Napoleão

Vítor Ângelo Russi
Departamento de Engenharias da Mobilidade
Universidade Federal de Santa Catarina – Centro Tecnológico de Joinville.
Joinville, Brasil
vitor.russi@hotmail.com

Palavras-chave—QtCreator; Napoleão; cartas; c++; POO; programação III

I. Introdução

Este relatório é referente ao trabalho final da disciplina EMB5631(Programação III), ministrada pelo professor Gian Ricardo Berkenbrock, no primeiro semestre do ano de 2017. O trabalho se dividiu em duas partes: construir um repositório em comum, para todos os participantes da matéria usarem, e depois especializá-lo, construindo o próprio código. O código deverá estar no GitHub no endereço "https://github.com/gianricardo/prog3", e o objetivo é fazer um jogo de cartas, a escolha do autor, e executá-lo utilizando a interface gráfica "QtCreator.". Toda a implementação do trabalho deve ser feita usando a linguagem C++, respeitando todos os conceitos de Programação Orientada a Objeto(POO).

II. Desenvolvimento

A. O jogo

O jogo escolhido pelo autor foi o "Napoleão". É um jogo de cartas que utiliza um baralho comum, de 52 cartas, onde ganha o jogador que fizer o maior número de pontos dentro das rodadas especificada pelos jogadores. Para a implementação no QtCreator, algumas regras foram alteradas.

B. Funcionamento do jogo

O jogo implementado tem as seguintes regras:

1. Apenas dois jogadores jogam.
2. O número de rodadas é pré-definido pelo usuário.
3. Cada rodada tem 5 turnos.
4. Os jogadores recebem 5 cartas cada um.
5. O jogo segue a mecânica de comparação de cartas: as cartas são comparadas, levando em consideração o naipe e o valor, e a que ganhar, o participante de jogou ganha o turno.
6. Os participantes então dizem quantos turnos eles acham que vão fazer, de 0 a 5. Se o primeiro jogador falar 0, o segundo tem que falar que faz pelo menos 1 turno. A pessoa que teve a maior aposta se torna o declarante.
7. O declarante então escolhe o naipe de trunfo: para o naipe escolhido, qualquer carta desse naipe terá uma vantagem sobre as cartas de naipe diferente.
8. Após realizado todos os acertos iniciais (rodadas e trunfo), começa o turno, onde o declarante escolhe uma carta e a joga no monte. O naipe da carta jogada é marcado, chamado de "naipe do turno".
9. O outro participante, então, escolhe uma carta para jogar no monte, e as duas cartas são comparadas na seguinte hierarquia:
    1. Se uma carta tiver o naipe de trunfo e a outra não, o trunfo ganha.
    2. Se uma carta tiver o naipe do turno e a outra não, o naipe da rodada ganha.
    3. Se as duas cartas tiverem o mesmo naipe, ganha a que tiver maior valor, onde o "Ás" é a mais fraca e o "Rei" é a mais forte.
10. No fim do turno, vê-se quem ganhou o turno: se foi o declarante, soma 1 no seu contador de turnos ganhos.

11. Após os 5 turnos, verifica-se o número de turnos ganhos pelo declarante, e compara com a aposta inicial: se o declarante fez o número de turnos que ele disse que ia fazer(nem mais nem menos), ele ganha pontos igual ao número de turnos declarado, senão o seu adversário ganha esses pontos.
12. Ao fim das rodadas, o jogador que tiver mais pontos ganha a partida.

C. O código

Para a construção do código, foi utilizado um repositório em comum à todos os participantes da materia, o “carteado”, localizado em “https://github.com/gianricardo/prog3/carteado”, para fazer todas as operações relacionadas com as criação de jogadores, cartas, baralhos, montes, mesas e regras. Na especialização para o jogo em questão, foi criada a classe “Napoleão”, que contém os métodos necessários para a execução do jogo. Essa classe tem um atributo do tipo “Jogo”, que faz a interação do Napoleão com as jogadas. Essa classe que executa todas as comparações, as movimentações de cartas e aplica as regras do jogo.

