UNIVERSIDADE FEDERAL DE MATO GROSSO

Universidade Federal de Mato Grosso
ICET - Instituto de Ciências Exatas e da Terra
Departamento de Ciência da Computação
Graduação em Ciência da Computação

Raissa Cavalcanti

Trabalho Prático - Sistema de Batalhas Pokémon

Raissa Cavalcanti

Barra do Garças
08 de Novembro de 2024

Raissa Cavalcanti

Trabalho Prático - Sistema de Batalhas Pokémon

Relatório referente ao Projeto de Desenvolvimento de Simulação de Pokémon.

Orientador: Professor Sandino

Barra do Garças
08 de Novembro de 2024

Introdução

A franquia Pokémon, criada em 1995, é um fenômeno global que começou com jogos lançados para o Game Boy original. Com foco em criaturas fictícias chamadas Pokémons, a série se destaca por sua mecânica de captura, treinamento e batalhas entre essas criaturas, onde cada Pokémon possui um ou mais tipos, como Fogo, Água e Planta. Esses tipos não apenas definem as características dos Pokémons, mas também influenciam diretamente a eficácia dos ataques em batalhas, criando uma dinâmica estratégica que desafia os jogadores a escolherem sabiamente suas criaturas e movimentos.

Este trabalho prático, desenvolvido no curso de Bacharelado em Ciência da Computação da Universidade Federal de Mato Grosso, tem como objetivo implementar uma simulação de batalha entre Pokémons utilizando a linguagem C++, explorando a mecânica dos jogos da série e proporcionando uma experiência interativa ao usuário.

Detalhamento

Etapa 1: Carregamento de Dados

A primeira etapa do programa consiste na leitura e processamento dos arquivos pokemons.txt e ataques.txt, os quais contêm informações sobre os Pokémons e seus ataques. O programa carrega uma lista de Pokémons e realizar o sorteio de três Pokémons para o jogador e três para a CPU. Além disso, o sistema atribui quatro ataques a cada Pokémon selecionado, respeitando as regras que permitem que ataques do tipo Normal sejam usados por qualquer Pokémon, enquanto outros tipos devem corresponder aos tipos dos Pokémons em questão.

Etapa 2: Configuração do Menu de Jogo

O menu de jogo oferece opções interativas para o jogador, incluindo o início de uma batalha, seleção da dificuldade do jogo (Fácil, Médio ou Difícil) e visualização do ranking dos jogadores. Ao iniciar uma batalha, o jogador pode escolher um dos Pokémons disponíveis e ajustar as configurações de dificuldade, que impactarão os níveis dos Pokémons da CPU e a estratégia de ataque utilizada.

Detalhamento

Etapa 3: Lógica de Batalha

A lógica de batalha implementa turnos alternados entre o jogador e a CPU, onde cada um pode escolher entre atacar ou trocar de Pokémon. Após cada ataque, o programa exibirá o HP atualizado dos Pokémons em batalha, e o jogo continuará até que todos os Pokémons de um dos lados sejam derrotados. O sistema de batalha é intuitivo, permitindo que o jogador tenha uma experiência similar à dos jogos originais.

Etapa 4: Cálculo do Dano

A parte crucial da simulação é o cálculo de dano, que é realizado com base em uma fórmula que considera múltiplos fatores, como o nível do Pokémon, o poder do ataque, e a efetividade do tipo. Além disso, o programa inclui uma verificação da precisão do ataque, exibindo mensagens apropriadas em caso de falha ou de ataques críticos.

Processo

O desenvolvimento do programa segue uma estrutura organizada. Primeiramente, os dados são carregados dos arquivos pokemons.txt e ataques.txt, preparando o cenário para a batalha. Em seguida, o menu interativo é implementado, garantindo que os jogadores possam facilmente navegar pelas opções disponíveis. A lógica de batalha foi codificada de forma a permitir uma troca dinâmica entre os turnos do jogador e da CPU, com as decisões de ataque e troca sendo processadas conforme as regras estabelecidas.

A implementação do cálculo de dano foi realizada em etapas, onde todos os fatores mencionados serão considerados. O código foi estruturado em múltiplos arquivos, com Main.cpp como o ponto de entrada do programa, e classes específicas para a gestão de Pokémons, ataques, e a lógica do jogo.

Resumo dos Arquivos utils.h e utils.cpp

O módulo utils contém funções essenciais para a manipulação e cálculo de interações entre Pokémons e seus ataques em um jogo baseado no universo Pokémon.

utils.h

O arquivo de cabeçalho utils.h contém as declarações de funções e tipos que são utilizados para manipular objetos relacionados aos Pokémon e seus ataques.

