UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

CAMPUS DE BAURU
FACULDADE DE CIÊNCIAS
Bacharelado em Sistemas de Informação

DIEGO FIGUEIREDO EBURNEO
BERNARDO BETETE DA SILVA FONSECA

USO DE INTELIGÊNCIA ARTIFICIAL EM RPGs DE TEXTO

Bauru
2024

DIEGO FIGUEIREDO EBURNEO
BERNARDO BETETE DA SILVA FONSECA

USO DE INTELIGÊNCIA ARTIFICIAL EM RPGs DE TEXTO

Trabalho de conclusão de curso apresentado ao curso de Sistemas de Informação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Douglas Rodrigues

Bauru
2024

Eburneo, Diego Figueiredo
Uso de inteligência artificial em RPGs de texto / Diego Figueiredo Eburneo, Bernardo Betete da Silva Fonseca. – Bauru, 2024
42 f. : il.

Trabalho de conclusão de curso (bacharelado – Sistemas de Informação)–Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientador: Douglas Rodrigues

1. IA (Inteligência artificial). 2. RPG. 3. Jogos eletrônicos. I. Fonseca, Bernardo Betete da Silva. II. Universidade Estadual Paulista. Faculdade de Ciências. III. Título.

RESUMO

Dentre os nichos do já estabelecido mercado de jogos, um dos mais interessantes se trata dos jogos baseados em texto, devido principalmente a sua importância histórica, contribuindo para as experiências ludo narrativas que gêneros como os RPGs (*Role Playing Games*) viriam a aperfeiçoar no futuro. Também tem-se outros méritos, como a sua facilidade de desenvolvimento e liberdade criativa dado seu ambiente baseado puramente em texto, o que pode proporcionar a entrada de desenvolvedores independentes no mercado, e até mesmo impulsionar áreas de estudo importantes como é o caso da Inteligência Artificial. Dessa forma, este trabalho visa empregar o uso de técnicas de IA para gerar eventos num RPG baseado em texto, assim como a geração procedural para a criação de mapas em ASCII.

Palavras-chave: Inteligência Artificial, text-based, jogos, geração procedural, Grandes Modelos de Linguagem.

ABSTRACT

Among the niches of the already established gaming market, one of the most interesting is that of text-based games, mainly due to their historical importance, contributing to the ludonarrative experiences that genres like Role Playing Games would later perfect. There's also merits such as their ease of development and creative freedom given their purely text-based environment, which can provide entry for independent developers into the market, and even drive important areas of study such as Artificial Intelligence. Therefore, this work aims to employ the use of AI techniques to generate events in a text-based RPG, as well as procedural generation for creating ASCII maps.

Keywords: Artificial Intelligence, text-based, games, procedural generation, Large Language Models.

LISTA DE FIGURAS

Figura 1 - Valor do mercado global de videogames de 2022 a 2032……..……..7
Figura 2 - Zork, série de jogos text-adventure dos anos 80……..…...…...8
Figura 3 - Os Módulos que Compõem a LangChain……11
Figura 4 - Funcionamento do LangChain……..13
Figura 5 - Diagrama do Modelo I.A……..……..……..…19
Figura 6 - Diagrama do Personagem…….....…….…….....20
Figura 7 - Diagrama de classe dos *Map Tiles*……..…..……..,…….....21
Figura 8 - Diagrama dos Menus..……..…….……..……..……..22
Figura 9 - Geração das cenas de forma simplificada……..22
Figura 10 - Diagrama do Menu Inicial……..…….....……..…23
Figura 11 - Diagrama da Geração de Cena……..24
Figura 12 - Diagrama do Menu de Batalha……...25
Figura 13 - Diagrama do Menu do Mercador…….....26
Figura 14 - Classes Scene Ambient.……..……..……..……..……...27
Figura 15 - Prompt de exemplo de cenas em floresta…….27
Figura 16 - Classes Scene Enemy…….……...28
Figura 17 - Classes Scene Merchant…….....28
Figura 18 - Template do mercador……..……..……..…..……..……....28
Figura 19 - Classes Menu……....29
Figura 20 - Classes Personagens……..……..…….....29
Figura 21 - Classes Map e Misc.…….……...……..……..……....29
Figura 22 - Menu Inicial……30
Figura 23 - Menu de Jogo……31
Figura 24 - Cena gerada nas montanhas……..32
Figura 25 - Menu de Batalha……32
Figura 26 - Menu de Diálogo……...……33
Figura 27 - Prompts de geração de terreno……..37
Figura 28 - Especificações para a IA……..…38

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
RL	Reinforcement Learning
IDE	Integrated Development Environment
NPC	Non-Player Character
RPG	Role-Playing Game
ASCII	American Standard Code for Information Interchange
LMM	Large Multimodal Model
LLM	Large Language Model
API	Application Programming Interface

SUMÁRIO

1 INTRODUÇÃO……..…...7

2 SOLUÇÕES EXISTENTES……10
2.1 TextWorld……10
2.2 API GPT-4 da OpenAI……...10
2.3 LangChain……...11
2.4 Comparação……...13

3 TECNOLOGIAS UTILIZADAS……16
3.1 Python……..16
3.2 PyCharm……..16
3.3 Large Language Models……..17
3.4 Geração Procedural……17

4 DESCRIÇÃO DO SISTEMA……...19
4.1 Diagramas de sequência……..…23
4.2 Desenvolvimento das classes do sistema…….27
4.3 Detalhamento do sistema em execução……30
4.4 Como jogar……....33
4.5 Modo de instalação……..35

5 VALIDAÇÃO DO SISTEMA……...37

6 CONCLUSÃO……....39

REFERÊNCIAS……..……..…41

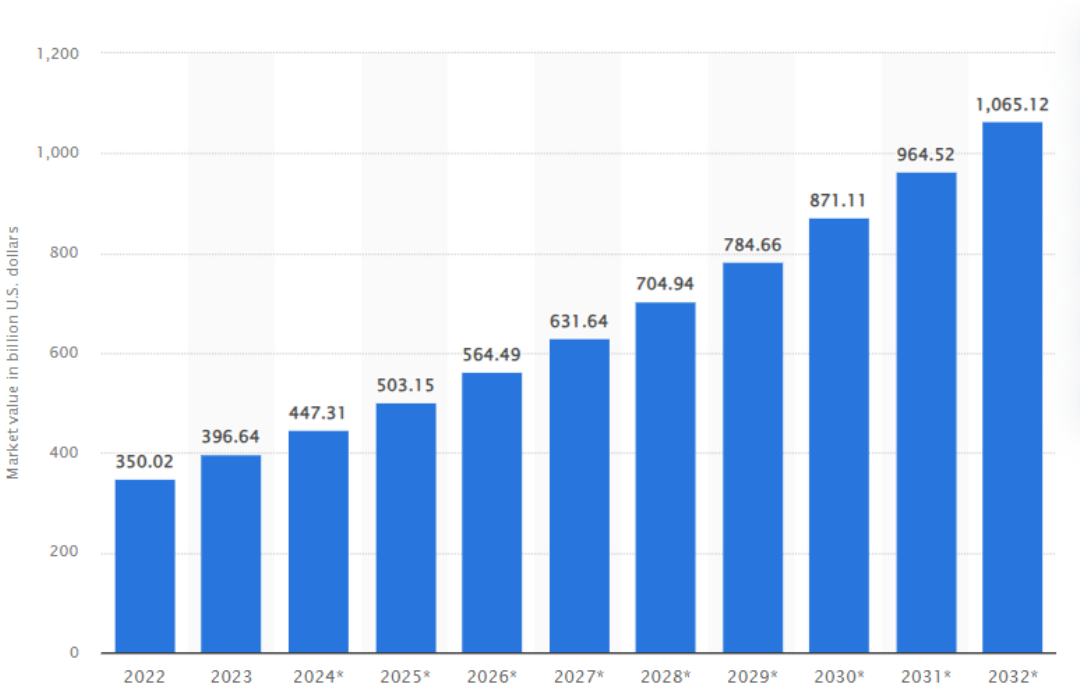
1 INTRODUÇÃO

A indústria de jogos eletrônicos passa por um crescimento de popularidade nunca antes visto, atingindo um sucesso financeiro absoluto no mercado de entretenimento, superando até mesmo gigantes como as indústrias cinematográfica e musical. Em 2023, o setor de games arrecadou aproximadamente US$188 bilhões, enquanto o cinema gerou cerca de US$77 bilhões no mesmo período (VEJA, 2023).

“De acordo com uma reportagem pela SuperData Research, o mercado global de videogames foi avaliado em US$159.3 bilhões em 2020. Para colocar em perspectiva, a indústria de música foi avaliada em US$19.1 bilhões em 2020, enquanto a indústria de filmes foi avaliada em US$41.7 bilhões no mesmo ano.” (GAMERHUB, 2023).

Além disso, conforme as ferramentas de software de desenvolvimento se tornam mais acessíveis e robustas, observa-se uma rápida crescente no mercado independente de jogos, no qual novos desenvolvedores e estúdios possuem uma chance de mostrar seu talento nesse já consolidado mercado. A Figura 1 demonstra a previsão de crescimento desse mercado até 2032:

Figura 1 - Valor do mercado global de videogames de 2022 a 2032

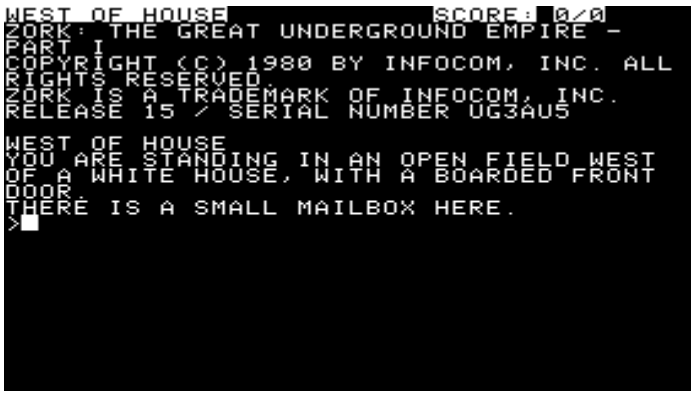


Fonte: Statista (2024)

Jogos *indie* (independentes) são desenvolvidos por uma pequena equipe, ou até mesmo uma única pessoa, que não possuem vínculos ou apoio financeiro de empresas externas maiores, portanto sua distribuição também é de responsabilidade dos desenvolvedores. Hoje no mercado de softwares de desenvolvimento de jogos, observa-se um foco maior em desenvolvedores independentes através de programas acessíveis e de interface amigável que os auxiliam no processo de desenvolvimento. Plataformas como Stencyl, Construct 3, GameMaker Studio 2, Unity e Unreal Engine oferecem recursos que facilitam a criação de jogos, mesmo para aqueles com pouca experiência em programação (VIDEOGAME WARLOCK, 2024).

Uma das principais características desse mercado é a variada gama de gêneros disponíveis, naturalmente, alguns gêneros se destacam mais que os outros no quesito popularidade, dominando a maior fatia do mercado. Dentre os nichos mais antigos, temos os jogos baseados em texto, nos quais a interface de usuário é puramente baseada em texto e caracteres modificáveis, como ASCII, e geralmente se tratam de aventuras em que o usuário envia comandos simples para interagir com o mundo e seus eventos, como no exemplo apresentado na Figura 2.