Também foi criada uma classe “Tela”, que faz a interação do jogo com o usuário. Essa classe controla a linearidade do jogo, decidindo quando acaba uma rodada, quando acaba o jogo, escolhe de quem é a vez e pede as informações para os jogadores. Essa classe possui um atributo do tipo Napoleão, e faz o link do jogo com o usuário.

Para poder jogar sozinho, foram criadas 2 classes, “NapoleaoPessoa” e “NapoleaoIA”, aonde o primeiro se relaciona com o usuário e o segundo se relaciona com a IA. A IA apenas faz escolhas aleatórias, limitado pelas opções existentes, não seguindo nenhum tipo de padrão. Para poder organizar os jogadores em um único vetor, foi criada uma classe abstrata “JogadorNapoleao”, que contém apenas métodos virtuais puros. Essa classe foi criada para, utilizando os conceitos de polimorfismo, organizar melhor os jogadores, chamando os seus métodos apenas pela sua posição no vetor de jogadores.

O código tem duas versões: a versão implementada no terminal e a versão implementada com interface gráfica(Qt). Ambos tem as mesmas classes, a maior diferença é que na versão do terminal, a classe “NapoleaoPessoa” tem os comando de entrada de informação do usuário, e a sua execução inteira é apenas no terminal, já a versão do Qt as interações com o usuário são feitas pela classe “MainWindow”, do próprio QtCreator.

D. O jogo em execução

No inicio do jogo, aparecerá a janela principal no fundo, com uma caixa de diálogo encima. Irão aparecer 3 caixas antes de poder ter acesso à janela principal.

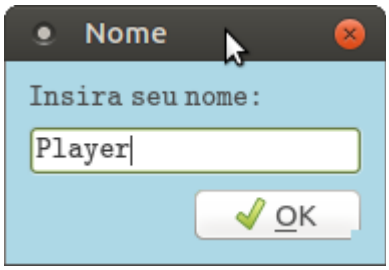

Fig. 1: Primeira caixa de diálogo

Após as configurações inicias, o jogo então poderá ser inicializado. As cartas foram distribuídas, todas viradas pra baixo, e para efetivamente começar o jogo deve-se clicar em “Inicio” na parte superior.

