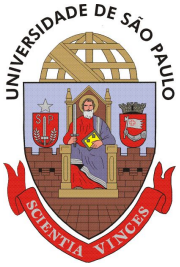

UNIVERSIDADE DE SÃO PAULO

SSC0600
INTRODUÇÃO A CIÊNCIA DA COMPUTAÇÃO I

RELATÓRIO TRABALHO IV
RUMMIKUB

Alunos:
Gabriel Santos Nicolau - 10684600
Vinícius RIbeiro da Silva - 10828141
Mateus Fernandes Doimo - 10691971
Marcelo Magalhães Coelho - 10716633
Diego da Silva Parra - 10716550
Leonardo Prado Dias - 10684642
Renan Peres Martins - 10716612

São Carlos, 28 de Junho de 2018

1.INTRODUÇÃO

Membros da equipe:

P = 0: Vinícius Ribeiro da Silva - 10828141
P = 1: Gabriel Santos Nicolau - 10684600
P = 2: Mateus Fernandes Doimo - 10691971
P = 3: Marcelo Magalhães Coelho - 10716633
P = 4: Diego da Silva Parra - 10716550
P = 5: Leonardo Prado Dias - 10684642
P = 6: Renan Peres Martins - 10716612

O trabalho consiste em implementar um programa na linguagem C para jogar Rummikub. As opções possíveis de jogo são:

- Aleatório (a distribuição de cartas iniciais aos jogadores é aleatória);
- Controlado (um arquivo de texto define a ordem da distribuição).

Obs.: O programa supõe que as cartas do arquivo estejam de acordo com as regras quanto à quantidade de cartas de cada tipo.

2. DESCRIÇÃO DO PROJETO

2.1 FUNCIONAMENTO DO PROGRAMA

Definições:

- Mesa: Ambiente onde o jogador deve colocar as cartas retiradas da mão;
- Mão: Conjunto de cartas que o jogador possui;
- Pilha de compras: Baralho em que o jogador deve pegar cartas caso não possa jogar na mesa.

O programa funciona utilizando a alocação dinâmica de memória, onde são criadas structs auto referenciadas, com ponteiros em seu interior apontando para outras structs do mesmo tipo. Dessa maneira, cria-se uma lista encadeada que guarda as informações das cartas, dos jogadores e da mesa.

As funções contidas nos arquivos ***.C*** realizam as operações com ponteiros das cartas e dos jogadores, o ***back-end***; já a interface gráfica dá os comandos para as funções, o ***front-end***.

Quando o jogador arrasta uma carta, as coordenadas na tela são armazenadas; concomitantemente o programa divide os baralhos da mesa em regiões, assim é possível saber em qual baralho o usuário soltou a carta. Dessa forma, o arquivo ***.C*** calcula e altera a posição da struct da carta modificada utilizando ponteiros.

2.2 AMBIENTE DE DESENVOLVIMENTO

O projeto foi desenvolvido no Windows 10 x64 como também no Ubuntu 16.04 LTS. Foram utilizados no Windows:

→ Code Blocks 17.12

→ Photoshop Cs6

No Ubuntu foram utilizados:

→ Sublime 3

→ Visual Studio Code

2.3 BIBLIOTECAS DE DESENVOLVIMENTO GRÁFICO

Foi utilizada a biblioteca Gtk 3.2

2.4 COMPILAÇÃO

Foram utilizados os compiladores Microsoft Visual C e MinGW GCC 6.3.0 no Windows, já no Ubuntu foi usado o compilador gcc 5.4.0.

2.5 CÓDIGOS FONTE

- Os arquivos fonte utilizados foram: main.c, system.c, eventos.c e interface.c
- A bibliotecas padrão utilizadas foram: stdio.h, stdlib.h e math.h
- As bibliotecas criadas pelo grupo foram: data.h, eventos.h, interface.h e system.h

3. TUTORIAL

3.1 COMPILAÇÃO

Ubuntu 18 (Obs.: Deve-se conter o GCC instalado) : Acesse a pasta onde estão localizados os arquivos fonte pelo terminal >> execute os comandos.

Sistema operacional:

Ubuntu com versão 18

Instalação do GCC (caso não venha pré instalado no Ubuntu):

$ sudo apt-get install gcc

Instalação do GTK:

$ sudo apt-get install libgtk-3-dev
Verificar se a versão instalada do GTK é igual ou superior a 3.2

Compilar programa:

$ make

Ou caso não deseje utilizar o makefile é possível compilar utilizando os comandos abaixo:

$ gcc -c -g -O0 -Wall -pthread -pipe src/main.c pkg-config --cflags --libs gtk+-3.0 -o main.o

$ gcc -c -g -O0 -Wall -pthread -pipe src/system.c pkg-config --cflags --libs gtk+-3.0 -o system.o

$ gcc -c -g -O0 -Wall -pthread -pipe src/eventos.c pkg-config --cflags --libs gtk+-3.0 -o eventos.o

