Maxwell, O viajante

Milena Ribeiro, Max Ramon, Marcos Vinnicius
ICEV - Instituto de Ensino Superior, Engenharia de Software,Teresina-PI
R. Dr. José Auto de Abreu, 2929 - São Cristóvão
Teresina - PI, 64055-260
27/06/2023

Resumo – trata-se da nossa solução para um projeto proposto em sala pelo o Prof. Dimmy que ministra a matéria estrutura de dados. Nosso desafio nesse trabalho era criar um jogo de simulação de logística baseado nos conceitos apresentados em sala. Nomeamos o projeto com "Maxwell,O viajante".

I INTRODUÇÃO

Nos últimos anos, os conceitos de estrutura de dados têm sido amplamente aplicados em diferentes áreas, incluindo jogos de simulação. Neste projeto, propomos a criação de um jogo de simulação de logística baseado em estruturas de dados, com o objetivo de oferecer aos jogadores a oportunidade de gerenciar uma rede de transporte de recursos.

O jogo tem como objetivo principal desafiar os jogadores a avaliar conexões, comprar e vender recursos, e competir com outros jogadores para maximizar o fluxo de recursos e minimizar os custos de transporte. Para implementar esse jogo, usamos três estruturas de dados que possibilitem a representação eficiente da rede de transporte de recursos e a tomada de decisões estratégicas.

Usamos grafo para representar as cidades, onde as arestas representam as fronteiras entre os pontos. Cada aresta terá um peso associado, representando o custo de transporte dos recursos através dela. Essa estrutura permitirá aos jogadores visualizarem e planejarem suas rotas de maneira eficiente.

Além disso, implementamos algoritmos de busca e fluxo de rede para auxiliar os jogadores a encontrar o caminho mais eficiente para transportar os recursos entre os pontos da rede. Esses algoritmos serão fundamentais para a tomada de decisões estratégicas durante o jogo, garantindo o melhor uso dos recursos disponíveis e maximizando o desempenho logístico.

Durante o jogo, os jogadores serão confrontados com diferentes eventos e desafios, nos quais serão solicitadas decisões estratégicas. Por exemplo, quando o personagem Maxwell chegar à cidade do Kingdom of Kalb, o jogo solicitará ao jogador que tome decisões importantes, como escolher a rota mais eficiente para transportar os recursos necessários.

Ao longo deste documento, apresentaremos a metodologia e as estratégias utilizadas na implementação do jogo, destacando a importância das estruturas de dados e dos algoritmos de busca e fluxo de rede. Também discutiremos os resultados obtidos e as implicações desse projeto no contexto da simulação de logística e do aprendizado de estruturas de dados.

II MECÂNICA DO JOGO

O jogo consiste em uma jornada em que o jogador controla o personagem Maxwell em um mundo onde ele deve gerenciar o poder de uma joia enquanto viaja entre cidades vizinha.O objetivo é maximizar o poder da joia e completar a jornada de Maxwell.

O poder da joia é crucial para a sobrevivência de Maxwell, e algumas regras determinam seu ganho e perda de poder. Se o poder da jóia ficar abaixo de zero é instantaneamente definido como zero. Por outro lado, se o poder da joia exceder sete, Maxwell morre. O limiar de poder da joia pode ser aumentado ou diminuído por meio de missões.

Para viajar entre as cidades, Maxwell deve obedecer a certas regras. Ele só pode se deslocar para uma cidade vizinha àquela em que está atualmente. Além disso, para se deslocar, ele precisa ter moedas de transporte.

Maxwell inicia sua jornada com três moedas de transporte. Cada vez que ele viaja para uma cidade, ele consome uma moeda. Se as moedas acabarem, Maxwell morre, portanto, é importante gerenciá-las adequadamente. Para viajar para uma cidade, ele deve ter pelo menos uma moeda de transporte em sua posse.

Ao longo da jornada, existem várias cidades com missões disponíveis. Essas missões podem otimizar a jornada de Maxwell, melhorando o poder da joia, fornecendo moedas de transportes adicionais ou oferecendo outros benefícios estratégicos.

O jogo desafia o jogador a tomar decisões estratégicas ao equilibrar o poder da joia, gerenciar as moedas de transporte e escolher as melhores rotas entre as cidades vizinhas. A conclusão das missões é fundamental para otimizar a jornada e maximizar as chances de sucesso de Maxwell.

Cidade	Quantidade de Poder
Ubud	0
Kingdom of Legmod	2
Principality of Nekikh	1
Principality of Gritesthr	5
Protectorate of Dogrove	3
Kingdom of Lastwatch	-2
Grand Duchy of Smalia	1
Kingdom of Oldcalia	4
Kingdom of Kalb	2
Aymar League	1
Defalsia	-3
Vunese of Empire	0
Principality of Karhora	-2
Chandir Sultanate	1
Bun	5
Principality of Kasya	-7

III Implementação

Nesta seção, serão abordados os aspectos-chave da implementação do jogo de simulação logística, considerando o uso da linguagem Java e as estruturas de dados grafos, encapsulamento e polimorfismo.