Figura 2 - Zork, série de jogos *text-adventure* dos anos 80



Fonte: The Digital Antiquarian (2012)

Levando em consideração a simplicidade desse gênero, dado seu ambiente baseado puramente em texto, observa-se um grande potencial para não somente a entrada de novos desenvolvedores no mercado, como também a aplicação de Inteligência Artificial (IA) para geração de eventos e mundos nesses jogos, o que pode levar a avanços nos estudos dessa área.

Este trabalho tem como principal objetivo o desenvolvimento de um jogo *text-based* utilizando a IA e Grandes Modelos de Linguagem (Large Language Models -

LMMs) para gerar os eventos, história, diálogos, personagens e missões, que contará também com cenários gerados proceduralmente e exibidos em ASCII. Alguns objetivos específicos são:

- Criar uma interface que possibilite ao jogador uma experiência imersiva e a capacidade de interagir com eventos gerados pela IA;
- Estabelecer um ciclo de *gameplay* no qual o jogador se fortalece através de batalhas e trocas com NPCs com o objetivo de enfrentar um inimigo final; e
- Disponibilizar o sistema criado para servir como base para outros projetos *text-based* de desenvolvedores independentes.

No Capítulo 2 serão abordadas as diferentes soluções que se assemelham ao projeto proposto. O Capítulo 3 elabora a respeito das tecnologias utilizadas para o desenvolvimento do sistema. A seguir, no Capítulo 4, será descrito o jogo e funcionamento planejado do sistema com diagramas de classe e sequência, detalhando a estrutura do programa desde a implementação da IA até a interação do jogador com eventos. Já no Capítulo 5 são apresentados os métodos de validação. Por fim, no Capítulo 6 demonstra-se os resultados obtidos, sugestões de aprimoramento e possíveis usos futuros do trabalho.

2 SOLUÇÕES EXISTENTES

Como parte desse trabalho, foi realizada uma pesquisa acerca de soluções disponíveis e formas catalogadas de utilização de IA em jogos *text-based.* Após a busca, algumas soluções se destacaram, porém apresentam vantagens e desvantagens nas quais uma solução pode se sobressair.

2.1 TextWorld

Desenvolvido pelo Laboratório de Pesquisas da Microsoft em Montreal, se trata de um *framework* Python *open-source* para geração de jogos *text-based*, no qual se tem um ambiente de aprendizado por reforço para o treino de agentes.

Por meio do TextWorld, esses agentes de IA podem ser treinados em habilidades como compreensão de linguagem natural, memorização, planejamento, exploração, dentre outras. Pesquisadores podem até mesmo usar dessa tecnologia para estudar no contexto de *Transfer Learning* (Aprendizado por Transferência) e Generalização.

TextWorld possui dois componentes principais, um gerador de jogo, e a *engine* de jogo propriamente dita. O gerador converte especificações de alto nível do jogo (níveis, número de objetos e sala, tamanho do jogo) em código fonte para o executável na linguagem Inform 7. A *engine* se trata de uma simples máquina de inferência que garante que cada passo do jogo gerado seja válido usando algoritmos como *one-step forward* e *backward chaining.*

No seu estado atual, o TextWorld cumpre o objetivo de geração de eventos e texto usando IA, porém, é limitado quanto à dinamicidade destes eventos, trazendo uma proposta mais estruturada e linear. Além disso, o framework possui aplicabilidade apenas em sistemas Linux, limitando sua usabilidade e acessibilidade.

2.2 API GPT-4 da OpenAI

A API do GPT-4, desenvolvida pela OpenAI, oferece uma interface que permite aos desenvolvedores incorporar as capacidades avançadas de linguagem do modelo GPT-4 em suas aplicações. Treinado em um extenso conjunto de dados textuais, o GPT-4 é capaz de realizar diversas tarefas relacionadas a texto, incluindo geração de conteúdo, tradução, resumo, resposta a perguntas e outras aplicações linguísticas.

Esta API, semelhante a outras APIs de modelos de linguagem, tem seu funcionamento baseado em *prompts*, instruções em texto enviadas pelo usuário ao modelo, que então gera uma resposta baseada no contexto fornecido. A interação com a API é simples e pode ser feita usando várias linguagens de programação, como Python.

Uma API como o GPT-4 se trata de uma solução poderosa para a criação de aventuras em texto adaptativas. Esses modelos podem gerar conteúdo procedural e responder de forma dinâmica às ações do jogador.

Tratando-se de uma API externa, ela pode ser custosa dependendo do tamanho dos *prompts* utilizados. Para o nosso sistema, o custo seria mínimo, porém, faz parte do nosso objetivo criar um jogo acessível e sem custos.

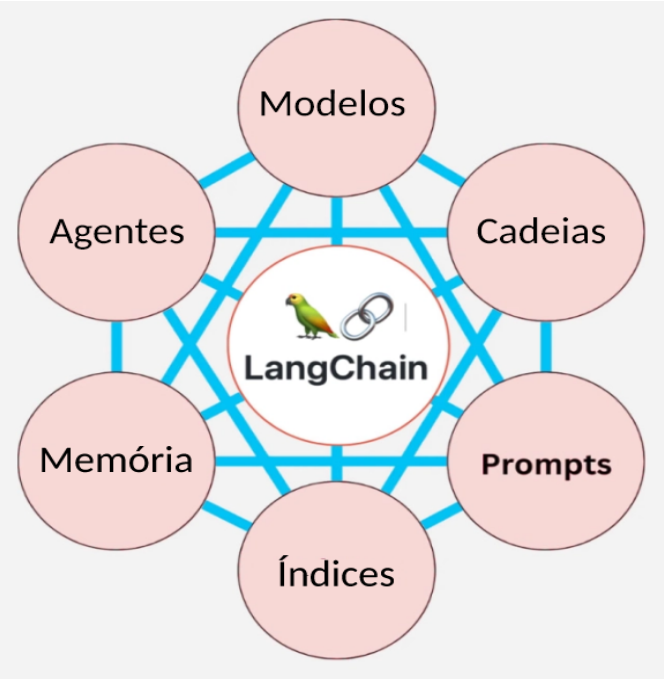
2.3 LangChain

LangChain se trata de uma *framework* para o desenvolvimento de aplicações utilizando LLMs. Disponível em bibliotecas Python e Java, as ferramentas e APIs do LangChain simplificam o processo de criação de aplicativos baseados em LLM, como chatbots e agentes virtuais.

Os LLMs não são aplicativos independentes: são modelos estatísticos treinados previamente que devem ser combinados com um aplicativo (e em alguns casos fontes de dados específicas) para atender sua finalidade.

Na Figura 3 tem-se uma ilustração dos módulos que compõem a LangChain, sendo esses:

Figura 3 - Os Módulos que Compõem a LangChain



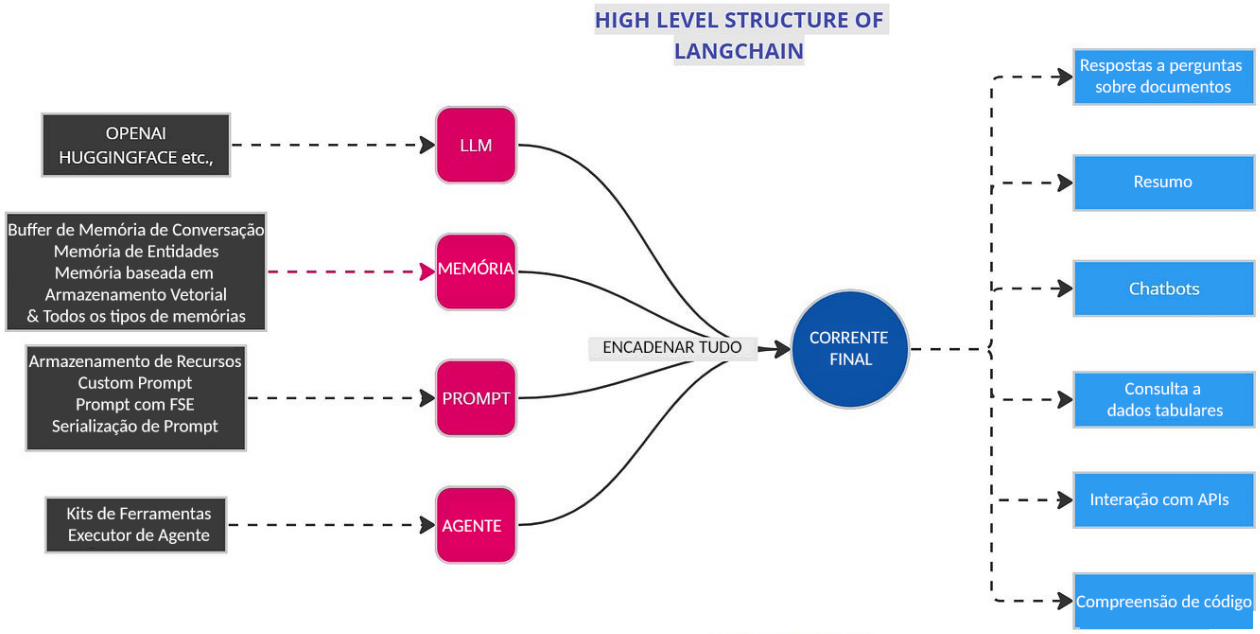
Fonte: Adaptado de ByteByteGo (2023)

- Modelos: São os dos Grandes Modelos de Linguagem. O LangChain serve de *framework* para as interações com estes modelos, como por exemplo, o LLaMA, modelo criado pela Meta AI, que foi utilizado neste sistema como modelo local;
- *Prompts*: O LangChain oferece uma variedade de ferramentas e classes que auxiliam a criação de prompts;
- Memória: Um recurso fundamental para chatbots é a incorporação de módulos de memória dedicados ao gerenciamento e alteração de conversas para que relembrem interações anteriores;
- Agentes: Os agentes inteligentes são equipados pelo LangChain com um kit de ferramentas amplo que é utilizado com base nas informações do usuário;
- Índices: Índices se referem à maneiras de estruturar documentos a fim de que Grandes Modelos de Linguagem interajam com estes da melhor maneira; e

- *Chain*: Usar apenas um modelo pode ser o suficiente para aplicações mais simples, porém o LangChain fornece uma interface padrão para encadeamento de modelos para aplicações mais complexas.

Na Figura 4 tem-se um diagrama demonstrando o funcionamento em conjunto dos módulos apresentados na Figura 3, ilustrando em alto nível a estrutura do LangChain.

Figura 4 - Funcionamento do LangChain



Fonte: Adaptado de Towards AI (2023)

2.4 Comparação

Comparando as soluções apresentadas observa-se, nos seguintes critérios, que:

- Usabilidade para RPG baseado em texto: O LangChain facilita a construção de aplicações de IA baseadas em linguagem, integrando modelos de linguagem de maneira flexível e modular, assim como a API GPT-4, enquanto o TextWorld é focado em jogos de aventura em texto com lógica estruturada, sendo adequado para RPGs mais lineares;
- Facilidade de Integração: O LangChain é altamente integrável com vários LLMs, permitindo fácil customização e extensibilidade, o GPT-4

possui integração direta e rápida via API, enquanto o TextWorld requer uso do sistema TextWorld Gym, mais complexo de integrar com modelos de linguagem não suportados diretamente;

- Criação de cenários: LangChain e GPT-4 oferecem estruturas modulares e *pipelines* para construção de cenários adaptativos, com eventos variados em tempo real, já o TextWorld requer a criação manual de eventos e tarefas com lógica específica, sendo mais limitado em flexibilidade narrativa;
- Acessibilidade: A API GPT-4, tratando-se de uma API externa, possui um custo que pode ser desinteressante para projetos novos de desenvolvedores independentes. O TextWorld, apesar de gratuito, possui aplicabilidade somente em sistemas Linux, reduzindo tremendamente sua acessibilidade, enquanto o LangChain, também gratuito, possui suporte não só a Windows, como também a Linux;
- Apoio a Geração Procedural: Tem-se no LangChain e no GPT-4 uma compatibilidade com geração procedural via modelos de IA, ideal para criar conteúdo dinâmico, enquanto o TextWorld, apesar de permitir a geração procedural de mapas e desafios, o faz com um foco mais estruturado e menos adaptativo, fugindo do propósito de nosso sistema.

