UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II

Projeto Final - Jogo Uno - Team 13

Eugênio Ribeiro Da Silva
Jonas Ferreira Da Trindade
Wesley Raul Santos Vieira

Belo Horizonte
2019

INTRODUÇÃO

Atualmente, os jogos digitais são uma categoria de software bem popular. Eles colocam o foco nos usuários a todo momento, pois existem puramente para proporcionar uma experiência que seja agradável a eles. Considerando estes pontos, decidimos que desenvolver um jogo seria uma boa forma de consolidar os conhecimentos adquiridos durante a disciplina. Por isso, objetivo deste trabalho é replicar uno, um jogo de cartas multiplayer, em um ambiente virtual.

Uno é um jogo de cartas simples no qual vence o jogador que consegue jogar todas as cartas que tem em sua mão. Podem jogar entre 2 e 10 jogadores, e cada jogador possui 9 cartas. O jogo é iniciado com a carta do topo do baralho, e cada jogador deve jogar uma carta da sua mão que contenha o mesmo símbolo ou cor da última carta jogada. Caso o jogador não possua em sua mão uma carta que satisfaça esses critérios, deve pegar cartas do baralho até encontrar alguma que possa jogar. O jogo é composto por 108 cartas, cada uma possuindo uma cor, podendo ela ser azul, verde, amarela ou vermelha. Cada carta também apresenta um dentre 15 símbolos, que são: números de 0 a 9;

- um símbolo que faz com que o próximo jogador tenha que pegar mais duas cartas;
- um símbolo que faz com que o próximo jogador perca a sua vez; • um símbolo que inverte a ordem do jogo;
- um símbolo que permite ao jogador que o jogou mude a cor da carta que o próximo jogador terá que lançar;
- um símbolo que permite ao jogador que o jogou mude a cor da carta que o próximo jogador terá que lançar e faz com que ele tenha que pegar mais 4 cartas. As cartas que contém os dois últimos símbolos são as únicas que não possuem cor, podendo ser lançadas independentemente de qual foi a última carta jogada.

IMPLEMENTAÇÃO

Descrição do Programa

Quando o programa é executado, um objeto Jogo é criado. O baralho é feito e embaralhado logo em seguida, e o usuário pode inserir informações sobre os jogadores. Para cada jogador, é dado um objeto do tipo Baralho contendo 9 cartas, que são as cartas que ele terá na mão. A carta que está no topo do baralho principal do jogo passa a ser a primeira da pilha de descarte e assim o loop do jogo inicia-se.

O jogo acontece em um loop while, que é executado até que haja um vencedor. Essa condição é checada ao final de cada jogada, quando é conferido se o jogador da vez possui 0 cartas em sua mão. Ao iniciar a vez de um jogador, primeiramente é checado se este jogador pode realizar a sua jogada. Se ele estiver sob o efeito da carta Bloquear, uma mensagem aparece em sua tela dizendo que ele está bloqueado. Caso contrário, as cartas que ele tem em sua mão aparecem, seguidas da última carta lançada, e ele pode lançar uma carta. Ao final da vez de

cada jogador, a tela é limpa para que um jogador não possa ver as cartas que os outros tem em mão. A ordem na qual os jogadores jogam é a mesma ordem na qual eles foram inseridos, e só muda quando alguém lança uma carta Inverte, fazendo com que a ordem passe a ser contrária.

Os objetos utilizados na implementação do projeto foram:

- Carta, uma classe para descrever as cartas do jogo. Cada objeto desta classe possui uma cor e um ID;
- Baralho, uma estrutura *vector* de cartas usada para a construção do baralho do jogo e para armazenar as cartas que o jogador tem em mãos;
- Pilha, uma classe para armazenar a pilha de descarte que se diferencia do baralho por ter a função reembaralhar;
- Jogador, no qual cada objeto desta classe armazena informações sobre nome e as cartas que um jogador específico tem em mãos;
- Jogo, que guarda informações como o sentido do jogo e quem será o próximo jogador.

Decisões de Implementação

As cartas do baralho são importadas de um arquivo, uma prática bem comum em jogos para garantir a integridade dos dados. Por exemplo, se há uma função para gerar o baralho e por algum motivo ela falha, a referência do baralho é perdida. Em um arquivo, sempre temos a referência certa guardada. Utilizando um arquivo também facilitamos os testes, pois podemos criar um baralho só com as cartas que queremos testar. Utilizamos a estrutura map pois manipular inteiros é mais simples, possui custo menor e não necessita de biblioteca, como é o caso ao manipular strings. Ao manipular as cartas como inteiros, simplificamos o programa e evitamos erros.

As classes usadas no programa são:

Baralho contendo os métodos de consulta topoDoBaralho, fundoDoBaralho, retornaCarta; métodos de ação do baralho comprarCarta, cartasNoBaralho, embaralhar; métodos de adicionar e remover removerCarta, adicionarCarta e método para montar o baralho importaBaralho.

Carta possui apena o constrututor que contendo atributos de id e cor.

Jogador contendo métodos de consultas do jogador cartasNaMao, checkCarta, imprimeMao; e métodos de ação do jogador jogarCarta, falarUno e compraCarta.

Jogo possui os métodos de sentido inverteSentido, métodos de criação de jogadores criaJoagadores, proximoJogador, setJogadorAtual, getJogadorAtual, setJogador; métodos de retorno de mapeamentos retornaMapaID, retornaMapaCor; e métodos de auxiliares do jogo imprimeJogada, aguardaComando, aguardaJogadaValida, checaIndiceValido, realizaEfeitoCartaEspecial.

Pilha contendo os métodos de imprimirTopo e reembaralhar.

TESTES

Os testes implementados verificam as principais funcionalidades do jogo, como o teste que verifica se quantidade de cartas nas mãos de dois jogadores está correta após uma sequência de jogadas; teste que verifica o retorno das cartas de baralho na mão de um jogador; teste que verifica se a lógica do jogo está retornando as cartas corretas do topo e do fundo do baralho e um teste de nomes de jogadores.

COMO COMPILAR

O primeiro passo para compilar o programa é clonar o repositório com o link https://github.com/pds2/20192-team-13.git. A compilação é realizada a partir do Makefile bastando executar os seguintes comandos:

```
make
make run
```

CONSIDERAÇÕES FINAIS

Este trabalho foi de grande importância para compreendermos os conceitos dados durante a disciplina. Pudemos entender na prática assuntos como a Orientação a Objetos, como fazer testes, tratamento de casos para garantir a robustez do nosso algoritmo e como trabalhar em equipe utilizando o controle de versão.