III.I Estruturas de Dados e Grafos

Para representar a rede de transporte de recursos e as conexões entre as cidades, foi utilizada a estrutura de dados de grafos. Cada cidade é representada como um nó no grafo, enquanto as arestas representam as fronteiras entre as cidades, com pesos associados a elas para representar os custos de transporte.A implementação do grafo foi realizada através de classes e objetos em Java. Cada nó do grafo foi encapsulado em uma classe chamada “Cidade”, que armazena informações como o nome da cidade (string Nome), (List<Aresta> ligacoes) que representa o custo de transporte entre as cidades.
A classe “Main” foi criada para gerenciar o grafo como um todo. Ela contém métodos para adicionar cidades e arestas,e realiza outras operações relacionadas ao grafo.

III.II Encapsulamento

O encapsulamento foi aplicado para garantir a integridade dos dados e controlar o acesso a eles.As classes foram projetadas com variáveis de instância privadas, públicas e default para manipular essas variáveis.Por exemplo, a classe “GerenciadorLPM”
Possui métodos para acessar e modificar os recursos disponíveis, e por serem públicas, podem ser acessadas diretamente fora da classe.

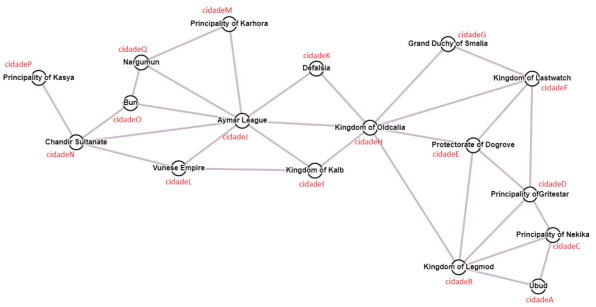
Além disso, os métodos foram desenvolvidos para garantir que as operações sobre os objetos sejam consistentes e seguras. Por exemplo, ao comprar ou vender recursos em uma cidade, são verificadas as condições necessárias, como a disponibilidade de recursos suficientes e a capacidade de transporte.

III.III Polimorfismo

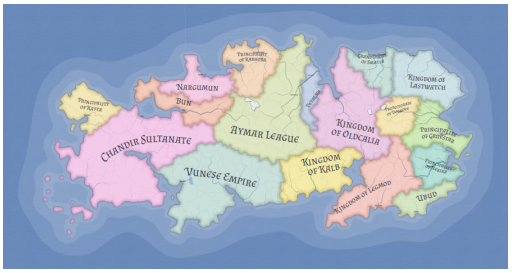
O polimorfismo foi usado para criar diferentes tipos de cidades e recursos no jogo, permitindo a extensibilidade e flexibilidade do sistema. Foram criadas construtores que permitem utilizar métodos de uma classe em outras. Utilizamos o construtor “GerenciadorLPM ger = new GerenciadorLPM();” que foi utilizado na classe “Game” para modificar e visualizar o valor da Moeda, Limiar e Poder da Jóia. Usando os métodos getters e setters de cada atributo.

IV METODOLOGIA

A classe Main foi pensada para que adicione o custo de deslocamento de uma cidade para outra. Cada cidade foi referenciada com o nome de cidadeA, cidadeB, cidadeC, etc… chamando a classe Cidade como construtora dentro da classe Main. Assim, foi possível referenciar cada vértice e aresta do grafo.



Para a definição de vizinho de cada cidade, foi utilizado o mapa fornecido para a atividade, as cidades com fronteiras são vizinhas e assim possuem arestas entre elas.



A classe Cidade é uma implementação simplificada de um grafo direcionado, onde cada nó representa uma cidade e as arestas representam as ligações entre as cidades. Foram utilizadas nessa classe a fim de armazenar as ligações (arestas) entre as cidades.:

```
import java.util.ArrayList;
```

```
import java.util.List;
```

V TESTE E VALIDAÇÃO

Durante o desenvolvimento do jogo, foram utilizadas diversas estratégias de teste para verificar a funcionalidade e a qualidade do jogo. A seguir, são satisfeitas as principais estratégias de teste adotadas:

1 - Testes de Unidade: Foram realizados testes de unidade para verificar o funcionamento correto de componentes e funcionalidades específicas do jogo. Cada unidade de código foi testada para garantir que suas funcionalidades estavam de acordo com o esperado.

