BlackJack (21)

André Luigi Bonote
Departamento de engenharias da mobilidade
Universidade Federal de Santa Catarina - Centro tecnológico de Joinville
Joinville, Brasil
andreluigibonote@icloud.com

Palavras chave—Qt; QtCreator; BlackJack; orientação a objeto; c++

I. INTRODUÇÃO

Este trata do desenvolvimento do trabalho de conclusão do curso programação III (EMB5631) no semestre 20171 (primeiro semestre do ano de 2017), em especifico da parcela individual executada pelo autor tendo como base o repositório "prog3/carteado/src/p3" disponível no GitHub no endereço https://github.com/gianricardo/prog3.git e desenvolvido em conjunto por todos os participantes do curso no semestre especificado, que deve conter uma interface gráfica para interação com o usuário desenvolvida através da IDE "Qt Creator".

II. JOGO DE CARTAS

A. O jogo escolhido

Tendo em vista que a proposta do trabalho era a utilização das classes disponíveis no repositório criado em conjunto para que fosse criado um jogo de cartas o autor escolheu uma variante do clássico jogo de BlackJack, também chamado de 21, do qual o desenvolvimento é relatado nesse.

B. Regras

O BlackJack na variante escolhida tem como mecanismo de jogo a seguinte dinâmica:
Dois participantes se confrontam sendo um o jogador, ou apostador e o distinto o dealer, que é o representante da banca;
Inicialmente o jogador realiza uma aposta que tem como valor mínimo $5,00, que pode ser incrementado com o passo de $1,00, sendo o máximo de $1000000,00;
O saldo máximo que o jogador pode atingir durante o jogo é de $999999999,00;
Caso o jogador não tenha saldo para apostar, ele não poderá jogar.
São então distribuídas 2 cartas para cada participante sendo que apenas uma das cartas do dealer não tem o valor revelado;
Nesse ponto o jogador deve tomar uma das seguintes ações até que decida terminar sua rodada:

Hit - O jogador recebe mais uma carta com o seu valor revelado (até atingir o máximo de 30 cartas);
Double - O valor da aposta do jogador é multiplicado por dois ou (caso o saldo não o permita) se torna o saldo total do jogador, o jogador pode dobrar sua aposta 3 vezes por partida;
Finalizar a jogada - O jogador informa que está satisfeito com as cartas que possui e o valor da aposta.

Após o jogador finalizar a jogada a carta oculta do dealer é revelada e ele deve ter as mesmas ações descritas anteriormente para o jogador, exceto dobrar a aposta;

Tendo então o dealer finalizado sua jogada é verificado o jogador da rodada da seguinte maneira, e nessa ordem de precedência:

Jogador ganha se somar 21 pontos;
Jogador ganha se o dealer somar mais de 21 pontos;
Jogador perde se fizer mais de 21 pontos;
Jogador ganha se estiver com uma soma mais próxima de 21 do que o dealer;
Jogador perde caso outra configuração de pontuação ocorra.

Sendo que as cartas valem o número de pontos de acordo com seu número, sendo "A" de valor 1 e as cartas acima de 10, valete, dama e rei, tem o valor de 10 pontos;
Encerrada a verificação o valor da aposta é subtraído do saldo do perdedor e adicionado ao saldo do ganhador;

A ultima etapa se trata da decisão do jogador em jogar uma nova partida ou encerrar o jogo.

III. PARTICULARIDADES DO CÓDIGO

Para a criação do BlackJack foram especializadas as classes: “JogoBasico” em “BlackJack" de modo a realizar as ações específicas do jogo e também para gerenciar as classes criadas especificamente para esse projeto e “Regra" em “Regra21" para que fosse possível especializar as regras do jogo. As classes criadas são: “IA_Dealer” responsável pelas decisões de jogada do dealer que nesse caso não é um participante real (humano), “Banco21" que tem por responsabilidade a “tesouraria" permitindo apostas, realizando as transações do fim de cada rodada e também impedindo que o jogador faça novas apostas caso não tenha mais saldo e finalmente uma classe “MainWindow" que dentro de seu escopo também gerencia outras diversas classes que são usadas para a realização de sua responsabilidade, a interação com o usuário humano através de uma interface gráfica.

A. Diagrama UML

A imagem abaixo apresenta através de um diagrama UML a interação entre os objetos das classes utilizados no jogo:

