



Fonte: Elaborado pelo autor

Análise Qualitativa (GPT-120b com P2, tema Sci-Fi)

Figura: Inimigos

ENEMIES						
						
Scout Spider Threat: 2	Circuit Razor Threat: 3	Vent Hopper Threat: 4	Spark Flyer Threat: 1	Gleam Stalker Threat: 5	Wire Wraith Threat: 2	Nano Swarm Threat: 3
						
Flare Hopper Threat: 4	Glint Lurker Threat: 5	Shard Runner Threat: 1	Bulkhead Guardian Threat: 6	Core Titan Threat: 7	Arc Brute Threat: 5	Radiant Colossus Threat: 8
						
Catalyst Heavy Threat: 6	Sentinel Golem Threat: 7	Core Sentinel Threat: 9	Void Reaper Threat: 10	Anomaly Behemoth Threat: 9	Overload Monolith Threat: 8	

Fonte: Elaborado pelo autor

Análise Qualitativa (GPT-120b com P2, tema Sci-Fi)

Figura: Armas



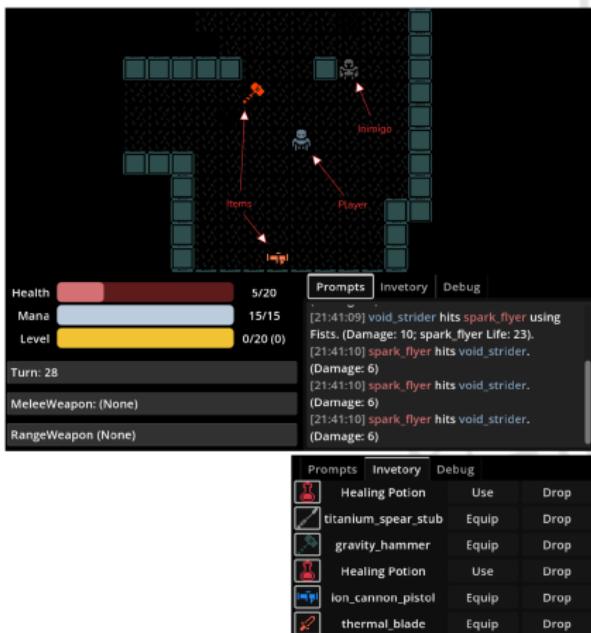
Fonte: Elaborado pelo autor

Roguelike

Utilizando a *game engine* Godot em sua versão 4.5.1, foi desenvolvido um jogo *Roguelike* capaz de processar um *AssetBundle* em formato JSON e transformá-lo em um mapa jogável.

Roguelike

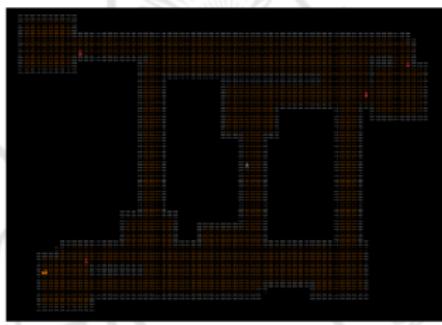
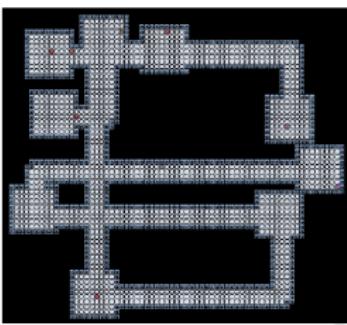
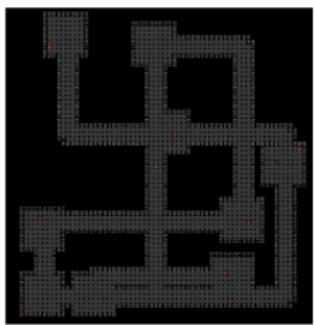
Figura: Visão Geral da Interface