* Teste de unidade para verificação de mecânicas de jogo específicas: Dependendo das mecânicas únicas do jogo, é importante realizar testes de unidade para verificar se essas mecânicas estão funcionando corretamente.(Esse exemplo foi utilizado principalmente na limiar do poder,para saber se suas funções estavam funcionando corretamente).

2- Testes de Integração: Após a conclusão dos testes de unidade, foram realizados testes de integração para verificar se os diferentes componentes do jogo funcionam corretamente em conjunto. Esses testes visavam identificar possíveis problemas de comunicação e integração entre os diferentes módulos do jogo(Teste como foco a interação do personagem Maxwell com o restante dos componentes do jogo).

3- Testes Funcionais: Foram realizados testes funcionais para verificar se todas as funcionalidades do jogo estavam operando conforme o esperado. Cada funcionalidade foi testada para garantir que ela funcionava corretamente e atendeu aos requisitos do jogo

4- Testes de Usabilidade: Foram cuidadosos testes de usabilidade com usuários reais para avaliar a experiência do usuário e identificar possíveis problemas de usabilidade no jogo. O feedback dos usuários foi coletado

para realizar melhorias e aprimorar a interface do usuário, controles e fluxo de jogo(Parte na qual observamos possíveis melhorias a serem feitas em relação a otimização do código).

Resultados dos testes: Durante os testes, foram identificados alguns problemas e erros que precisaram ser corrigidos.

o principal erro encontrado foi:

1- (loop do jogo infinito),o jogo n,ou seja revelou um bug ou uma falha na lógica do jogo.

V DESCRIÇÃO DE EQUIPE

Guilherme Vinnicius: criou as classes Main e Aresta. Na classe Main adicionou as arestas pertencentes ao grafo e chama um método de execução da classe Game e a classe Aresta que é responsável por armazenar informações sobre as ligações entre as cidades no grafo, como a cidade de destino e o custo associado.

Max Ramon: criou as classes Cidade e Mercador. A classe Cidade fornece uma estrutura básica para modelar um grafo direcionado de cidades interconectadas e suas ligações. A classe Mercador define o rumo da quantidade de moedas e limiar que Maxwell possui após a conversa com o npc.

Luiz Antonio: criou a classe Missao que possui métodos que fazem parte do sistema que verifica as missões disponíveis em diferentes cidades e exibe mensagens relacionadas a elas.

Marcos Vinnicius: criou a classe GerenciadorLPM que gerencia os atributos privados limiar, moeda e poder da joia, para que possam ser atribuídos novos valores e/ou utilizados em outras classes, como a classe Game.

Milena Duarte: criou a classe Game que une todas as classes (menos a Main) utilizando Construtores que chamam as outras classes. Nessa classe é feita a interação de Maxwell com o NPC, verificação de cidades vizinhas e apresentação de menus de viagem, abandono de missão e verificação de situação das missões.

VI RESULTADOS

Resultados obtidos:

- O terminal exibe o menu inicial que pergunta para onde o personagem deseja viajar, onde só é possível se Maxwell possuir as moedas necessárias para o deslocamento desde que sobre pelo menos uma moeda, caso seu saldo fique negativo ou zerado, Maxwell morre.

- Em seguida é verificado se Maxwell possui missão em andamento e é questionado se deseja abandoná-la. Podendo ser abandonada sempre ao chegar em uma cidade.

- Logo após vem a verificação de cidade, se Maxwell estiver em uma das três cidades que possuem missão, é perguntado se ele deseja participar, se sim sempre que o loop recomeçar será verificado se está na cidade de destino da missão e se deseja abandonar a missão.

- É mostrado também antes de viajar, o diálogo entre ele e o mercador, que pode gastar ou somar no valor de suas moedas, e enfim o loop recomeça.

VII CONCLUSÃO

A atividade nos deu a oportunidade de explorar os conhecimentos adquiridos em sala na matéria de Estrutura de Dados. Utilizar os conceitos de Polimorfismo, Grafo e Encapsulamento com uma atividade interessante é fundamental para o aprendizado, pensar na metodologia e o raciocínio que as classes teriam para que o jogo funcionasse foi bastante desafiador, apresenta alguns erros, porém de longe a melhor oportunidade de exercitar os conceitos vistos.

VIII REFERÊNCIAS

[1] **Canal do YouTube**: RyiSnow:How to Make a 2D Game inJava: [https://www.youtube.com/watch?v=om59cwR7p-_sI&abchannel=RyiSnow] (acessado em 25 de junho de 2023)

[2] **IEEE Advancing Technology for Humanity**

[3] https://chat.openai.com/ em 22/02/2023

[4] **Java completo** 2023 Programação Orientada a Objeto + projetos [https://www.udemy.com/course/java-curso-completo/learn/lecture/12006452?start=1#overview]