Considerando as opções apresentadas e o resumo das mesmas apresentado no Quadro 1, neste sistema utilizamos o LangChain, pois é o melhor para a criação de cenários complexos com eventos dinâmicos, diálogos adaptáveis, geração procedural em tempo real, além de maior acessibilidade, dado que é gratuito e disponível tanto para Windows quanto Linux.

Quadro 1 - Comparação das Soluções Existentes

Critério	TextWorld	API GPT-4	LangChain
Usabilidade para RPGs	Adequado para jogos lineares e estruturados	Permite criação de aventuras adaptativas	Ideal para construção modular e dinâmica
Criação de Cenários	Limitada e manual	Geração procedural dinâmica	Estruturas adaptativas em tempo real
Acessibilidade	Gratuito, mas restrito ao Linux	Custos elevados dependendo do uso	Gratuito e multiplataforma
Integração	Requer ambiente Linux	Simples via API	Altamente integrável com vários LLMs

Fonte: Elaborado pelos autores (2024)

3 TECNOLOGIAS UTILIZADAS

3.1 Python

Python é a terceira linguagem de programação mais utilizada no mundo (Coursera, 2024), e a mais utilizada quando se trata de IA (UpWork, 2023). Sua sintaxe simples aliada ao modelo de código aberto favorece a existência de uma grande biblioteca criada por uma comunidade muito engajada.

Esta linguagem também oferece uma extensa biblioteca de ferramentas e frameworks que facilitam a integração com modelos de linguagem, como LangChain, permitindo a criação de interações dinâmicas e narrativas geradas proceduralmente. Essas características tornam Python a linguagem ideal para o sistema.

3.2 PyCharm

O PyCharm é uma das Ferramentas de Desenvolvimento Integradas (Integrated Development Environment - IDEs) mais populares e amplamente utilizadas para programação em Python. Desenvolvido pela JetBrains, o PyCharm oferece uma ampla gama de recursos que tornam o desenvolvimento de projetos baseados em Python mais eficiente e organizado.

Acredita-se que para o desenvolvimento deste sistema, as funcionalidades que o PyCharm trás serão essenciais, sendo essas:

- Autocompletar Inteligente: Facilita a escrita de código, reduzindo erros e aumentando a produtividade. Isso é especialmente útil na integração do Python com o framework LangChain, que exige o uso de bibliotecas específicas e estruturas complexas;
- Suporte a Bibliotecas Externas: O PyCharm permite a configuração fácil de ambientes virtuais, o que é crucial para gerenciar as dependências do sistema, como o LangChain e bibliotecas de IA;
- Gestão de Projetos: A estrutura de projetos do PyCharm permite organizar eficientemente os arquivos do sistema, incluindo os scripts de geração de cenas, menus, e o modelo de IA local; e
- Suporte a Git: Durante o desenvolvimento colaborativo, o PyCharm foi integrado ao repositório GitHub do jogo, simplificando o versionamento e o compartilhamento de código entre os desenvolvedores.

Com esses recursos, o PyCharm se mostrou como a escolha ideal para este sistema.

3.3 *Large Language Models*

LLMs se tratam de uma classe de modelos base, treinados com enormes quantidades de dados para fornecer as capacidades fundamentais necessárias para impulsionar múltiplos casos de uso e aplicações, bem como resolver uma infinidade de tarefas. Isso contrasta fortemente com a ideia de construir e treinar modelos específicos de domínio para cada um desses casos de uso individualmente.

Em suma, LLMs são desenvolvidos para interpretar e gerar texto como humanos, em adição a outras formas de conteúdo, baseado na vasta quantidade de dados utilizados em seu treinamento. São capazes de realizar essas tarefas devido a bilhões de parâmetros que capacitam a captura de padrões de linguagem e realizam uma vasta gama de tarefas relacionadas à palavras.

3.3.1 LLaMA 2 *Model*

Neste sistema, optamos por utilizar um modelo de chat local LLaMA 2, devido ao custo de APIs externas online como o GPT-4 e sua eficiência como modelo para geração de textos dinâmicos e simples. No momento, o modelo LLaMA 2 tem quase todo seu treinamento realizado em inglês, tendo maior eficácia nessa língua, portanto, os textos gerados neste sistema assim como as partes interativas são em inglês.

3.4 Geração Procedural

A geração procedural é utilizada para criar conteúdo de maneira dinâmica, sem depender de recursos estáticos pré-criados, sendo empregada, normalmente, na criação de cenários para jogos. Muito utilizada no gênero *roguelike*, como por exemplo o título *Dead Cells*, e também em jogos de mundo aberto, como o *Minecraft,* um dos exemplos mais icônicos, utilizando geração procedural para criar mundos vastos com biomas, cavernas, montanhas e estruturas. A principal vantagem que a geração procedural trás é o aumento da *replayabilidade*, isto é, a diversificação de cada nova jogatina, proporcionando um tempo de jogo maior e mais interessante para o jogador.

Neste sistema, esta tecnologia será utilizada para a geração de mapas que

são aleatórios a cada jogatina, porém contando sempre com os mesmos elementos de terreno e eventos. Para o propósito deste sistema, nos baseamos num projeto de geração procedural já existente[1].

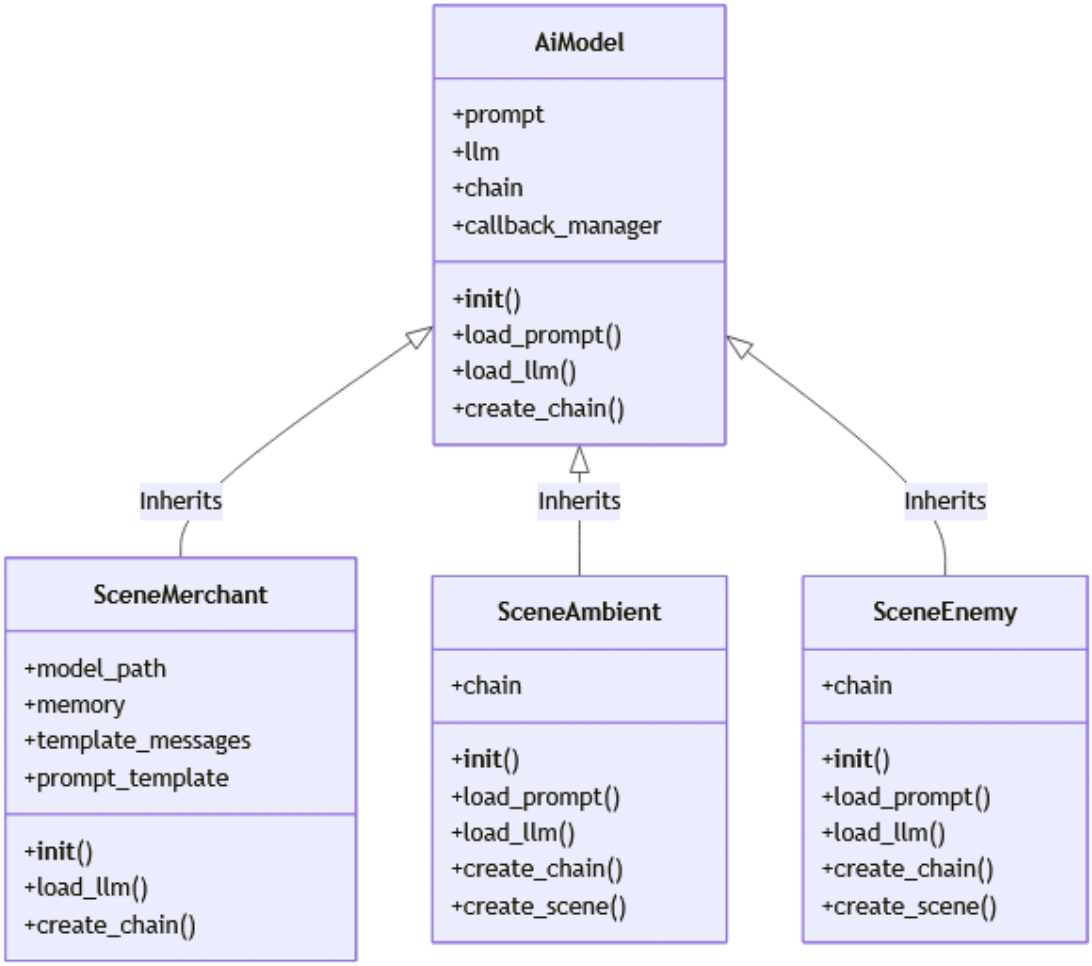
1 Disponível em: <https://github.com/orkslayergamedev/random-ascii-map-generation>

4 DESCRIÇÃO DO SISTEMA

O jogo, disponibilizado em um repositório no GitHub[2], foi desenvolvido com base nos padrões do gênero de jogos *text-RPG*, sendo composto inicialmente por menus que permitem interação no terminal, eventos e cenas gerados por IA, um mapa gerado proceduralmente, e a opção de salvar e carregar o progresso do jogador. Na sequência inicia-se a implementação do modelo IA e das classes que o utilizam.

O diagrama da Figura 5 representa o funcionamento do modelo IA, que recebe prompts específicos das classes *scene* para a geração e exibição de cenas ao jogador.

Figura 5 - Diagrama do Modelo IA

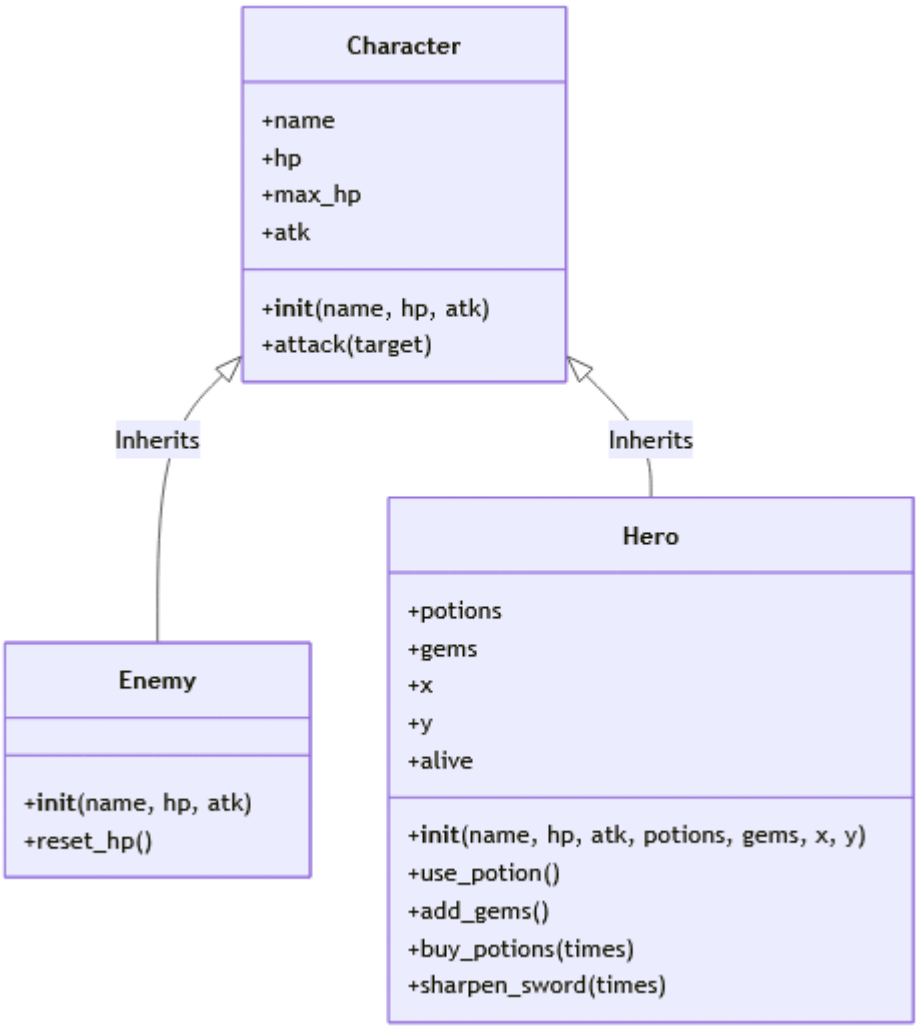


Fonte: Elaborado pelos autores (2024)