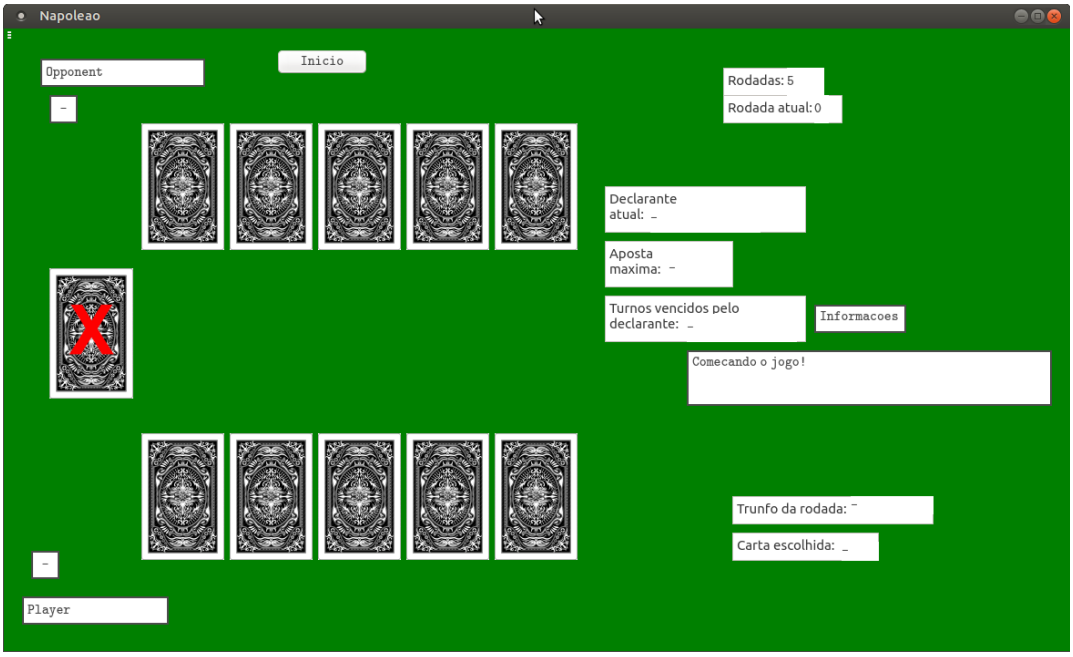

Fig. 2: Tela logo após o fim das configurações iniciais

Na caixa “Opponent”, vai aparecer o nome do adversário, a mesma coisa pra caixa “Player”. Nas pequenas caixas quadras próximas aos nomes vão aparecer as pontuações, na direita tem algumas informações impontantes, e logo abaixo da caixa “Informações” irão aparecer mensagens, dizendo para o usuário o que ele deve fazer, e dando algumas informações básicas sobre o que tá acontecendo.

Logo após pressionar o botão inicio, o jogo vai mostrar as cartas do usuário, vai mostrar alguns botões à direita, e perguntar o numero de turnos que ele acha que vai fazer.

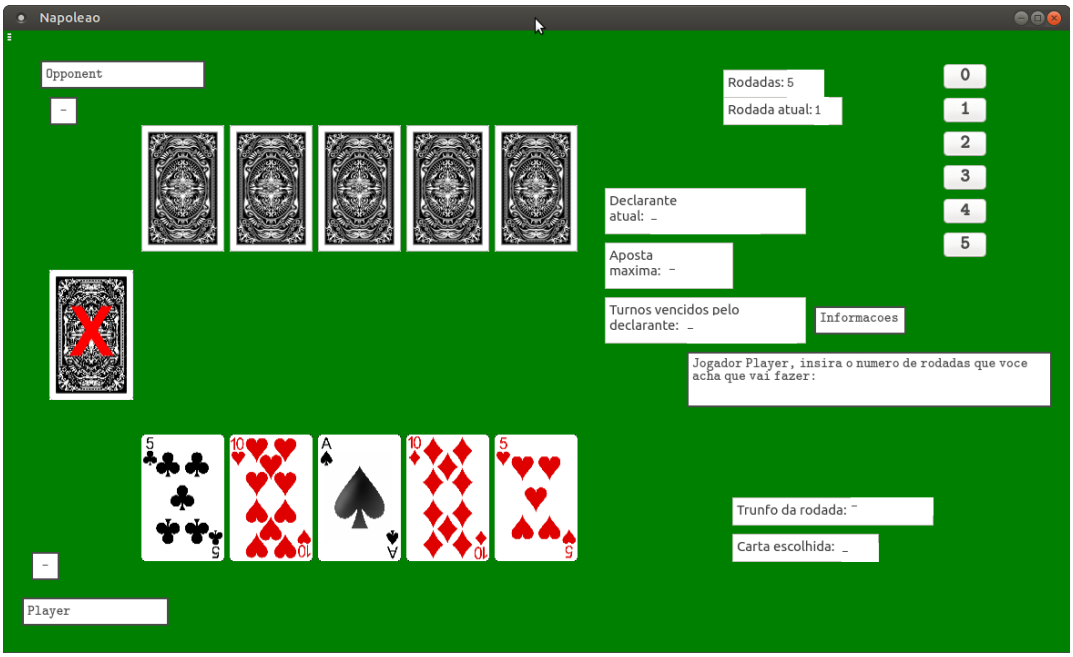

Fig. 3: Tela depois de pressionar Inicio

Os botões que apareceram à direita são relativos ao numero de turnos que o usuário acha que vai fazer. Após isso, mais botões vão aparecer, relativo ao naipe, caso o usuário seja o declarante.