Funções Definidas

1. carregarPokemons:
   - Propósito: Carregar uma lista de Pokémon.
   - Parâmetros:
     - pokemons: Um vetor de objetos Pokemon que será preenchido com a lista de Pokémon.
     - quantidade: O número de Pokémon a serem carregados (com valor padrão de 3). Esse parâmetro permite carregar uma quantidade variável de Pokémon dependendo da necessidade.
2. carregarAtaques:
   - Propósito: Carregar os ataques disponíveis para um Pokémon.
   - Parâmetros:
     - ataques: Um vetor de objetos Ataque, que será preenchido com os ataques do Pokémon.
     - pokemon: O Pokémon para o qual os ataques devem ser carregados.
3. calcularDano:
   - Propósito: Calcular o dano de um ataque em uma batalha entre dois Pokémon.
   - Parâmetros:
     - atacante: O Pokémon que realiza o ataque.
     - ataque: O ataque a ser usado.
     - defensor: O Pokémon que recebe o ataque.
   - Retorno: Retorna um valor float que representa o dano calculado.

4. ataqueBemSucedido:
   - Propósito: Determinar se um ataque foi bem-sucedido.
   - Parâmetros:
     - ataque: O ataque a ser verificado.
   - Retorno: Retorna um valor booleano, onde true indica que o ataque foi bem-sucedido e false indica que falhou.

Resumo dos Arquivos utils.h e utils.cpp

utils.cpp

Onde são carregados dados de Pokémon e ataques a partir de arquivos externos, e o dano de ataques é calculado com base em várias variáveis. O sistema leva em consideração a eficácia dos tipos de Pokémon e a chance de sucesso de um ataque. As principais funções e processos do algoritmo são:

1. **Carregar Pokémon:**
   - Lê dados de Pokémon a partir de um arquivo CSV (pokemons.txt).
   - Atribui atributos como nome, tipos e estatísticas (HP, ataque, defesa, etc.).
   - Os Pokémon são embaralhados e selecionados aleatoriamente para a batalha.
2. **Carregar Ataques:**
   - Lê ataques de um arquivo CSV (ataques.txt), selecionando ataques compatíveis com os tipos do Pokémon.
   - Até 4 ataques são atribuídos a cada Pokémon.
3. **Cálculo de Dano:**
   - O dano de um ataque é calculado considerando o nível do atacante, o poder do ataque, as estatísticas do defensor, e fatores como:
     - STAB (Same Type Attack Bonus): Se o tipo do ataque corresponde ao tipo do Pokémon atacante, o dano é amplificado.
     - Efetividade dos Tipos: O tipo de ataque pode ser mais forte ou mais fraco dependendo do tipo do defensor.
     - Golpe Crítico: Existe uma chance de 1/16 de o ataque ser um golpe crítico, dobrando o dano.
     - Aleatoriedade: O dano final é ajustado por um fator aleatório.
4. **Verificar Sucesso de Ataque:**
   - A função verifica a precisão do ataque e calcula a probabilidade de falha, determinando se o ataque é bem-sucedido ou não.
5. **Estrutura de Tipos de Pokémon:**
   - O algoritmo usa um mapa de efetividade de tipos para definir a força de um ataque contra diferentes tipos de Pokémon, como "Fogo" sendo forte contra "Planta", mas fraco contra "Água".

Resumo dos Arquivos pokemon.h e pokemon.cpp

O arquivo de cabeçalho **pokemon.h** define duas classes: Ataque e Pokemon, que são usadas para representar Pokémon e os ataques que eles podem usar durante uma batalha.

1. Classe Ataque
A classe Ataque representa um ataque que pode ser usado por um Pokémon. Ela contém os seguintes atributos:

- nome: O nome do ataque (ex: "Ataque Rápido").
- categoria: A categoria do ataque, que pode ser "Fisico" ou "Especial".
- poder: O poder do ataque, que determina sua força base.
- precisao: A precisão do ataque, ou seja, a probabilidade de ele acertar.
- tipo: O tipo do ataque (ex: "Fogo", "Água", etc.), que pode influenciar sua eficácia.

Construtor:

- O construtor inicializa os atributos da classe a partir dos valores passados como parâmetros.

2. Classe Pokemon
A classe Pokemon representa um Pokémon e contém atributos que definem suas características e os ataques que ele pode usar:

- nome: O nome do Pokémon (ex: "Pikachu").
- tipo1 e tipo2: Os tipos do Pokémon (ex: "Elétrico", "Fogo"). Um Pokémon pode ter até dois tipos.
- hp: A quantidade de pontos de vida (HP) do Pokémon.
- nivel: O nível do Pokémon, que afeta seus atributos e o dano que ele causa.
- ataque: O valor do ataque físico do Pokémon.
- defesa: O valor da defesa física do Pokémon.
- velocidade: A velocidade do Pokémon, que pode influenciar a ordem dos turnos na batalha.
- ataqueEspecial: O valor do ataque especial do Pokémon.
- defesaEspecial: O valor da defesa especial do Pokémon.
- ataques: Um vetor de objetos Ataque, que armazena os ataques que o Pokémon pode usar.