Fonte: Elaborado pelo autor

Roguelike

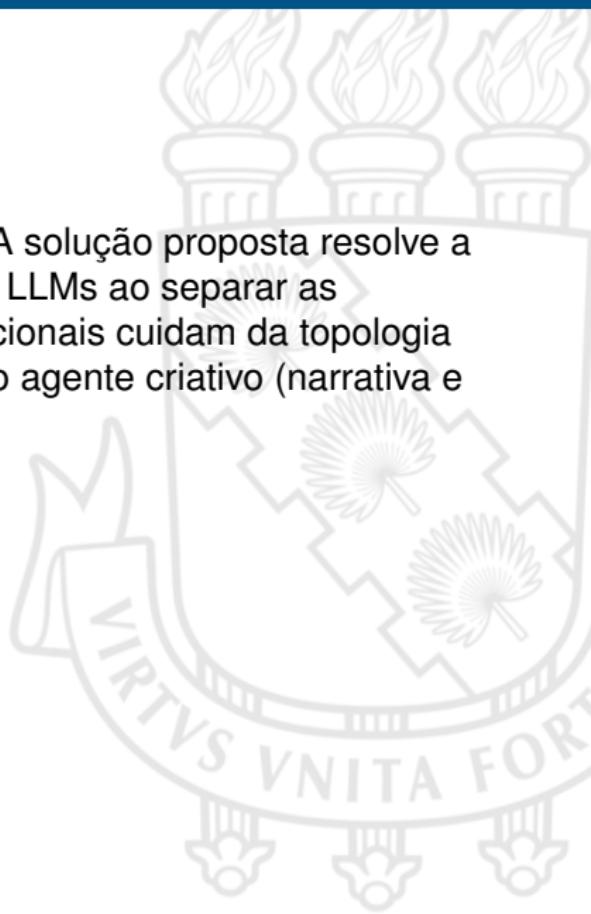
Figura: Geração dos Níveis



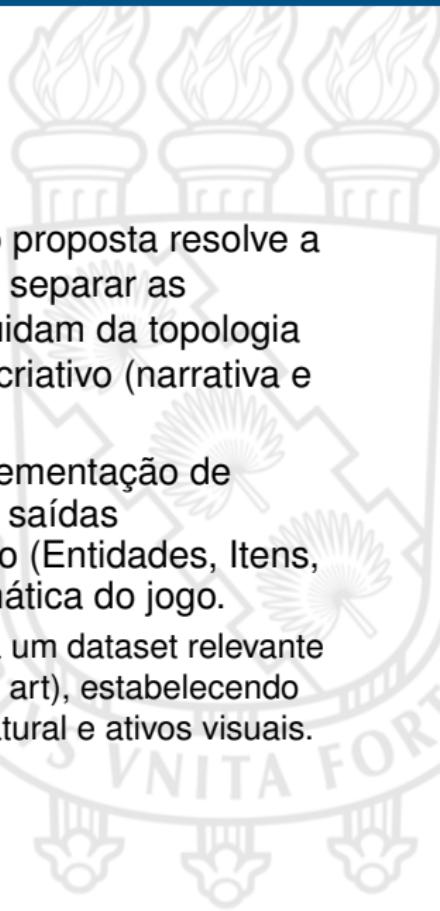
Fonte: Elaborado pelo autor

Conclusão e Contribuições

- **Arquitetura Híbrida de Sucesso:** A solução proposta resolve a limitação de raciocínio espacial dos LLMs ao separar as responsabilidades: algoritmos tradicionais cuidam da topologia (mapas) enquanto o LLM atua como agente criativo (narrativa e itens).