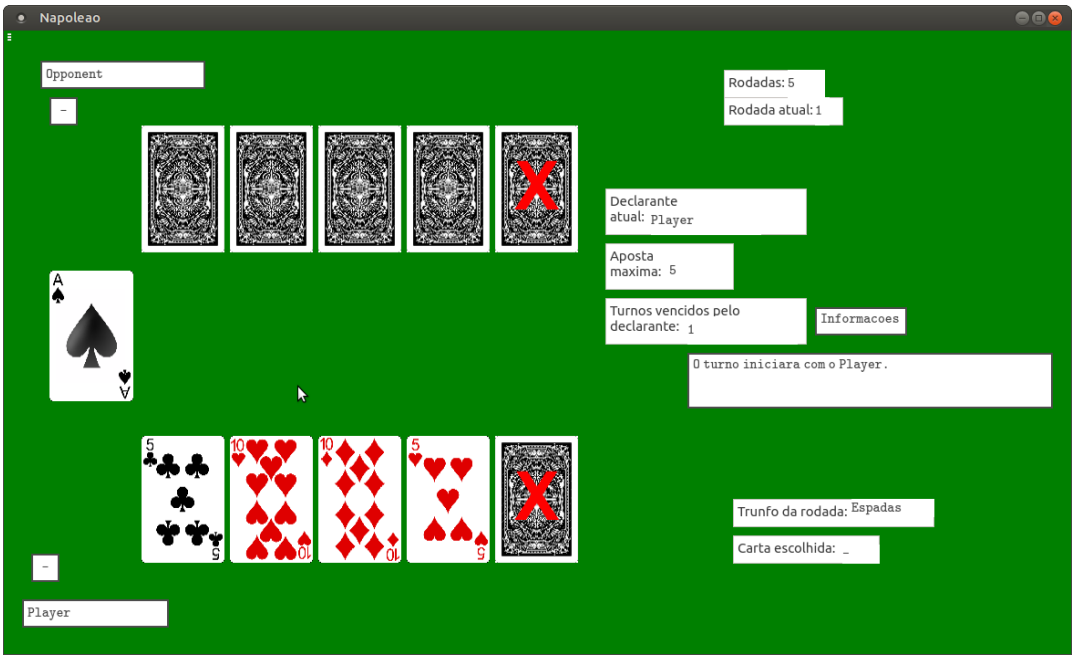

Fig. 4: Tela após primeira jogada

Ao final de cada turno, o jogo diz quem que ganhou o turno, e atualiza a caixa “Turnos vencidos pelo declarante”. No final da rodada, o jogo verifica quem ganhou, e soma os pontos corretamente. Dessa forma, o jogo vai se desenrolando até a ultima rodada, que mostra se o usuário ganhou, empatou ou perdeu o jogo, e logo depois a janela principal fecha automaticamente.

Na implementação do jogo, foi feito uma opção para mostrar a mão do adversário. A execução é a mesma, apenas que em cada turno, o usuário pode ver a mão do adversário, o resto continua tudo igual. Também foi implementado um modo simulação, onde toda parte onde o usuário iria inserir uma entrada(cliques), o valor é gerado como se fosse um jogador IA, o único problema é que a impressão da mão do usuário, quando o modo simulação tá ativo, fica estranho, repetindo a imagem das cartas. Para ativar esses modos, basta apenas modificar as constantes no arquivo “tela.h” e recompilar.

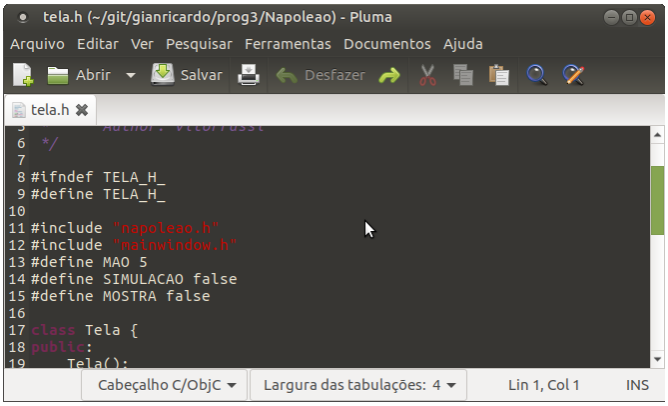

Fig. 5: Arquivo “tela.h” com as constantes “SIMULACAO” e “MOSTRA”