Construtor:

- O construtor inicializa os atributos do Pokémon com os valores passados como parâmetros.

Método adicionarAtaque:

- Este método permite adicionar um ataque ao vetor ataques do Pokémon.

Propósito e Funcionalidade

- A classe Ataque é responsável por armazenar as informações de cada ataque disponível.
- A classe Pokemon é responsável por armazenar as informações do Pokémon e seus ataques, além de permitir a adição de novos ataques.

Resumo dos Arquivos pokemon.h e pokemon.cpp

pokemon.cpp

1. Construtor da Classe Ataque

- Objetivo: O construtor da classe Ataque inicializa um objeto de ataque com os seguintes atributos:
  - nome: O nome do ataque (ex: "Chute", "Raio Solar").
  - categoria: Define a categoria do ataque, que pode ser "Fisico" ou "Especial".
  - poder: A força do ataque, que é usada para calcular o dano causado.
  - precisao: A probabilidade de o ataque acertar o alvo.
  - tipo: O tipo do ataque (ex: "Fogo", "Água"), que influencia sua eficácia contra diferentes tipos de Pokémon.
- O inicializador de lista (parte após os :) é utilizado para inicializar os membros da classe com os valores passados como parâmetros.

2. Construtor da Classe Pokemon

- Objetivo: O construtor da classe Pokemon inicializa um Pokémon com os seguintes atributos:
  - nome: O nome do Pokémon (ex: "Pikachu").
  - tipo1 e tipo2: Os tipos do Pokémon (ex: "Elétrico", "Fogo"). Um Pokémon pode ter até dois tipos.
  - hp: Pontos de vida do Pokémon.
  - nivel: O nível do Pokémon, que influencia as estatísticas de combate.
  - ataque e defesa: Os valores de ataque e defesa física do Pokémon.
  - velocidade: A velocidade do Pokémon, que pode influenciar a ordem dos ataques na batalha.
  - ataqueEspecial e defesaEspecial: Os valores de ataque e defesa especial do Pokémon.
- Assim como no construtor da classe Ataque, o inicializador de lista é usado para atribuir os valores aos membros da classe.

3. Método adicionarAtaque

- Objetivo: Este método adiciona um ataque ao vetor de ataques (ataques) de um Pokémon. O vetor ataquesarmazena todos os ataques que o Pokémon pode usar durante uma batalha.
- O parâmetro ataque é passado por referência constante (const Ataque&), o que garante que o ataque não será modificado dentro do método.
- O método usa o método push_back da classe std::vector para adicionar o objeto ataque ao final do vetor ataques.

Resumo dos Arquivos jogo.h e jogo.cpp

jogo.h

1. Estrutura Jogador

- Objetivo: Armazenar informações de um jogador, incluindo pontuação, vitórias e derrotas.

2. Classe Usuario

- Objetivo: Representar um jogador dentro do jogo, armazenando dados como nickname, pontuação, vitórias e derrotas.
- Métodos:
- Métodos de acesso (getters e setters) para pontuação, vitórias e derrotas.
- Métodos para incrementar vitórias e derrotas.
- Salvar e carregar o usuário de/para um arquivo (salvar e carregar).
- Carregamento automático do nickname em letras minúsculas.

3. Classe Jogo

- Objetivo: Gerenciar o ciclo do jogo, interações entre o jogador e a CPU, e o ranking de jogadores.
- Atributos:
  - nickname: Nome do jogador.
  - usuarioJogador: Objeto da classe Usuario que contém os dados do jogador.
  - ranking: Mapa de jogadores e suas respectivas pontuações e estatísticas.
  - jogadorPokemons e cpuPokemons: Vetores contendo os Pokémon do jogador e da CPU, respectivamente.
  - dificuldade: Nível de dificuldade do jogo, com valores definidos pelo enum Dificuldade.
  - Flags:
    - jogadorVenceuUltimaBatalha: Indicador se o jogador venceu a última batalha.
    - batalhaEmAndamento: Flag que indica se uma batalha está em andamento.
- Métodos:
  - Iniciar e controlar batalhas: Métodos para iniciar a batalha, executar turnos e ataques, escolher Pokémon e determinar a dificuldade.
  - Gerenciar ranking: Métodos para atualizar o ranking, salvar e carregar o ranking de um arquivo.
  - Salvar pontuação e usuário: Métodos para salvar a pontuação e os dados do usuário, além de carregar ou criar um novo usuário.
  - Interações com a interface de usuário: Métodos para exibir o menu, a batalha e o ranking.