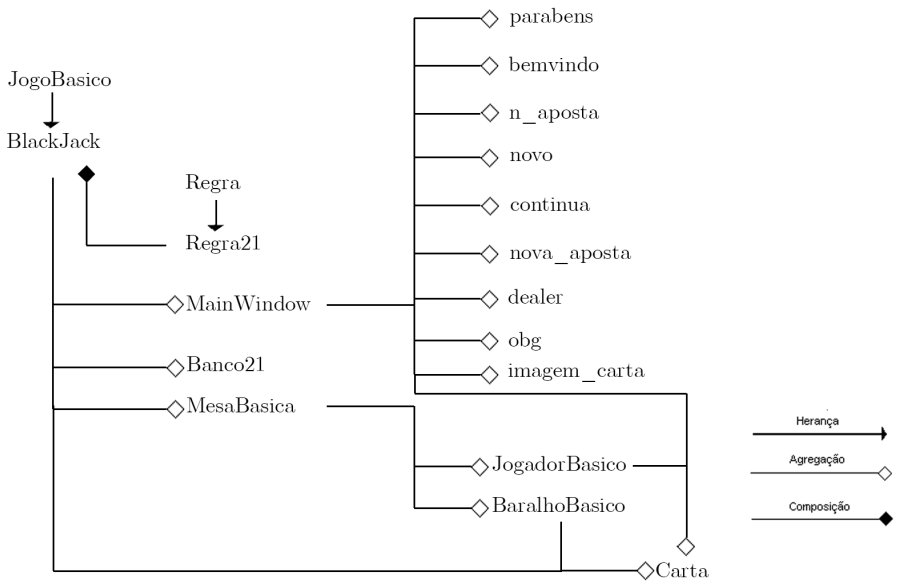

Fig. 1. Diagrama UML.

É de BlackJack a responsabilidade de gerenciar as ações e interações de todas as classes a ele ligadas;

Regra21 tem a informação e é responsável pela verificação da pontuação da carta;

Banco21 cuida de todas as transações envolvendo valores de aposta e saldo;

MainWindow é a interface gráfica, gerencia as janelas de diálogo e funciona como intermediário da interação entre o jogo e o jogador humano.

As decisões sobre a criação e herança de classes se deu analisando uma maneira de obter os resultados (funcionamento do jogo) respeitando a discussão inicial de modelagem do repositório disponível e também de modo a tentar simular o ambiente do jogo que nesse caso seria um cassino.

As implementações de métodos e parâmetros foram feitos buscando que os resultados fossem obtidos mantendo as responsabilidades acima citadas e são melhor acompanhadas através da documentação (comentários) presentes nos arquivos do tipo header (.h) onde além de todas as ações realizadas por esses também especifica seus retornos e parâmetros.

B. Dificuldades encontradas e como foram contornadas

Devido a pouca familiaridade com os conceitos de orientação a objeto a modelagem do diretório apresentava algumas incoerências sendo a que mais poderia prejudicar este seria a impossibilidade de acessar um objeto de uma classe derivada de

Regra pois a classe JogoBasico possui um ponteiro que é exclusivo do tipo da classe base. Para solucionar isso a opção foi criar o método necessário, que neste caso seria a verificação de quantos pontos a carta deveria valer (1 para A, seu número para valores entre 2 e 10 e 10 para J, Q e K);, e no momento que é necessário fazer o uso desse método é criado um objeto do tipo específico da classe derivada de regra, no escopo onde este é requerido, o método é utilizado e o objeto destruído.

Outra questão importante foi a criação da classe Banco21 pois mesmo sendo possível adaptar o sistema de pontuação já presente na modelagem do repositório, esse apresentava diversos incoerências com o conceito do jogo BlackJack, no momento em que o mesmo estava sendo desenvolvido, e exigiria que fossem feitas muitas alterações e adaptações para funcionar de forma satisfatória, além de que não seria totalmente condizente com o “objeto" que deveria manipular, nesse caso uma representação de valores financeiros, o autor optou por criar a nova classe em questão para gerir tais responsabilidades.

Por fim uma das fases do projeto que mais apresentou dificuldades foi a compreensão da forma como o Qt faz o gerenciamento de memória, que muitas vezes causa falsas acusações de vazamento por diversos verificadores, para que estas fossem desconsiderados e fossem corrigidos apenas aqueles que eram causados pelos códigos em si.

IV. O JOGO EM FUNCIONAMENTO

Ao iniciar o jogo será apresentada a janela principal sobreposta com a de novo jogo, onde o jogador deverá inserir seu nome para iniciar a partida:

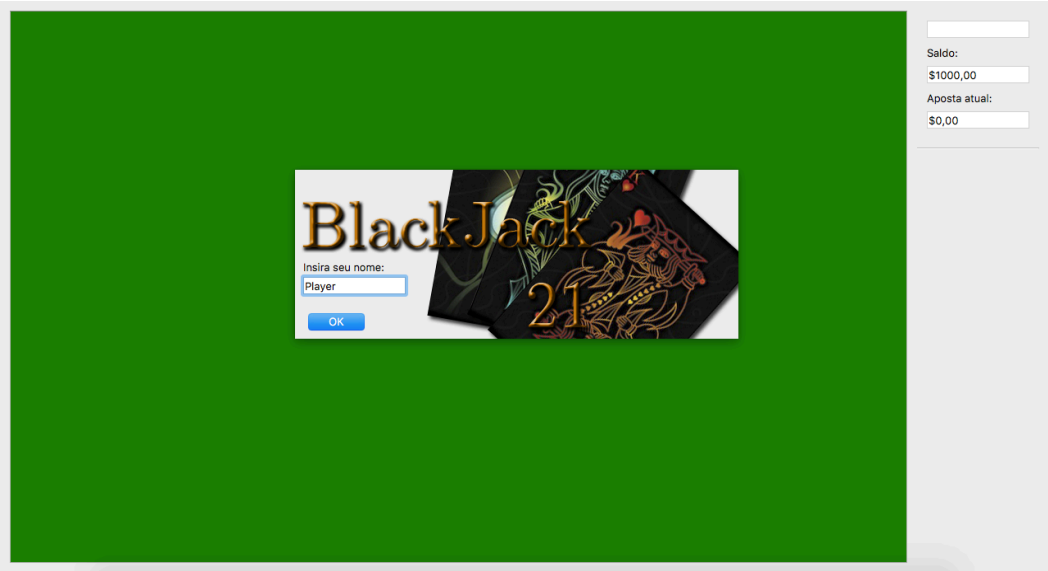

Fig. 2. Jogo ao iniciar.

Apos inserir seu nome é solicitado que o jogador faça uma aposta, o valor é inserido girando o dial e mostrado no display:

Fig. 3. Realizando aposta (apenas janela de diálogo).

As cartas serão então distribuídas e apresentadas na parte verde, também é mostrado o menu de ações que o jogador pode realizar:

Fig. 4. Jogada.

Após finalizada a jogada é anunciada a jogada do dealer e as cartas serão exibida conforme as ações deste e será anunciado o vencedor da rodada:

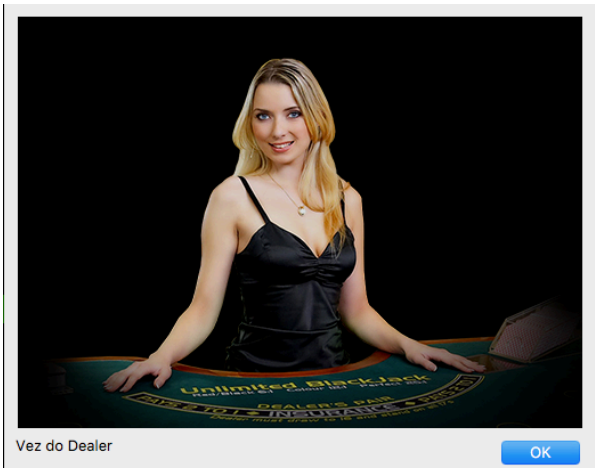

Fig. 5. Vez do dealer (apenas janela de diálogo).

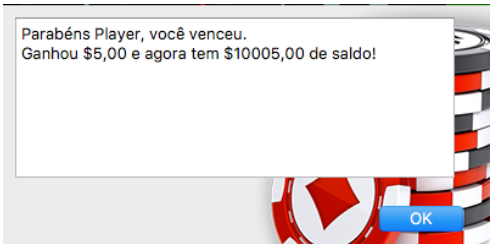

Fig. 6. anuncio do vencedor (apenas janela de diálogo).

A próxima etapa é perguntar ao jogador se ele deseja jogar novamente ou finalizar o jogo. Caso decida continuar o jogo retorna para a aposta, caso não é exibida uma mensagem de agradecimento e o aplicativo é encerrado assim que o jogador clicar em “ok”:

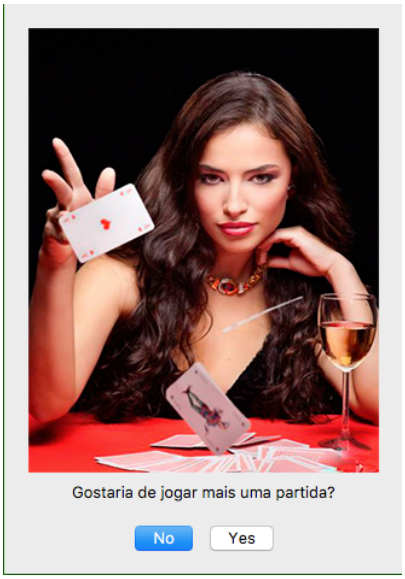

Fig. 7. Jogador decide se continua(*apenas janela de diálogo*).

Fig. 8. Agradecimento (*apenas janela de diálogo*).

Se em algum momento o jogador não estiver mais apto a jogar a seguinte mensagem será mostrada:

Fig. 9. Saldo insuficiente (*apenas janela de diálogo*).

Algumas das imagens podem apresentar algumas alterações durante a execução do jogo, como por exemplo o design dos botões quando o jogo é compilado para e executado em um sistema baseado em linux.

V. Conclusão

Com a realização desse foi possível, ao autor, aprender e familiarizar-se com as ferramentas Git, através do Github, e Qt, através da IDE QtCreator, aumentando assim o arcabouço de conhecimentos úteis para o desenvolvimento de aplicativos e relacionados de modo prático e apresentando resultados concretos desse aprendizado. Também foi extremamente enriquecedor aos já existentes conhecimentos sobre orientação a objeto em linguagens de programação, sejam estes práticos ou conceituais.

O resultado do que fora apresentado aqui pode é afirmado pelo “produto" em si, que também é o objeto desse, a aplicação BlackJack.