III. DISCUSSÕES

A. Dificuldades encontradas

O código, quando feita a versão no terminal, não apresentava nenhum vazamento de memória, nenhum *bug*, podia jogar de 2 à 6 pessoas e até 1000 rodadas, isso porque a familiaridade do autor com a programação voltada para o terminal é muito maior do que voltara para uma interface gráfica. O código foi feito para um número variável de jogadores, na versão do terminal, porém como a versão do Qt tem apenas 2 jogadores, algumas partes do código ficaram “estranhas”, pois o mesmo utiliza vetores, laços for, para apenas 2 jogadores. A falta de familiaridade com o QtCreator impossibilitou a organização correta dos arquivos no repositório do GitHub.

Outra grande dificuldade foi tentar entender como que o QtCreator faz o gerenciamento de memória das suas variáveis dinâmicas, pois na versão do terminal, o código não teve vazamentos de memória, pois o autor apenas utilizou a classe std::vector e ponteiros inteligentes (unique_ptr). Porém ao executar a ferramenta de verificação de erros de memória *Valgrind*, este acusou diversos erros, que na confecção do código, estes erros foram ignorados.

Um grande problema enfrentado foi na organização das classes: como a classe Tela tem um atributo Napoleão, não tem como a classe Napoleão pedir as jogadas do usuário, como controlar a chamada dos métodos dos jogadores, obrigando a passar uma parte dessa função para a Tela. Para conseguir integrar o código com a interface gráfica, ou seja, a Tela com a MainWindow, foi feito da seguinte forma: a tela possui um ponteiro para MainWindow, e a classe MainWindow possui um ponteiro para a Tela, para que a Tela possa chamar corretamente os métodos de entrada de dados, e a MainWindow possa enviar os sinais que ela recebe e poder trocar informações com a Tela.

Por conta que a comparação entra as cartas do jogo Napoleão é única(naipe de trunfo, naipe de rodada), não foi possível apenas utilizar o operador padrão da classe Carta, foi necessário implementar um método novo que faz essa comparação. Para facilitar a linha de raciocínio do autor, também foi implementado o seu próprio método de controle de turnos/rodadas, apesar de existir isso no repositório comum “carteado”.

IV. CONCLUSÃO

A realização desse trabalho possibilitou ao autor ter um contato maior com as ferramentas do C++, pois como o mesmo apenas teve contato com a linguagem C, essa implementação possibilitou ampliar os conhecimentos. Fazer um projeto faz com que o programador desenvolva a sua lógica e criatividade, além de melhorar a sua capacidade de resolver problemas(que por acaso foram muitos), além de conhecer uma nova ferramenta que é o GitHub, e por fim o trabalho apresentou ao autor um contato com uma IDE voltada para interface gráfica, que o ajudou a conhecer como que funciona uma implementação de um jogo simples utilizando uma interface que não é o próprio terminal do Ubuntu.

Por fim, esse trabalho trouxe experiência para o autor em trabalhar em equipe, em criar/utilizar uma biblioteca, em desenvolver uma interface gráfica e em produzir o jogo em si, chamado Napoleão.

AGRADECIMENTOS

A Universidade Federal de Santa Catarina, que possibilitou a criação e realização da matéria Programação III.

Ao professor, que auxiliou e acompanhou o projeto, assim como auxiliou na estruturação do trabalho, na utilização da linguagem C++ e nos conceitos de POO.

Aos participantes da matéria Programação III, que também são os integrantes do grupo que fez o repositório, pelo auxilio na confecção dos códigos e em opiniões pessoais sobre como fazer a implementação.