🎮 Duality – RPG de Batalha

IFPE Campus – Jaboatão dos Guararapes
TADS – Estrutura de Dados
Período – 2º Noite
Equipe: João Pedro da Costa Carvalho e Marciojunior Almeida da Silva Filho

Sumário

Relatório Técnico

Descrição do Problema Resolvido

O projeto **"Duality – RPG de Batalha"** consiste em um jogo de simulação baseado em texto, no qual o jogador percorre uma árvore de decisões enfrentando inimigos em diferentes níveis de dificuldade. O jogo visa resolver o desafio de representar lógicas de tomada de decisão, filas de eventos, hierarquias de combate e controle de estados com o uso de estruturas de dados. Ele simula cenários interativos com inimigos, combates e rolagem de dados, promovendo aprendizado prático de estruturas fundamentais como **árvore binária**, **fila**, **dicionário** e **heap**.

Justificativa da Escolha do Tema

A escolha do tema foi motivada pela necessidade de tornar o estudo de estruturas de dados mais interativo, lúdico e aplicável a situações do mundo real. Jogos de RPG oferecem uma excelente oportunidade para explorar árvores de decisão, gestão de filas de ações, ordenação de inimigos por força e uso contextual de dicionários. Além disso, o projeto promove aprendizado significativo ao unir programação orientada a objetos com lógica de jogo e organização modular de código.

Descrição de cada Estrutura Utilizada e Justificativa

Estrutura de Dados	Aplicação no Projeto	Justificativa
Árvore Binária (arvore.py)	Representa o caminho do jogador entre os níveis do jogo (nível 1 ao 4), onde cada nó é uma sala com possibilidade de batalha ou fuga.	Ideal para modelar decisões em jogos, oferecendo estrutura hierárquica clara e navegação binária entre opções (esquerda/direita)
Fila (fila.py)	Controla o histórico de ações realizadas durante o jogo (como ataques e defesas).	Garante a ordem cronológica das ações, sendo uma estrutura perfeita para processar eventos em sequência.

Dicionário (dicionario.py)	Armazena atributos de cada inimigo e jogador (vida, tipo de ataque/defesa, fraquezas).	Permite acesso rápido a atributos e facilita mapeamentos entre ações e respostas no jogo.

Desafios Enfrentados e Soluções encontradas

Desafio	Solução
Implementar a árvore de decisões com combate dinâmico por níveis	Utilização de árvore binária manualmente construída para garantir caminhos fixos com inimigos distintos
Gerar aleatoriedade no surgimento de inimigos e eventos vazios	Uso de funções com rolagem de dados para controlar aleatoriedade, com lógica específica por nível
Evitar repetição de mensagens de salas vazias	Refatoração da função verificar_aparicao_inimigo() para garantir que o print ocorra apenas uma vez por tentativa
Organizar o código de forma modular e reutilizável	Separação dos arquivos em módulos específicos (game, utils, estruturas), utilizando boas práticas de POO e __init__.py
Documentar todas as funções de forma automatizada	Uso de pydoc e padronização de docstrings em todos os arquivos .py com geração de documentação .txt

Instrução para Execução do Projeto

Pré-requisitos

- Python 3.10 ou superior instalado.

Passos para execução

1. Clonar Repositório

   Github - Duality

2. Certifique-se de que a estrutura de pastas está assim:

```
duality/
├── estruturas/
|   ├── arvore.py
|   ├── dicionario.py
|   ├── fila.py
|   └── heap_sort.py
├── game/
|   ├── batalha.py
|   └── persona.py
├── utils/
|   ├── acao.py
|   └── dados.py
```

├── main.py

├── README.md

└── LICENSE

3. Execute o jogo com o seguinte comando:

   python main.py

Documentação Técnica dos Códigos

A seguir, apresenta-se a documentação automática gerada pelo pydoc, com base nas docstrings de cada módulo Python do projeto.

estruturas.arvore.py

Python Library Documentation: module estruturas.arvore in estruturas

NAME

estruturas.arvore - módulo arvore.py

DESCRIPTION

Este módulo contém funções relacionadas a criação e implementação da Árvore Binária.