2 Disponível em: <https://github.com/bernardobsfonseca/ai-text-rpg>

Na Figura 6 tem-se um diagrama detalhando os atributos dos personagens, ou seja, do jogador e dos inimigos. O jogador possui nome, pontos de vida, pontos de ataque e posição no mapa, além de poções e um item para afiar a espada, aumentando o ataque. Os inimigos possuem nomes, pontos de vida e pontos de ataque.

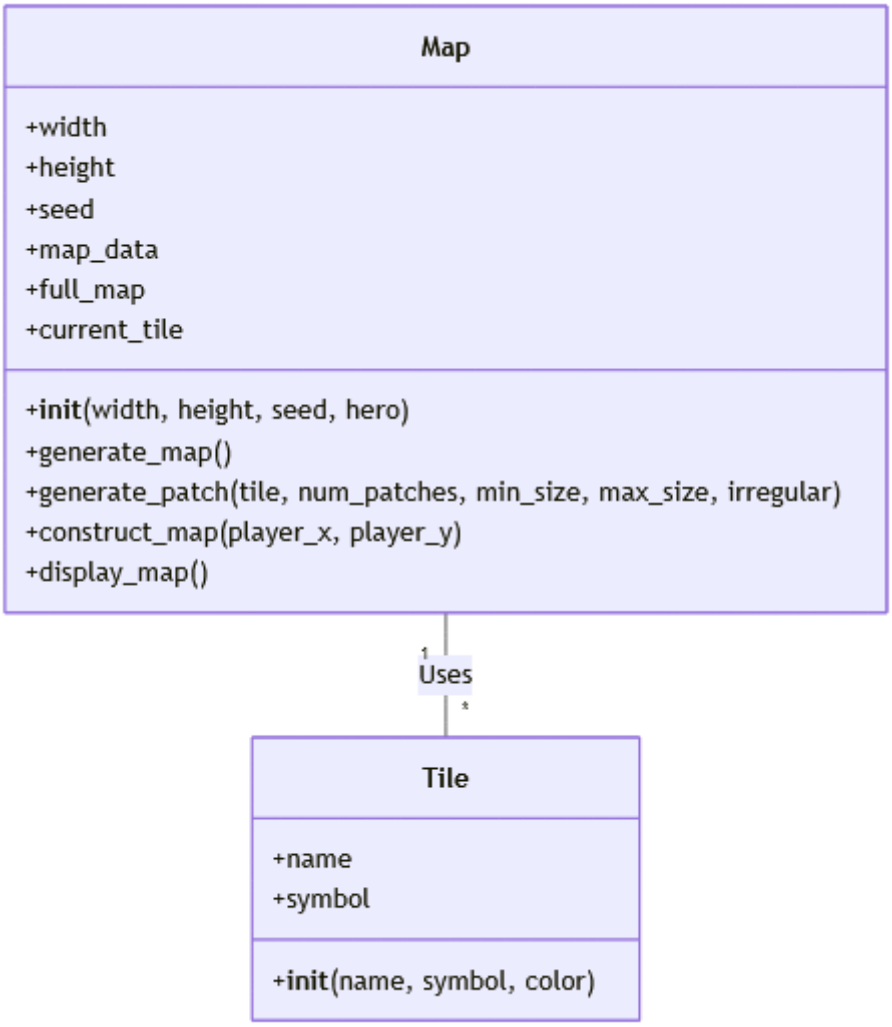
Figura 6 - Diagrama do Personagem



Fonte: Elaborado pelos autores (2024)

O diagrama da Figura 7 detalha os atributos utilizados para a geração procedural do mapa em que o jogo se passa, utilizando *tiles* (blocos correspondentes à coordenadas no espaço) como unidades na grade do mapa, cada *tile* representa uma posição e uma área diferente.

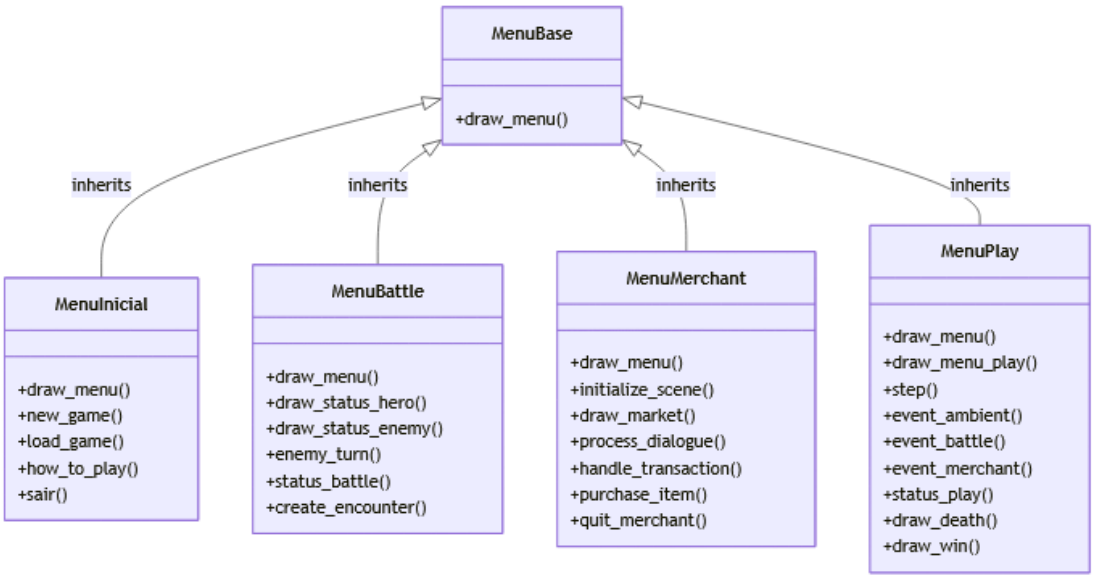
Figura 7 - Diagrama de classe dos *Map Tiles*



Fonte: Elaborado pelos autores (2024)

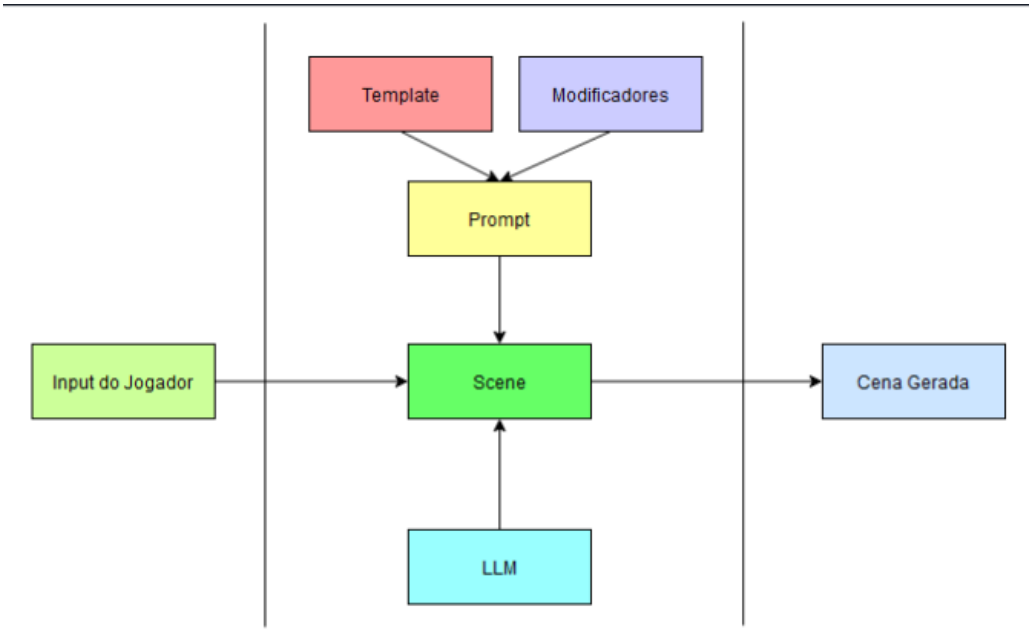
Na Figura 8, o diagrama dos menus estabelece a lógica das interfaces de interação presentes no jogo, todos remontando à uma classe base que inicializa e desenha os menus específicos. Enfim, na Figura 9, um diagrama que apresenta de forma simples e resumida o processo de criação de uma cena.

Figura 8 - Diagrama dos Menus



Fonte: Elaborado pelos autores (2024)

