UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Programação e Desenvolvimento de Software II

Projeto final - Batalha RPG

Gabriela Pereira Faria Paixão
Isadora Alves de Salles
João Marcos Machado Couto
Roberto Gomes Rosmaninho Neto

Belo Horizonte
29 de novembro de 2018

1 Introdução

Este trabalho consiste em uma implementação de batalhas RPG inspiradas na narrativa de Harry Potter, série de fantasia escrita pela autora britânica J. K. Rowling. O projeto tem como objetivo implementar os conceitos de Programação Orientada a Objetos e Boas Práticas de programação, aprendidos durante a disciplina, de uma forma criativa e criando uma interação com o usuário.

2 Implementação

O programa foi desenvolvido utilizando orientação a objetos por meio da linguagem de programação C++.

2.1 Classes

Classe *Character*: representa um personagem qualquer no contexto do jogo. A partir dela foram criadas duas classes mais específicas, *Wizard* e *Enemy* - representando, respectivamente, os jogadores(bruxos) e os inimigos -, que herdam os atributos e métodos de *Character,* inclusive um método virtual, pois nesse caso usamos polimorfismo para diferenciar a forma de instanciar os feitiços do inimigo e do jogador.

Classe *Spells:* representa os feitiços, que podem ser utilizados tanto pelo jogador como pelo inimigo no momento da batalha.

Classe *Objects:* representa objetos que também podem ser usados na batalha, além dos feitiços. É uma classe mais geral de *Potions* e *Artifacts* (poções e artefatos).

Classe *Game:* inicializa o jogo. Inicializa o jogador, após ser feita a escolha de sua varinha, de sua casa e do patronum, instancia os feitiços, poções, artefatos e ataques especiais e inicializa também os inimigos que aparecerão durante o jogo.

Struct *Stats:* Struct que agrupa todos os atributos de um personagem relacionados a batalha, o que facilitou a manipulação destes de acordo com os acontecimentos da batalha.

Classe *Battle:* implementação da batalha

- A batalha é inicializada, recebendo os atributos do jogador e do inimigo que participarão dela.

- Cada batalha é formada por vários “rounds” (rodadas) e em cada rodada é definido se é a vez do jogador ou do inimigo
- Os stats de cada personagem são alterados de forma simultânea de acordo com os efeitos dos ataques, poções e feitiços e são armazenados em um vector durante a batalha, de forma que é possível acessar o valor do stats atual e os anteriores, o que é importante para o funcionamento de efeitos que tem uma duração limitada e do vira-tempo, por exemplo, que volta uma rodada e restaura o valor de stats anterior. Para isso existem os debuffs, tanto do jogador como do inimigo, que são vectors que armazenam temporariamente as mudanças que vão ocorrer nos stats dos personagens em cada rodada. Outros exemplos disso:
- Capa da Invisibilidade: o jogador fica invisível para seu inimigo durante 2 rounds, de forma que o inimigo não consegue atacá-lo, ou seja, ele joga por 2 rounds seguidos.
- Varinha das Varinhas: os stats do jogador são alterados durante 2 rounds, de forma que ele fica mais poderoso, ou seja, seus ataques causam maior dano ao inimigo.
- Quando o inimigo não for do tipo humano, seu vector de feitiços estará vazio e ele terá um ataque especial. Existem poções contra alguns desses ataques, por exemplo, se o inimigo for do tipo Aranha, é possível usar um antídoto contra veneno de aranha, que não terá nenhum efeito se utilizado contra outro tipo de inimigo.

Classe *Exceções:* classe criada para o tratamento de exceções, isto é, quando o jogador tentar fazer algo inválido, como inserir um caractere inválido ou tentar usar um artefato impossível naquele momento, avisá-lo e não permitir para evitar problemas no funcionamento do jogo.

Além do tratamento de exceções, pudemos aplicar outros conhecimentos de boas práticas de programação, como refatoração, atacando os “bad smells” do código, nesse caso destacaram-se algumas ocorrências de números mágicos.

3 Testes

Os testes realizados procuraram testar o código em suas menores frações e, como desenvolvemos com c++ orientado a objetos, estas correspondem aos

métodos de cada classe. Assim realizamos o seguinte conjunto de testes para aferir o comportamento do programa em determinadas circunstâncias.

Character: Nesta classe testamos o construtor para verificar sua validade em situações adversas, testamos os métodos de retorno de informações individuais e de instanciar novo bloco de informações (*struct* _baseStats).

Wizard: Além do construtor e dos métodos de retorno, testamos a lógica da distribuição de pontos de habilidade, o uso dos vetores que são instanciados no o objeto e seus respectivos retornos.

Enemy: Esta é uma classe menor, sendo assim, apenas foi necessário testar o construtor, métodos de retorno e a lógica de decremento de habilidades no objeto.

Objects: Esta é uma classe mais generalizada que utiliza o conceito de hierarquia como classe mãe de Potions e Artefacts, sendo assim apenas foi necessário testar o construtor e os retornos de métodos.

Spells, Potions e Artefacts: Assim como Enemy, estas também são classes pequenas, onde apenas foi necessário testar o construtor, os métodos de retorno e de incremento.

Game: Esta é uma grade grande e complexa que implementa diversas subclasses, assim apenas foi necessário testar se os erros lançavam exceções.

Battle: Esta também é uma grande classe que implementa diversas subclasses, no entanto foi possível testar os construtores e a lógica de decremento de habilidades, vida e magic points.

4 Conclusão

Neste trabalho foi possível aprimorar os conhecimentos de programação na linguagem C++ e aplicar de forma prática os conceitos de POO e de boas práticas adquiridos em sala de aula.