CLASSES

builtins.object

ArvoreBinaria

class ArvoreBinaria(builtins.object)

| ArvoreBinaria(valor=None)

|

| Classe que representa uma árvore binária onde cada nó pode conter um inimigo.

|

| Attributes:

| inimigo (Inimigo): Inimigo presente na sala.

| esquerda (ArvoreBinaria): Caminho à esquerda.

| direita (ArvoreBinaria): Caminho à direita.

|

| Methods defined here:

|

| __init__(self, valor=None)

| Initialize self. See help(type(self)) for accurate signature.

|

```
 |  __str__(self)
 |      Return str(self).
 |
 |  inserir_direita(self, valor)
 |      Insere um nó à direita.
 |
 |  inserir_esquerda(self, valor)
 |      Insere um nó à esquerda.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables
 |
 |  __weakref__
 |      list of weak references to the object
FILE
    .\duality\estruturas\arvore.py
```

estruturas.dicionario.py

```
Python Library Documentation: module estruturas.dicionario in estruturas

NAME
    estruturas.dicionario - módulo dicionario.py

DESCRIPTION
    Módulo que implementa um wrapper para dicionários com métodos utilitários.

CLASSES
    builtins.object
        Dicionario

    class Dicionario(builtins.object)
     |  Dicionario(dados=None)
     |
     |  Classe de dicionário personalizada para ataques e defesas.
```

```
    |
    |  Methods:
    |      get(tipo: str): Retorna o valor associado ao tipo.
    |
    |  Methods defined here:
    |
    |  __init__(self, dados=None)
    |      Initialize self.  See help(type(self)) for accurate signature.
    |
    |  __str__(self)
    |      Return str(self).
    |
    |  adicionar(self, chave, valor)
    |      Adiciona um par chave-valor ao dicionário.
    |
    |  obter(self, chave)
    |      Obtém o valor associado a uma chave.
    |
    |  remover(self, chave)
    |      Remove uma chave do dicionário.
    |
    |  ----------------------------------------------------------------------
    |  Data descriptors defined here:
    |
    |  __dict__
    |      dictionary for instance variables
    |
    |  __weakref__
    |      list of weak references to the object

FILE
    .\duality\estruturas\dicionario.py
```

estruturas.fila.py

Python Library Documentation: module estruturas.fila in estruturas

NAME

estruturas.fila - módulo fila.py

DESCRIPTION

Este módulo contém funções relacionadas a criação e implementação da Fila, no qual vamos guardar todos os históricos de movimentação.

CLASSES

builtins.object

Fila

class Fila(builtins.object)

| Fila básica para armazenar eventos do jogo.

|

| Attributes:

| data (list): Lista de elementos na fila.

|

| Methods:

| enqueue(item): Adiciona item à fila.

| dequeue(): Remove e retorna o primeiro item da fila.

| is_empty(): Verifica se a fila está vazia.

|

| Methods defined here:

|

| __init__(self)

| Initialize self. See help(type(self)) for accurate signature.

|

| __str__(self)

| Return str(self).

|

| desenfileirar(self)

| Remove e retorna o item do início da fila.

|

| enfileirar(self, item)

| Adiciona um item ao final da fila.

|

| vazia(self)

| Verifica se a fila está vazia.

|

| --

| Data descriptors defined here:

|

| __dict__

| dictionary for instance variables

|

| __weakref__

| list of weak references to the object

FILE

.\duality\estruturas\fila.py

game.batalha.py

Python Library Documentation: module game.batalha in game

NAME

game.batalha - Módulo que implementa as funções de combate entre o herói e os inimigos.

DESCRIPTION

Funções:

aplicar_defesa: Aplica lógica de mitigação de dano conforme defesa escolhida.

verificar_aparicao_inimigo: Sorteia chance de aparição do inimigo.

iniciar_combate: Executa o combate baseado em turnos.

tentar_fuga: Avalia se o herói consegue escapar da batalha.

CLASSES

builtins.object

Batalha

class Batalha(builtins.object)

| Batalha(heroi: game.persona.Heroi, historico: estruturas.fila.Fila)

|

| Gerencia o sistema de combate do jogo.

|

| Methods defined here:

|

| __init__(self, heroi: game.persona.Heroi, historico: estruturas.fila.Fila)

| Initialize self. See help(type(self)) for accurate signature.

|

| iniciar_combate(self, inimigo: game.persona.Inimigo) -> bool

| Inicia e gerencia um combate completo.

|

| tentar_fuga(self, inimigo: game.persona.Inimigo) -> bool

| Tenta fugir do combate contra o inimigo. Retorna True se a fuga foi bem-sucedida (nenhum combate ocorre)

| ou se o herói venceu o combate após a fuga falhar. Retorna False se o herói foi derrotado.

|

| --

| Data descriptors defined here:

|

| __dict__

| dictionary for instance variables

|

| __weakref__

| list of weak references to the object

FUNCTIONS

aplicar_defesa(tipo_inimigo: str, defesa_escolhida: str, dano: int) -> int

Aplica a lógica de defesa, reduzindo o dano conforme a combinação de ataque e defesa.

verificar_aparicao_inimigo(inimigo: game.persona.Inimigo) -> bool

Determina aleatoriamente se um inimigo aparece, com base em uma rolagem de dado.

O Dragão (chefe final) sempre aparece.

FILE

.\duality\game\batalha.py

game.persona.py

Python Library Documentation: module game.persona in game

NAME

game.persona - Módulo que define as classes de personagens do jogo: Herói e Inimigo.