Figura 9 - Geração das cenas de forma simplificada



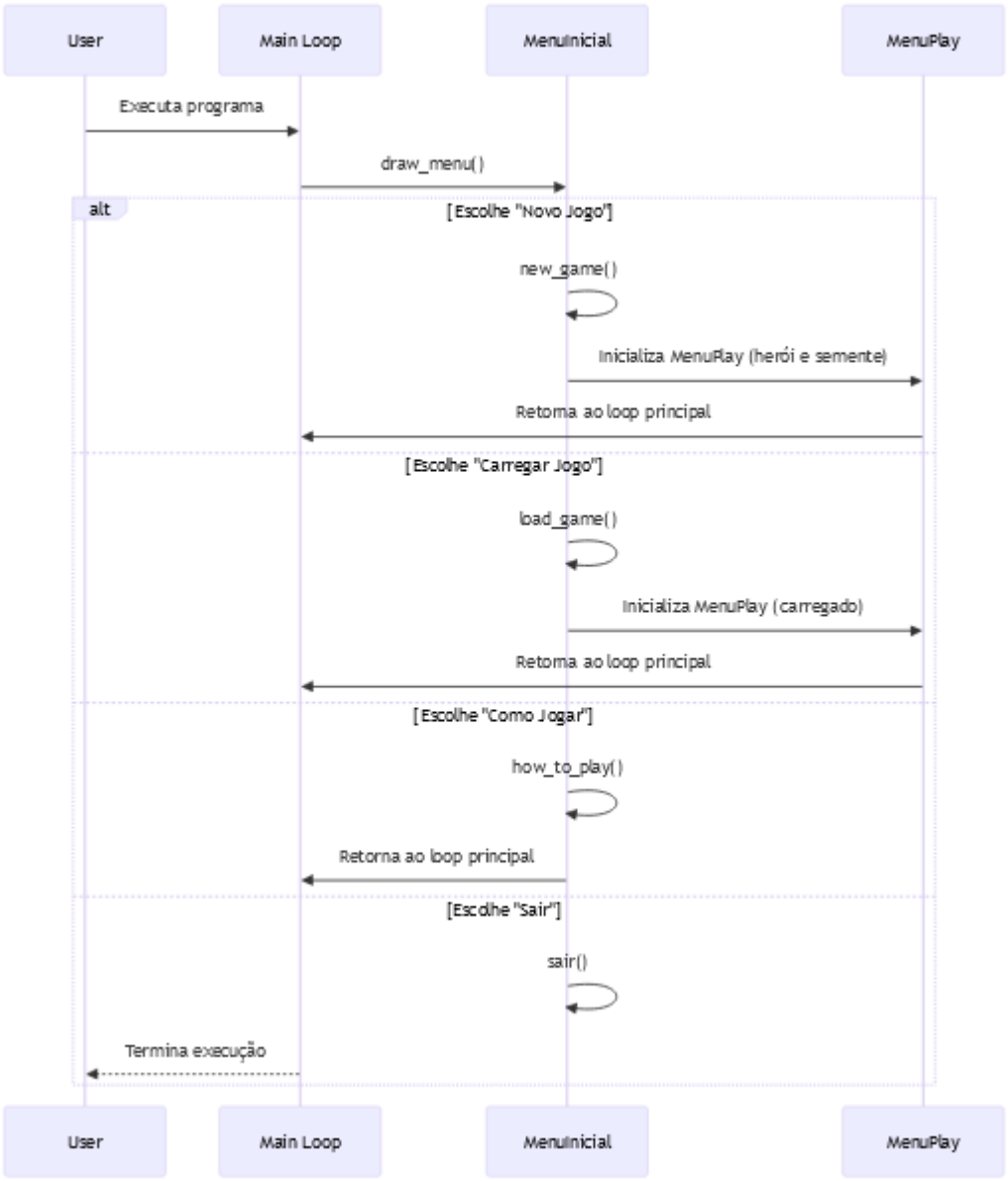
Fonte: Elaborado pelos autores

4.1 Diagramas de sequência

A seguir tem-se os diagramas de sequência elaborados para detalhar os ciclos de funcionamento e utilização dos componentes do jogo pelo usuário.

Na Figura 10 tem-se um diagrama de sequência que fornece uma visualização do funcionamento e ciclo do menu inicial do jogo, detalhando as opções disponíveis.

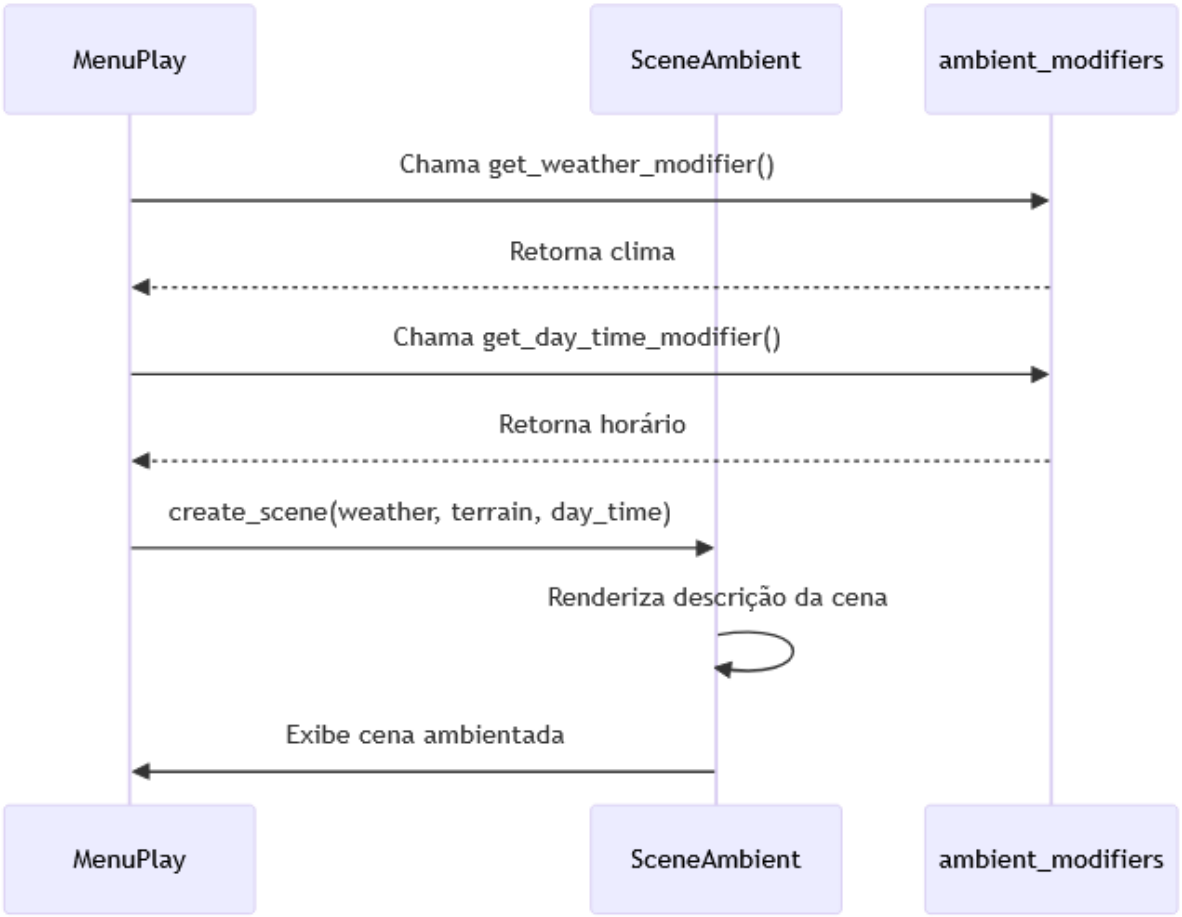
Figura 10 - Diagrama do Menu Inicial



Fonte: Elaborado pelos autores (2024)

Na Figura 11, um breve diagrama que detalha o ciclo de geração de cenas. A classe *SceneAmbient* renderiza as cenas com base em *prompts* com exemplos de cenas e descrições, e com modificações baseadas nos modificadores de clima e tempo do dia presentes no *ambient_modifiers*.

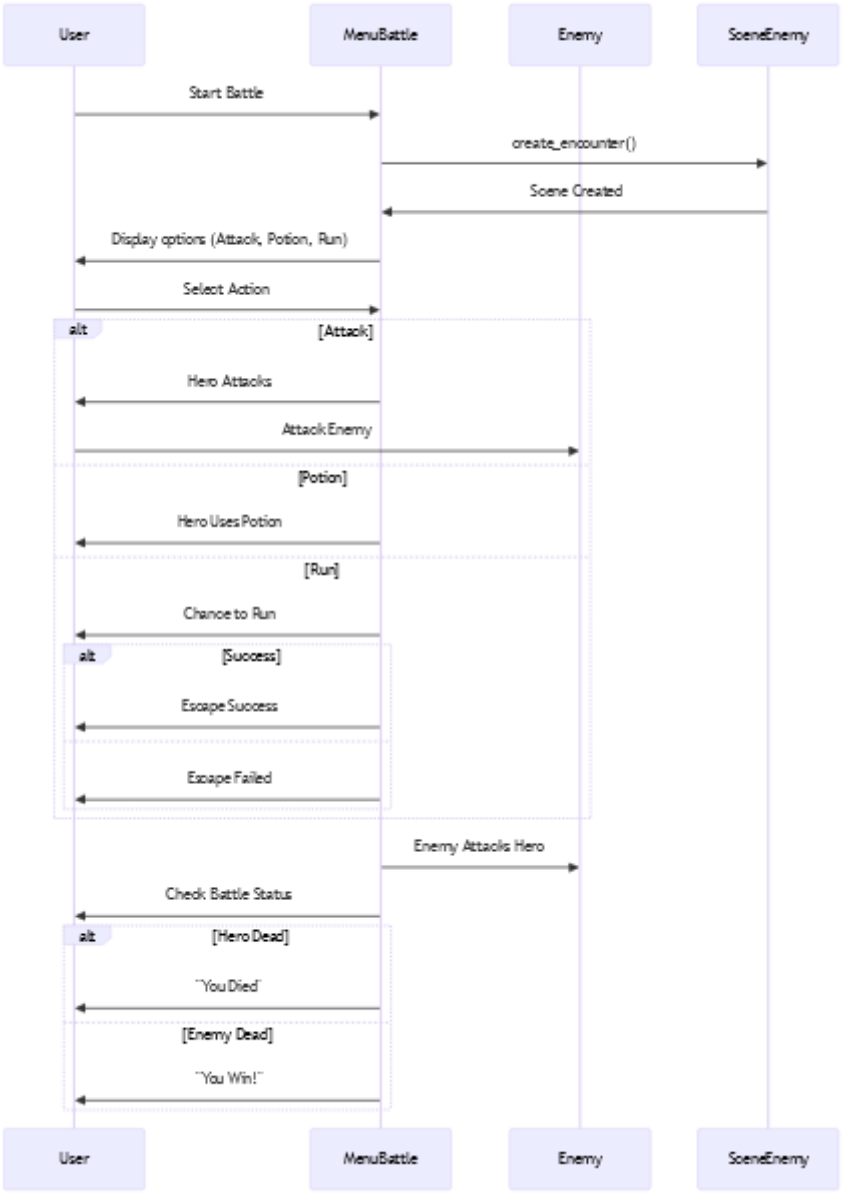
Figura 11 - Diagrama da Geração de Cena



Fonte: Elaborado pelos autores (2024)

O diagrama da Figura 12 apresenta o menu de batalha e a forma como o ciclo de batalha ocorre, detalhando as opções de ação disponíveis ao jogador.