Resumo dos Arquivos jogo.h e jogo.cpp

jogo.cpp

Este código implementa a lógica central do jogo de batalha Pokémon, permitindo ao jogador interagir com o sistema, batalhar contra a CPU, e gerenciar o ranking de jogadores. O jogo oferece um sistema de batalhas com Pokémon, onde o jogador pode escolher Pokémon, atacar, trocar Pokémon e gerenciar sua pontuação, vitórias e derrotas.

1. Função clamp()

- Objetivo: Limitar o valor a um intervalo definido entre min e max. Isso é usado para garantir que a escolha do jogador esteja dentro dos limites válidos.

2. Construtor da classe Jogo

Objetivo: Inicializa um novo jogo com o nickname do jogador, carrega o ranking de jogadores, carrega os Pokémon para o jogador e para a CPU, e também carrega os ataques desses Pokémon.

3. Menu principal (exibirMenu())

Objetivo: Exibe o menu principal onde o jogador pode escolher entre iniciar uma batalha, selecionar a dificuldade, exibir o ranking ou sair do jogo.

4. Seleção de Dificuldade (selecionarDificuldade())

Objetivo: Permite ao jogador escolher o nível de dificuldade para a batalha, que pode influenciar a dificuldade da CPU.

5. Iniciar Batalha (iniciarBatalha())

- Objetivo: Inicia a batalha. O jogador escolhe um Pokémon inicial e a batalha segue até que um dos lados vença. Durante a batalha, o jogador e a CPU se alternam realizando ações (ataques ou troca de Pokémon).
- Pontuação: Se o jogador vencer, ganha pontos dependendo da dificuldade. A pontuação é atualizada no ranking.

6. Executar Turno do Jogador (executarTurnoJogador())

- Objetivo: Executa a lógica do turno do jogador, permitindo escolher entre atacar ou trocar de Pokémon. Se o Pokémon do jogador for derrotado, o próximo é escolhido (se houver).

7. Executar Turno da CPU (executarTurnoCPU())

Objetivo: Executa a lógica do turno da CPU, onde a CPU escolhe um ataque e o aplica ao Pokémon do jogador.

Resumo dos Arquivos jogo.h e jogo.cpp

jogo.cpp

8. Atualizar Ranking (atualizarRanking())

- Objetivo: Atualiza o ranking dos jogadores após cada batalha, adicionando vitórias e derrotas, e reordenando o ranking com base na pontuação.

9. Salvar e Carregar o Ranking

- Salvar Ranking (salvarRanking()): Salva os dados de pontuação, vitórias e derrotas de todos os jogadores no arquivo.
- Carregar Ranking (carregarRanking()): Carrega os dados do ranking a partir de um arquivo para o jogo.

Resumo dos Arquivo main.cpp

main.cpp

1. Entrada do Nickname

- Objetivo: Solicita ao jogador que forneça um nome (nickname) para ser usado durante a partida. O std::cin lê o nome digitado pelo jogador e o armazena na variável nickname.

2. Criação do Objeto Jogo

- Objetivo: Cria um objeto da classe Jogo, passando o nickname do jogador como argumento para o construtor da classe Jogo. O construtor da classe Jogo irá inicializar o jogo, carregar o ranking, os Pokémon e ataques, e preparar o estado inicial para o jogador.

3. Chamada ao Menu

- Objetivo: Chama a função exibirMenu() da classe Jogo, que exibe o menu de opções para o jogador (iniciar batalha, escolher dificuldade, exibir ranking ou sair).

4. Retorno do main()

Objetivo: Indica que o programa foi executado com sucesso, retornando o valor 0 para o sistema operacional. Isso é comum em programas que retornam um status de execução para o sistema.

Considerações Finais

Considerações Finais

O desenvolvimento do jogo de Pokémon apresentado neste projeto ilustra uma implementação básica e funcional de um sistema de batalha entre Pokémon. A estrutura do código, organizada em classes e arquivos, facilita a manutenção e a expansão do jogo. As principais classes, como Pokemon, Ataque, Usuario, e Jogo, trabalham em conjunto para proporcionar uma experiência de jogo interativa.

O uso de um Makefile simplifica o processo de compilação, permitindo que o programador compile e limpe o projeto com comandos simples no terminal. Isso é particularmente útil em projetos maiores, onde a gestão de múltiplos arquivos-fonte e suas dependências pode se tornar complexa.

Conclusão

Este projeto de jogo de Pokémon é um excelente exemplo de como conceitos fundamentais de programação orientada a objetos podem ser aplicados para criar um aplicativo interativo e divertido. A interação entre as classes, a lógica de batalha, e a persistência de dados com um sistema de ranking demonstram habilidades práticas em C++.