CLASSES

builtins.object

Persona

Heroi

Inimigo

class Heroi(Persona)

| Classe que representa o herói jogável.

|

| Methods:

| restaurar_vida(): Restaura HP total do herói.

|

| Method resolution order:

| Heroi

| Persona

| builtins.object

|

| Methods defined here:

|

| __init__(self)

| Initialize self. See help(type(self)) for accurate signature.

|

| restaurar_vida(self) -> None

| Restaura toda a vida do herói.

|

| ----------

| Methods inherited from Persona:

|

| __str__(self) -> str

| Return str(self).

|

| receber_dano(self, dano: int) -> bool

```
|  Aplica dano e retorna True se ainda estiver vivo.
|
| ----------------------------------------------------------------------
| Data descriptors inherited from Persona:
|
| __dict__
|  dictionary for instance variables
|
| __weakref__
|  list of weak references to the object


class Inimigo(Persona)
| Inimigo(nome: str, hp: int, tipo_atk: str, fraqueza: str)
|
| Classe para representar inimigos do jogo.
|
| Attributes:
|  tipo_atk (str): Tipo de ataque principal do inimigo.
|  fraqueza (str): Tipo de ataque ao qual é vulnerável.
|
| Method resolution order:
|  Inimigo
|  Persona
|  builtins.object
|
| Methods defined here:
|
| __init__(self, nome: str, hp: int, tipo_atk: str, fraqueza: str)
|  Initialize self.  See help(type(self)) for accurate signature.
|
| ----------------------------------------------------------------------
| Methods inherited from Persona:
|
```

```
|  __str__(self) -> str
|      Return str(self).
|
|  receber_dano(self, dano: int) -> bool
|      Aplica dano e retorna True se ainda estiver vivo.
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from Persona:
|
|  __dict__
|      dictionary for instance variables
|
|  __weakref__
|      list of weak references to the object

class Persona(builtins.object)
|  Persona(nome: str, hp: int, ataque: dict, defesa: dict)
|
|  Classe base para personagens.
|
|  Attributes:
|      nome (str): Nome do personagem.
|      hp (int): Pontos de vida.
|      ataque (dict): Tipos de ataque e seus valores.
|      defesa (dict): Tipos de defesa e seus valores.
|
|  Methods:
|      receber_dano(dano): Aplica dano e retorna se ainda está vivo.
|
|  Methods defined here:
|
|  __init__(self, nome: str, hp: int, ataque: dict, defesa: dict)
|      Initialize self.  See help(type(self)) for accurate signature.
```

```
    |
    |  __str__(self) -> str
    |      Return str(self).
    |
    |  receber_dano(self, dano: int) -> bool
    |      Aplica dano e retorna True se ainda estiver vivo.
    |
    |  ----------------------------------------------------------------------
    |  Data descriptors defined here:
    |
    |  __dict__
    |      dictionary for instance variables
    |
    |  __weakref__
    |      list of weak references to the object

FILE
    .\duality\game\persona.py
```

utils.acao.py

```
Python Library Documentation: module utils.acao in utils

NAME
    utils.acao - Módulo que define ações que o jogador pode tomar em combate.

DESCRIPTION
    Funções:
        escolher_acao(): Solicita ao jogador a ação desejada (atacar ou defender).
        escolher_tipo_ataque(): Retorna o tipo de ataque escolhido.
        escolher_defesa(): Retorna o tipo de defesa escolhido.
        rolar_dado(): Simula um dado D20.

FUNCTIONS
    escolher_acao()
        Exibe as opções de ação de combate e retorna a escolha do jogador.


    escolher_defesa()
```

Exibe os tipos de defesa disponíveis e retorna a defesa escolhida.

escolher_tipo_ataque()

Exibe os tipos de ataque disponíveis e retorna o tipo escolhido.

rolar_dado()

Gera um valor aleatório de 1 a 20, simulando um dado D20.

FILE

.\duality\utils\acao.py

main.py

Python Library Documentation: module main

NAME

main - Arquivo principal do jogo Duality - RPG de Batalha.

DESCRIPTION

Este módulo inicializa o jogo, constrói a árvore de decisões,

e gerencia o loop principal de navegação pelas salas.

CLASSES

builtins.object

Duality

class Duality(builtins.object)

| Classe principal que gerencia o jogo.

|

| Methods defined here:

|

| __init__(self)

| Initialize self. See help(type(self)) for accurate signature.

|

| jogar(self)

| Inicia o jogo, gerenciando o loop de exploração e combate.

|

| ----------

| Data descriptors defined here:

```
    |
    |  __dict__
    |      dictionary for instance variables
    |
    |  __weakref__
    |      list of weak references to the object

FILE
    .\duality\main.py
```