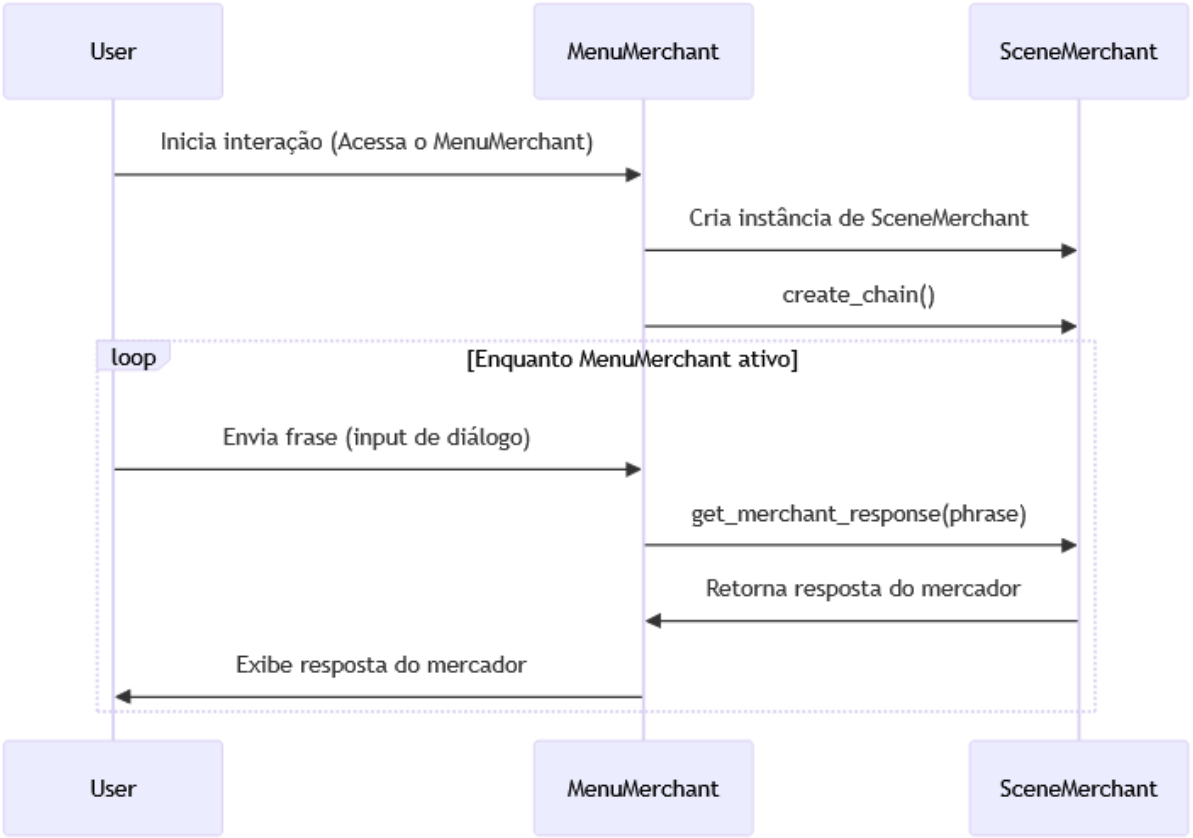
Figura 12 - Diagrama do Menu de Batalha



Fonte: Elaborado pelos autores (2024)

Na Figura 13, detalha-se o ciclo de interação entre o usuário e o mercador, cujos diálogos são gerados por IA em resposta ao *input* de diálogo.

Figura 13 - Diagrama do Menu do Mercador



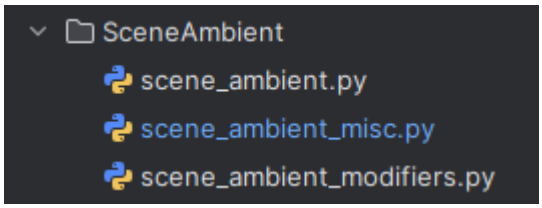
Fonte: Elaborado pelos autores (2024)

4.2 Desenvolvimento das classes do sistema

O jogo foi desenvolvido tendo como base três grupos de classes: *scenes*, *menus* e *characters*.

As *scenes* (cenas), se dividem entre *ambient scene,* tratando-se da cena de ambiente, responsável pela ambientação, e *enemy scene,* sendo esta a cena do inimigo, responsável pela apresentação dos inimigos nos momentos de combate. O nome scene foi escolhido pois, quando requisitada, a IA apenas fará a descrição de uma cena para o jogador. Estas requisições estão definidas nas classes *scene_ambient.py*, *scene_enemy.py* e *scene_merchant.py*, presentes na Figura 14.

Figura 14 - Classes *Scene Ambient*



Fonte: Elaborado pelos autores (2024)

Na classe *scene_ambient_misc.py* foram definidos os *prompts* com exemplos de cenários e descrições para a geração da cena, como demonstrado na Figura 15. A tradução do prompt exemplificado para a cena fica como: “Uma floresta chuvosa durante um pôr do sol”, e para a descrição: “À medida que o sol se põe por trás da densa copa das árvores, chuvas suaves criam uma atmosfera serena na floresta”.

Figura 15 - Prompt de exemplo de cenas em floresta

```
examples = [
    {
        "scene": "A raining forest in sunset",
        "description": "As the sun sets behind the thick canopy, gentle rain showers create a serene ambiance in the forest.
    },
```

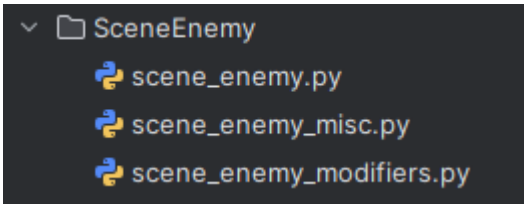
Fonte: Elaborado pelos autores (2024)

Na classe *scene_ambient_modifiers.py* estão localizados os modificadores de cena, palavras como “*stormy*”, “*sunny”* e “*night*”, que são utilizadas pelo *scene_ambient.py* para definir modificações correspondentes na cena gerada.

A mesma estrutura de código se faz presente nas classes *scene_enemy* apresentadas na Figura 16, porém, os *prompts* e modificadores utilizados são

exemplos de descrição de inimigos, como por exemplo: "*A furious sylph in a rainy forest of pines*".

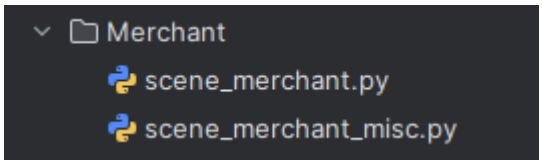
Figura 16 - Classes *Scene Enemy*



Fonte: Elaborado pelos autores (2024)

Na classe *scene_merchant_misc.py*, é definido o *template* com as instruções de como o NPC mercador deve agir em seus diálogos, como apresentado nas Figuras 17 e 18, a tradução sendo: “Aja como um mercador medieval que vende apenas poções (somente de cura, 15 gemas) e oferece o serviço de afiar espadas (30 gemas). Suas frases devem ter no máximo 50 palavras.”

Figura 17 - Classes *Scene Merchant*



Fonte: Elaborado pelos autores (2024)

Figura 18 - *Template* do mercador

```
template = """
            Act like a medieval merchant who sells only potions (healing only, 15 gems)
            and sharpen sword (30 gems).

            Your sentences must have a maximum of 50 words.
            """
```

Fonte: Elaborado pelos autores (2024)

Os *menus*, definidos nas classes apresentadas na Figura 19, se dividem entre menu inicial, onde o jogador poderá começar e carregar uma partida ou ver as regras, *menu play,* o menu de jogo onde o jogador irá interagir com o mapa, *menu battle*, aqui o jogador irá lutar contra inimigos, e *menu merchant*, no qual o jogador pode interagir com o NPC mercador para a aquisição de itens. Os *menus* compõem a parte interativa do jogo, onde o jogador poderá tomar decisões e interagir com o mapa e itens.

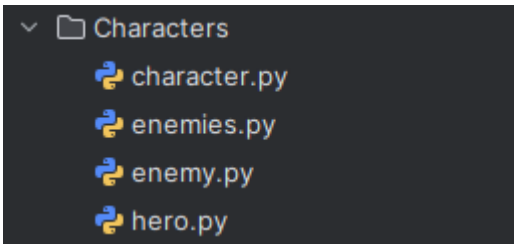
Figura 19 - Classes Menu



Fonte: Elaborado pelos autores (2024)

Os *characters*, vide Figura 20, são os personagens que irão aparecer durante o jogo, e se dividem entre o *player* ou *hero*, jogador/herói, personagem que o jogador controla, e os *enemies*, inimigos, que o jogador irá enfrentar nas batalhas.

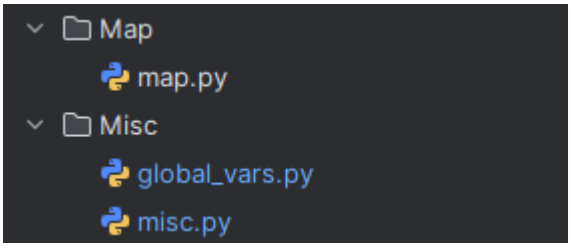
Figura 20 - Classes Personagens



Fonte: Elaborado pelos autores (2024)

Além disso existem os componentes secundários: *map*, o mapa do jogo, e a classe *misc*, na Figura 21, abreviação de “miscelânea”, conjunto de diferentes funções que serão utilizadas ao longo da programação do jogo.

Figura 21 - Classes Map e Misc



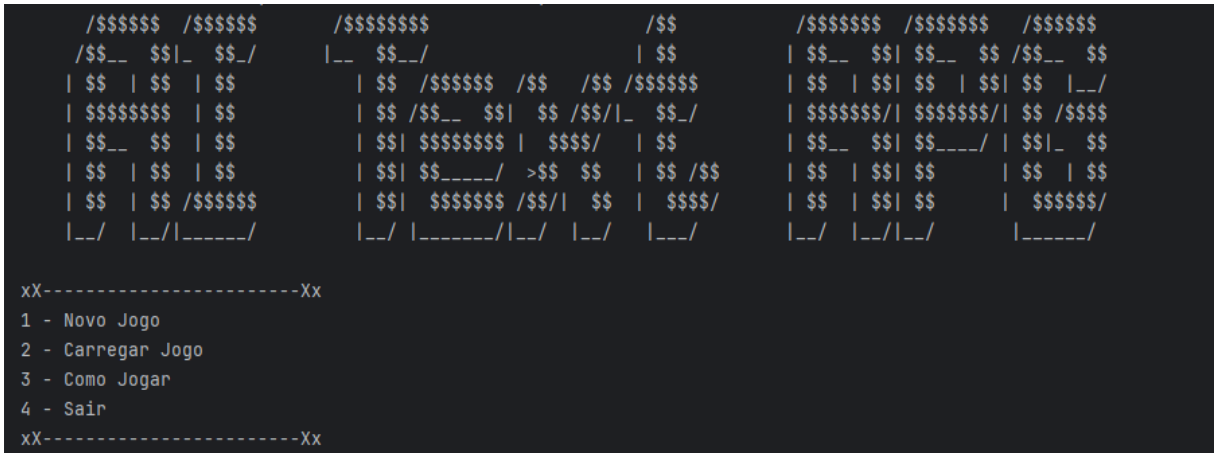
Fonte: Elaborado pelos autores (2024)

4.3 Detalhamento do sistema em execução

Na sequência, uma demonstração do jogo em funcionamento, contando com um detalhamento das telas e menus presentes:

- Menu Inicial: Apresentado na Figura 22, aqui o jogador encontrará as opções de criar um novo jogo, ou carregar um anterior, além do acesso ao “Como Jogar”, que detalha as instruções de *gameplay* e objetivos do jogo. O jogador deve selecionar a opção desejada digitando o número correspondente no terminal.