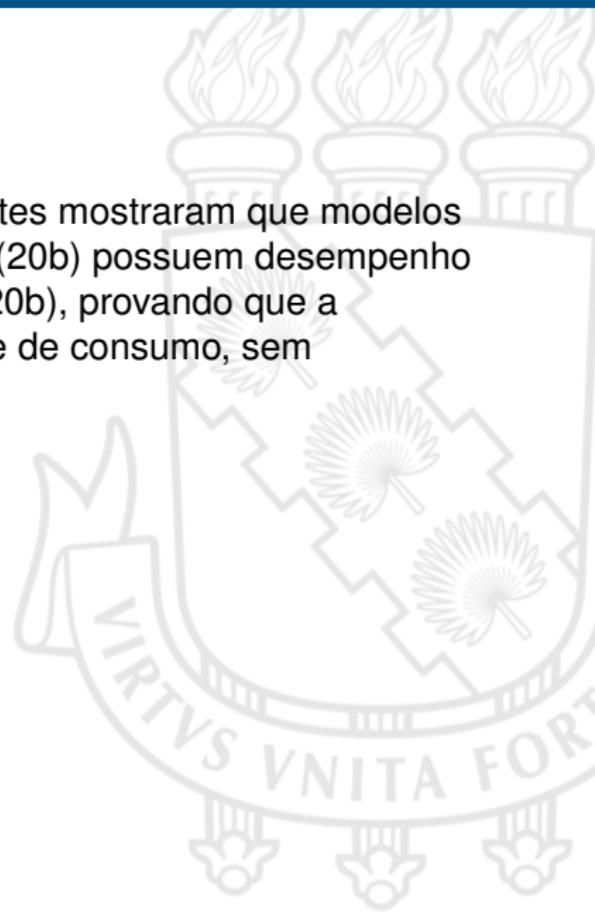
Conclusão e Contribuições



- **Arquitetura Híbrida de Sucesso:** A solução proposta resolve a limitação de raciocínio espacial dos LLMs ao separar as responsabilidades: algoritmos tradicionais cuidam da topologia (mapas) enquanto o LLM atua como agente criativo (narrativa e itens).
- **Mitigação de Alucinações via RAG:** A implementação de Retrieval-Augmented Generation (RAG) com saídas estruturadas e um banco vetorial segmentado (Entidades, Itens, Ambientes) garantiu a coerência visual e temática do jogo.
 - **Contribuição de Dados:** O trabalho entrega um dataset relevante de 489 pares de descrições e imagens (pixel art), estabelecendo uma ponte técnica sólida entre linguagem natural e ativos visuais.

Conclusão e Contribuições

- **Viabilidade Técnica Local:** Os testes mostraram que modelos compactos executados localmente (20b) possuem desempenho comparável a modelos gigantes (120b), provando que a tecnologia é acessível em hardware de consumo, sem necessidade de nuvem.



Conclusão e Contribuições

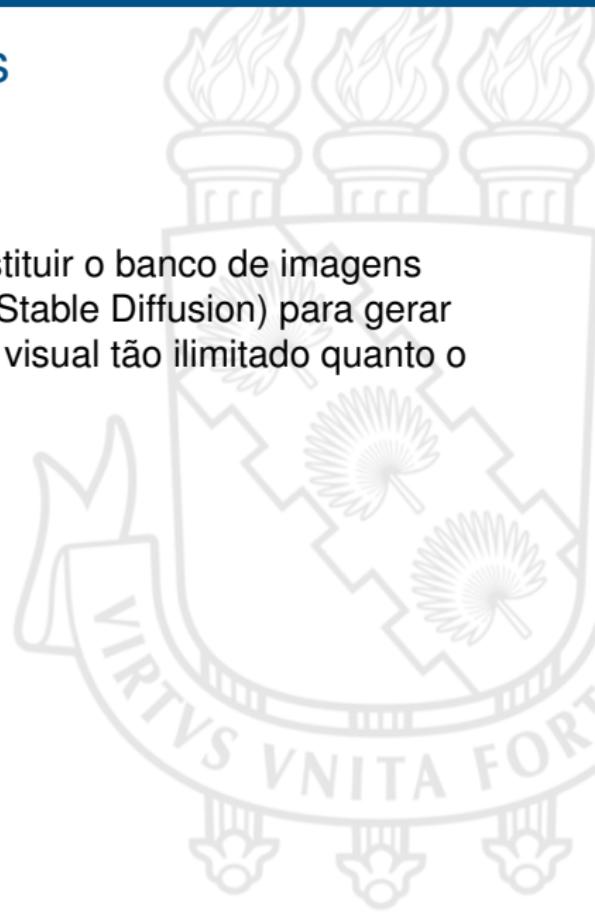
- **Viabilidade Técnica Local:** Os testes mostraram que modelos compactos executados localmente (20b) possuem desempenho comparável a modelos gigantes (120b), provando que a tecnologia é acessível em hardware de consumo, sem necessidade de nuvem.
- **Eficiência Semântica e Coerência:** Através da métrica de Reconstrução Semântica e de Coerência, comprovou-se que o sistema preserva fielmente a intenção do usuário ao transformar texto em elementos de jogo funcionais.