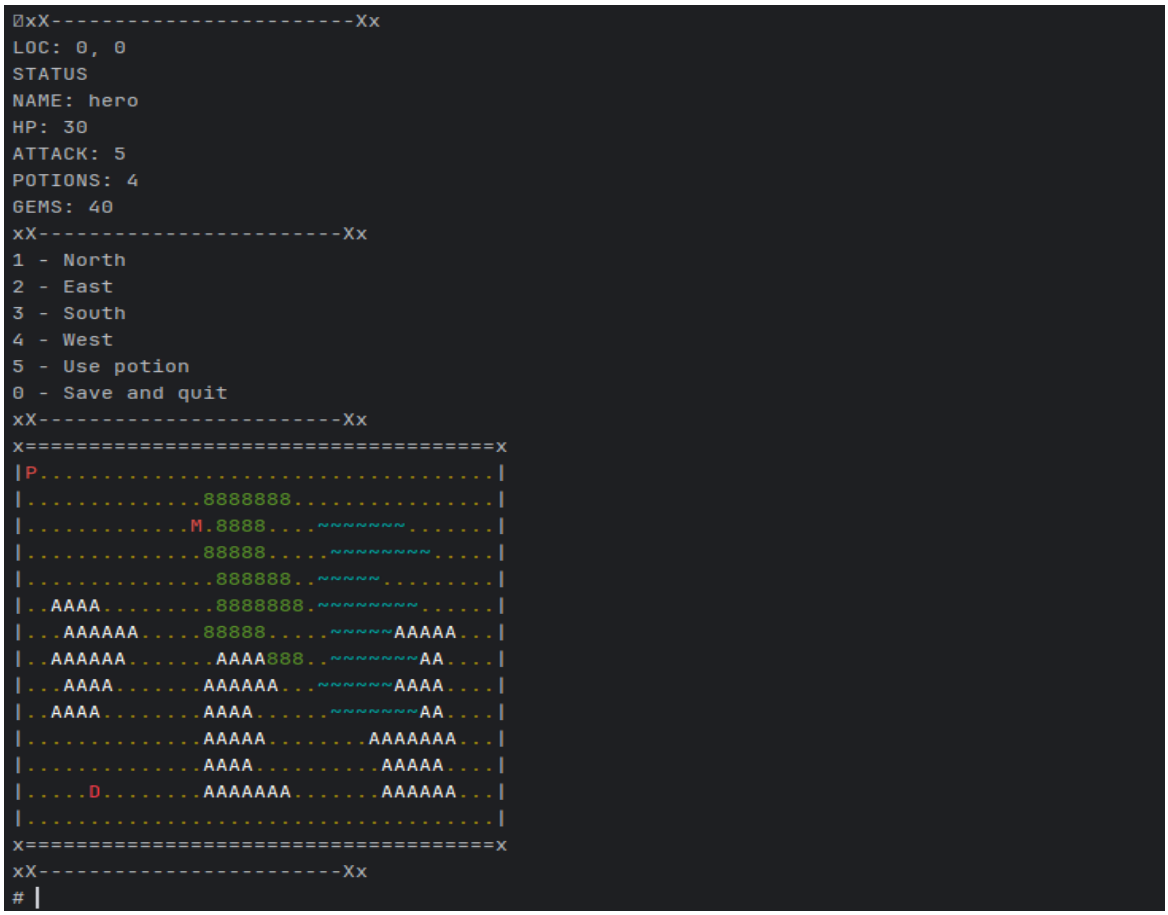
Figura 22 - Menu Inicial



Fonte: Elaborado pelos autores (2024)

- Menu de Jogo: é o principal componente do jogo, com todas as informações sobre o jogador, inclusive o mapa do jogo e sua localização. Na Figura 23 observa-se, além do menu de opções do jogo, o mapa gerado proceduralmente que conta com os seguintes elementos:

Figura 23 - Menu de Jogo



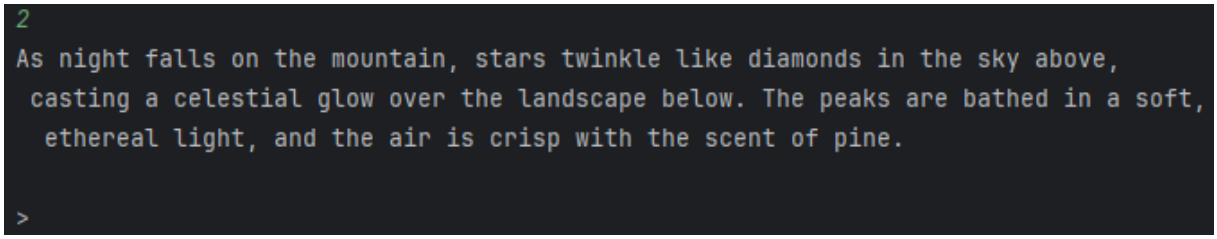
Fonte: Elaborado pelos autores (2024)

- Letra P: Representa o personagem do jogador, sendo controlado diretamente pelo mesmo.
- Letra M: Representa o NPC Mercador, ao mover-se à sua localização, inicia-se o menu de diálogo para troca de itens.
- Letra D: Representa o Dragão, inimigo mais difícil e objetivo final do jogo.
- *Tiles* brancos: Regiões montanhosas.
- *Tiles* verdes: Florestas.
- *Tiles* azuis: Rios.

Tem-se na Figura 24 um exemplo de cena gerada quando o jogador se move até as montanhas, que traduzida fica: “À medida que a noite cai na montanha, as

estrelas brilham como diamantes no céu acima, lançando um brilho celestial sobre a paisagem abaixo. Os picos são banhados por uma luz suave e etérea, enquanto o ar está fresco com o aroma de pinho”.

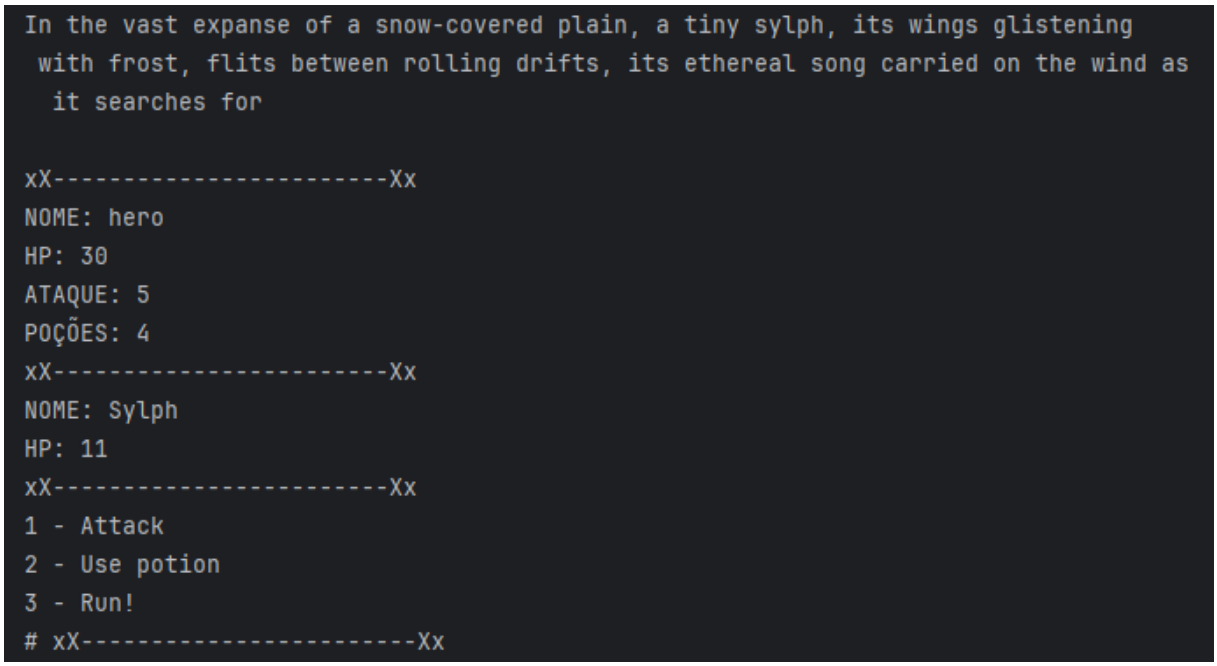
Figura 24 - Cena gerada nas montanhas



Fonte: Elaborado pelos autores (2024)

- Menu de Batalha: Na Figura 25, há um menu apenas com as informações que interessam num momento de batalha, como por exemplo os status do jogador e do inimigo, breve descrição das interações e ações disponíveis ao jogador.

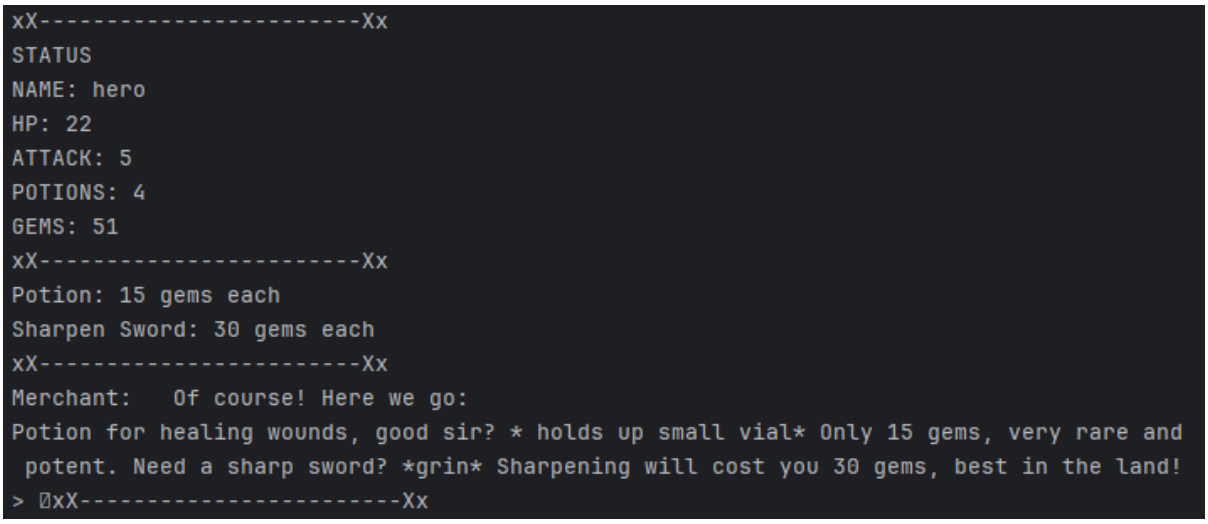
Figura 25 - Menu de Batalha



Fonte: Elaborado pelos autores (2024)

- Menu de diálogo: A seguir, na Figura 26, o menu de interação com NPCs. Nele é possível dialogar para conseguir itens

Figura 26 - Menu de Diálogo



Fonte: Elaborado pelos autores (2024).

4.4 COMO JOGAR

No menu inicial, está disponível ao jogador uma opção que o leva a um breve tutorial de como jogar o jogo, onde está incluso os comandos, legendas, e objetivos, apresentado a seguir:

4.4.1 Inputs:

- O sinal '>' espera a tecla 'Enter' para trocar de tela;
- O sinal de '#' espera um input do jogador.

4.4.2 Movimentação e itens:

- O 'P' é o jogador
- Mova-se pelo mapa utilizando as teclas '1', '2', '3' e '4'
- Use poções utilizando a tecla '5'
- Salve o estado atual e saia do jogo utilizando a tecla '0'

4.4.3 Terrenos:

- Os '.'s são terrenos de planície
- Os '8's são florestas
- Os '~'s são terrenos alagados
- Os 'A's são terrenos montanhosos

- Os 'M's são florestas de pinheiros
- A cada mudança de terrenos será gerada uma nova descrição

4.4.4 Batalha:

- Use '1' para atacar
- Use '2' para usar uma poção
- Use '3' para tentar fugir do inimigo

4.4.5 Mercador:

- Representado pelo 'M' no mapa
- É possível ter um diálogo com ele
- O momento de falar é indicado pelo 'You:' no input
- Para comprar poções digite 'potion'/'potions' e o número de poções na mesma frase. Ex: 'i want 5 potions'
- Para comprar ataque digite 'sharpen'/'sharpness' e o número de 'ataques' na mesma frase. Ex: 'i want 5 sharpness'
- Para sair digite 'bye'

4.4.6 Objetivo - O Dragão:

- Para vencer você deve matar o Dragão
- Representado pela letra 'D' no mapa
- Boa sorte

4.5 MODO DE INSTALAÇÃO

Para auxiliar na instalação do jogo, inclui-se em sua pasta um arquivo README.md com as seguintes instruções:

4.5.1 Requisitos do Sistema:

- Python 3 ou superior;
- IDE com suporte a Python.