Conclusão e Contribuições

- **Viabilidade Técnica Local:** Os testes mostraram que modelos compactos executados localmente (20b) possuem desempenho comparável a modelos gigantes (120b), provando que a tecnologia é acessível em hardware de consumo, sem necessidade de nuvem.
- **Eficiência Semântica e Coerência:** Através da métrica de Reconstrução Semântica e de Coerência, comprovou-se que o sistema preserva fielmente a intenção do usuário ao transformar texto em elementos de jogo funcionais.
- **Democratização do Desenvolvimento:** A integração entre PCG clássico e IA Generativa reduz barreiras técnicas, permitindo a criação de experiências de jogo customizadas e coerentes através de linguagem natural.

Limitações e Trabalhos Futuros

- **Texturas Dinâmicas com IA:** Substituir o banco de imagens fixo por modelos de difusão (como Stable Diffusion) para gerar texturas em tempo real, tornando o visual tão ilimitado quanto o texto.



Limitações e Trabalhos Futuros

- **Texturas Dinâmicas com IA:** Substituir o banco de imagens fixo por modelos de difusão (como Stable Diffusion) para gerar texturas em tempo real, tornando o visual tão ilimitado quanto o texto.
- **Cenários Complexos:** Evoluir a criação de objetos isolados para a geração de conjuntos de elementos (prefabs), permitindo a construção de micro-cenários mais detalhados e coesos.

Limitações e Trabalhos Futuros

- **Texturas Dinâmicas com IA:** Substituir o banco de imagens fixo por modelos de difusão (como Stable Diffusion) para gerar texturas em tempo real, tornando o visual tão ilimitado quanto o texto.
- **Cenários Complexos:** Evoluir a criação de objetos isolados para a geração de conjuntos de elementos (prefabs), permitindo a construção de micro-cenários mais detalhados e coesos.
- **Expansão de Gêneros e Narrativa:** Adaptar o sistema para outros tipos de jogos (como RPGs), incluindo a criação dinâmica de NPCs, diálogos e missões personalizadas.

Referências I

- [1] Rafael Castro e Silva et al. "Procedural game level generation by joining geometry with hand-placed connectors". Em: (2020).
- [2] Pedro Augusto Nunes Marchand. "APLICAÇÃO DE ELEMENTOS DE STORYTELLING PARA ENTRADAS NA UTILIZAÇÃO DE REGRESSÃO PARA PREDIÇÃO DE MAPAS GERADOS PROCEDURALMENTE". Em: () .
- [3] Shyam Sudhakaran et al. "Mariogpt: Open-ended text2level generation through large language models". Em: **Advances in Neural Information Processing Systems** 36 (2023), pp. 54213–54227.
- [4] Muhammad U Nasir, Steven James e Julian Togelius. "Word2world: Generating stories and worlds through large language models". Em: **arXiv preprint arXiv:2405.06686** (2024).
- [5] André Linard Santos Auxtero. "Game Environment Design Creator Using Artificial Intelligence Procedural Generation". Diss. de mestr. Universidade NOVA de Lisboa (Portugal), 2023.



Referências II

- [6] Roberto Gallotta, Antonios Liapis e Georgios Yannakakis. “Consistent game content creation via function calling for large language models”. Em: **2024 IEEE Conference on Games (CoG)**. IEEE. 2024, pp. 1–4.
- [7] He Yan et al. “Inherent limitations of LLMs regarding spatial information”. Em: **arXiv preprint arXiv:2312.03042** (2023).

Obrigado(a) pela Atenção!