4.5.2 Instalação e Preparação do ambiente

- Clone o repositório;
- Abra utilizando sua IDE;
- Abra o prompt da IDE na pasta do sistema e crie a pasta do Ambiente Virtual[3] com o comando: `python -m venv venv;`
- Ative o ambiente virtual com o comando:

  No Windows: `venv\Scripts\activate`

  No Linux: `source venv/bin/activate`

- Já no ambiente virtual digite o comando para instalar as bibliotecas requeridas: `pip install -r requirements.txt`

4.5.3 Download do Modelo de IA

- Disponível em: <https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGUF/blob/main/llama-2-7b-chat.Q4_K_M.gguf>
- Click em "*download*"
- Preferencialmente, coloque o modelo num repositório próximo à pasta do jogo.

4.5.4 Utilizando o Modelo

- Abra o jogo na sua IDE
- Abra a pasta "Misc"

---

3 Ambientes Virtuais isolam as dependências de cada projeto, permitindo usar diferentes versões de pacotes sem conflitos. Isso facilita a organização, manutenção e compartilhamento de projetos Python.

- Substitua o valor da variável "model_path" pelo caminho até o modelo que você baixou[4]

4.5.5 Iniciando o Jogo

- Abra a pasta do jogo utilizando um terminal, como da própria IDE, Windows Powershell ou o de sua preferência.
- Acesse o ambiente virtual (se ainda não estiver nele)
- Inicie o jogo digitando com o comando: `python main.py`
- Aguarde o modelo ser carregado nos geradores de cenas[5]

4 Coloque o ".gguf" junto. Ex: C:/modelo/llama-2-7b-chat.Q4_K_M.gguf

5 Pode demorar mais ou menos a depender da capacidade de processamento do computador.

5 VALIDAÇÃO DO SISTEMA

A validação inicial foi realizada verificando o funcionamento adequado dos menus, a geração dos cenários, a movimentação do personagem no mapa gerado proceduralmente, encontros aleatórios e batalhas com inimigos, e a interação com o mercador.

Por se tratar de um modelo local, optou-se por realizar o carregamento dos geradores de cenários e batalhas logo ao abrir o jogo, para que as chamadas do modelo IA sejam realizadas mais rapidamente.

Para validar o desempenho do modelo IA em gerar cenários, diferentes instruções e prompts foram testados, buscando o resultado mais agradável e natural possível, como apresentado na Figura 27. Por exemplo, o prompt escolhido para a geração de cenas montanhosas, traduzido, é: "cena": "Um terreno montanhoso ao amanhecer", "descrição": "À medida que a primeira luz do amanhecer surge sobre os picos escarpados, o terreno montanhoso é banhado por um caloroso brilho âmbar. Sombras dançam pelas encostas rochosas, revelando os contornos da paisagem".

Figura 27 - Prompts de geração de terreno

```
examples = [
    {
        "scene": "A raining forest in sunset",
        "description": "As the sun sets behind the thick canopy, gentle rain showers create a serene "
                       "ambiance in the forest.\n The golden hues of dusk mingle with the cool tones "
                       "of rain, painting a tranquil scene."
    },
    {
        "scene": "A mountainous terrain at sunrise",
        "description": "As the first light of dawn crests over the rugged peaks,\n "
                       "the mountainous terrain is bathed in a warm, amber glow. "
                       "Shadows dance across the rocky slopes, revealing the contours of the land."
    },
    {
        "scene": "A green plain with wind",
        "description": "Across the vast expanse of the green plain, a steady wind sweeps through the "
                       "tall grasses,\n causing them to sway in graceful undulations. "
                       "The air is crisp and invigorating, carrying the scent of wildflowers."
    },
```

Fonte: Elaborado pelos autores (2024)

Durante as validações, notou-se o constante risco da geração de texto realizada pela IA não agir como esperado, entregando na tela informações desnecessárias ou até mesmo as próprias instruções. Como solução, deixamos explícito no prompt ordens para que a IA não faça isso, a exemplo da Figura 28, que apresenta as requisições: "Os seguintes são trechos de descrições de cenários para

um RPG. Use-os como exemplos para descrever a última cena e não crie suas próprias cenas. Use os trechos fornecidos como a única base para suas descrições. Aderir estritamente à estrutura, linguagem e atmosfera dos trechos dados. Evite introduzir conceitos ou detalhes externos. Limite todas as descrições a um máximo de 50 palavras. Mantenha-as concisas e fieis ao material original. Evite usar termos como 'cena' ou linguagem equivalente. Estruture a descrição no estilo dos trechos. Mantenha a consistência com os exemplos”.

Figura 28 - Especificações para a IA

```
prefix = """
    The following are exerts of descriptions of scenarios for a rpg.
    Use them as examples to describe the last scene, and don't create
    your own scenes.
    Use the excerpts provided as the sole basis for your descriptions.
    Adhere strictly to the structure, language, and mood of the given excerpts.
    Avoid introducing external concepts or details.
    Limit all descriptions to a maximum of 50 words.
    Keep them concise and faithful to the source material.
    Avoid using terms like scene or equivalent language.
    Frame the description using the style of the excerpts.
    Maintain consistency with the examples.
"""
```

Fonte: Elaborado pelos autores (2024)

6 CONCLUSÃO

Visto que a área de jogos eletrônicos possui espaço para desenvolvimento e aprimoramento de muitas tecnologias sendo uma delas a IA, o sistema se mostrou promissor em demonstrar a eficácia do uso de grandes modelos de linguagem para a criação de jogos baseados em texto.

O desenvolvimento do jogo atendeu os objetivos propostos. A interface criada proporcionou uma experiência imersiva ao jogador, permitindo interações dinâmicas com eventos gerados pela IA que reforçam a imersão e a jogabilidade. Além disso, foi estabelecido um ciclo de gameplay sólido, no qual o jogador progride ao fortalecer seu personagem por meio de batalhas e interações com NPCs, culminando no enfrentamento de um inimigo final desafiador, o dragão.

Durante o desenvolvimento do jogo, uma das principais dificuldades encontradas foi a desobediência da IA na geração de cenas conforme o esperado. Em alguns casos, o modelo retornava informações inconsistentes, como a repetição de instruções do prompt ou a criação de descrições fora do contexto solicitado. Esse problema foi solucionado por meio da elaboração de prompts mais específicos e detalhados, com instruções explícitas para evitar respostas inadequadas. Além disso, foi necessário realizar uma série de testes iterativos para ajustar os parâmetros da geração de texto, garantindo que as cenas fossem coerentes com o ambiente e os modificadores definidos no jogo. Essa abordagem permitiu que o modelo gerasse resultados mais naturais e alinhados com a proposta imersiva.

Segue-se também a possibilidade de readaptação de conceitos e técnicas utilizadas neste trabalho por outros desenvolvedores para a criação dos mais variados jogos baseados em texto, uma vez que todos os componentes principais já estão implementados.

Por fim, diante dos resultados obtidos, é possível afirmar que existem diversas possibilidades de trabalhos futuros a serem desenvolvidos neste sistema. Alguns exemplos incluem:

- Aprimoramento de sistemas do jogo: Tratando-se de um RPG, é cabível a implementação de sistemas de equipamento e nivelamento de personagens, a fim de aprimorar a imersão e jogabilidade;
- Adição de novos componentes IA ao sistema: O sistema atual inclui componentes básicos para a geração de texto com base em LLM, mas pode ser

expandido para incluir outros componentes, como treinamento de agentes com tecnologias *Machine Learning*;

- Criação de um sistema de missões: Como é padrão no gênero, o jogo pode ser expandido com um sistema de missões, objetivos secundários que o jogador pode fazer a fim de obter recompensas;
- Adição de novos personagens: Assim como missões novas, é benéfico à um jogo do gênero a presença de mais personagens e diálogos, podendo estes mesmos entregar missões ao jogador;
- Teste do sistema em diferentes plataformas: O jogo foi desenvolvido e testado em um ambiente de computador, mas pode ser adaptado para outras plataformas, como dispositivos móveis; e
- Implementação de modo multijogador: Jogos multijogadores baseados em texto já foram populares no passado, os chamados MUDs, ou *Multi-user Dungeon*, precursores dos MMORPGs. A adição de um modo multijogador pode trazer um valor imenso ao jogo, além de interações e eventos interessantes por conta da implementação IA.

Em resumo, a área de jogos baseados em texto possui muitas possibilidades para desenvolvimentos futuros se inserida no contexto de IA, e o jogo apresentado oferece uma série destas possibilidades, além de atender um nicho do mercado com enorme potencial, mesmo sendo uma pequena amostra do que pode ser alcançado com essas tecnologias.

REFERÊNCIAS

BHALERAO, Chinmay. Theoretical understanding of chains, prompts, and other important modules in Langchain. Medium, 2023. Disponível em: https://pub.towardsai.net/understanding-langchain-%EF%B8%8F-part-1-98499559f4c4. Acesso em: 13 nov. 2024.

CLEMENT, J. Global video game market value from 2022 to 2032. Statista, 2024. Disponível em: https://www.statista.com/statistics/292056/video-game-market-value-worldwide/. Acesso em: 10 mai. 2024.

DIVERS, Gavin. Gaming Industry Dominates as the Highest-Grossing Entertainment Industry. GamerHub, 2023. Disponível em: https://gamerhub.co.uk/gaming-industry-dominates-as-the-highest-grossing-entertainment-industry/#:~:text=To%20put%20that%20in%20perspective,much%20as%20the%20movie%20industry. Acesso em: 10 mai. 2024.

HUGGING FACE. LLaMA 2 model documentation. Hugging Face, 2024. Disponível em: https://huggingface.co/docs/transformers/model_doc/llama2. Acesso em: 16 nov. 2024.

JETBRAINS. Tools and Features – PyCharm. JetBrains, 2024. Disponível em: https://www.jetbrains.com/pycharm/features/tools.html. Acesso em: 18 nov. 2024.

LANGCHAIN. GitHub, 2024. Disponível em: https://github.com/langchain-ai/langchain. Acesso em: 10 mai. 2024.

LEMOS, Diego. O que é a API do GPT-4 e seus custos? IA Básico, 2024. Disponível em: https://www.iabasico.com/posts/o-que-e-a-api-do-gpt-4-e-seus-custos. Acesso em: 16 nov. 2024.

PEREIRA, Fabio. O que é Langchain e LLM? Medium, 2024. Disponível em: https://medium.com/@fabio.alvaro/o-que-%C3%A9-langchain-e-llm-ed95880001e8. Acesso em: 13 nov. 2024.

TAY, Wendy; TRISCHLER, Adam; FINE, Emery. TextWorld: A learning environment for training reinforcement learning agents, inspired by text-based games. Microsoft Research Blog, 2018. Disponível em: https://www.microsoft.com/en-us/research/blog/textworld-a-learning-environment-for-training-reinforcement-learning-agents-inspired-by-text-based-games/. Acesso em: 10 mai. 2024.

VEJA. Mudou de fase: mercado de games já fatura mais que o de cinema. Disponível em: https://veja.abril.com.br/tecnologia/mudou-de-fase-mercado-de-games-ja-fatura-mais-que-o-de-cinema/. Acesso em: 2 dez. 2024.

VIDEOGAME WARLOCK. Game engines: conheça 5 ótimas ferramentas para o desenvolvimento de games. Disponível em: https://videogamewarlock.com/game-engines-conheca-5-otimas-ferramentas-para-o-desenvolvimento-de-games/. Acesso em: 2 dez. 2024.