$ gcc -c -g -O0 -Wall -pthread -pipe src/interface.c pkg-config --cflags --libs gtk+-3.0 -o interface.o

$ gcc -c -g -O0 -Wall -pthread -pipe src/init_game.c pkg-config --cflags --libs gtk+-3.0 -o init_game.o

$ gcc -o template_app main.o system.o eventos.o interface.o init_game.o -pthread pkg-config --cflags --libs gtk+-3.0 -export-dynamic

EXECUÇÃO

Linux (Ubuntu 18): Navegue pelo terminal e acesse a pasta do arquivo compilado >> Execute o comando:

$ ./rummikub

3.2 CARREGAMENTO DO ARQUIVO DE DISTRIBUIÇÃO DE CARTAS

O arquivo que dita a ordem da distribuição de cartas deverá se chamar **baralho.txt** e precisa estar na mesma pasta do arquivo **main.c**, que se localiza na pasta /src.

3.3 COMO JOGAR

O jogo consiste em passar as cartas do baralho do jogador para a mesa, mas para isso, ela deverá pertencer a um dos dois grupos de cartas válidas na mesa:

- Sequência: cartas em sequência de mesmo naipe;
- Repetição: cartas com o mesmo número, mas de naipes diferentes.

A primeira jogada de cada jogador deverá possuir uma soma de pontos maior ou igual a 30 e o jogador não poderá manipular as peças que estejam na mesa. Durante a rodada, desde que não seja a primeira de cada jogador, as cartas da mesa poderão ser remanejadas a fim de que se possa criar novos grupos válidos. Caso o jogador não consiga baixar cartas de seu baralho, o mesmo deverá pegar uma carta da pilha de compras e a vez será passada para o próximo jogador. O primeiro jogador a ficar sem cartas vence o jogo.

Obs.: Caso as cartas da pilha de compras acabem e ainda não haja jogador sem cartas, o jogo se encerrará e o aquele que possuir menor quantidade de pontos vencerá o jogo.

3.4 EXECUÇÃO DO PROGRAMA

Ao executar o programa será será perguntado ao usuário se deseja carregar o baralho de um arquivo .txt

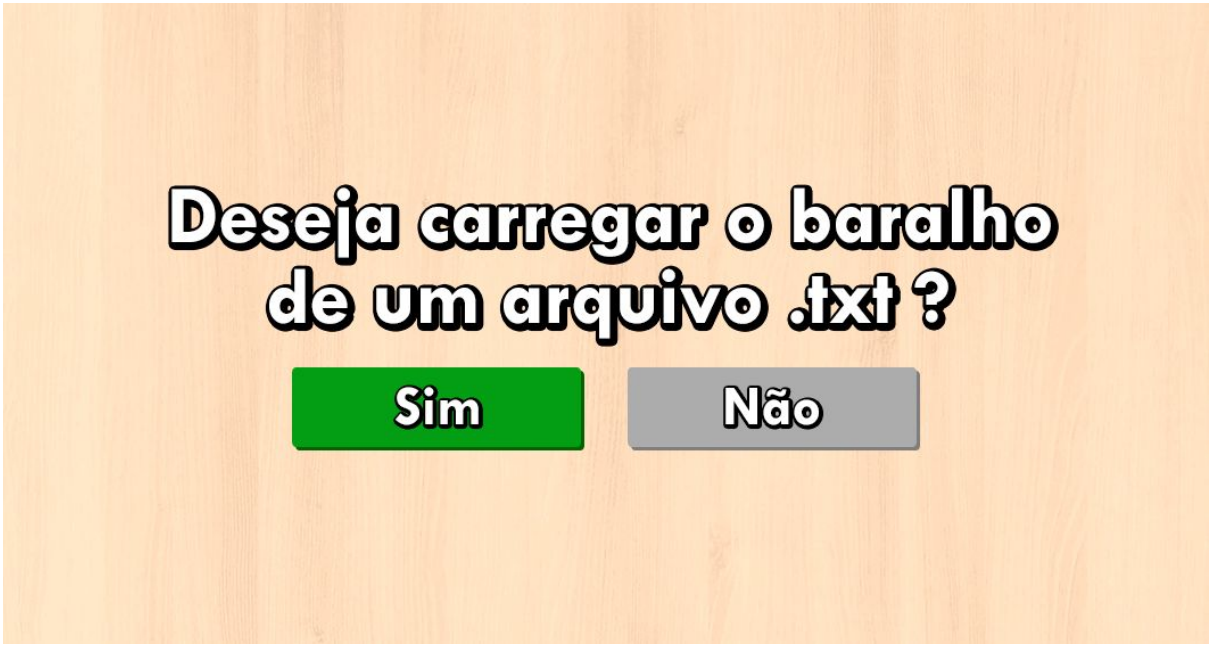

Figura 1. Opção de carregamento de cartas (Elaborada pelos autores)

Caso seja escolhida a opção de carregamento e não seja encontrado o arquivo, a seguinte tela será exibida.

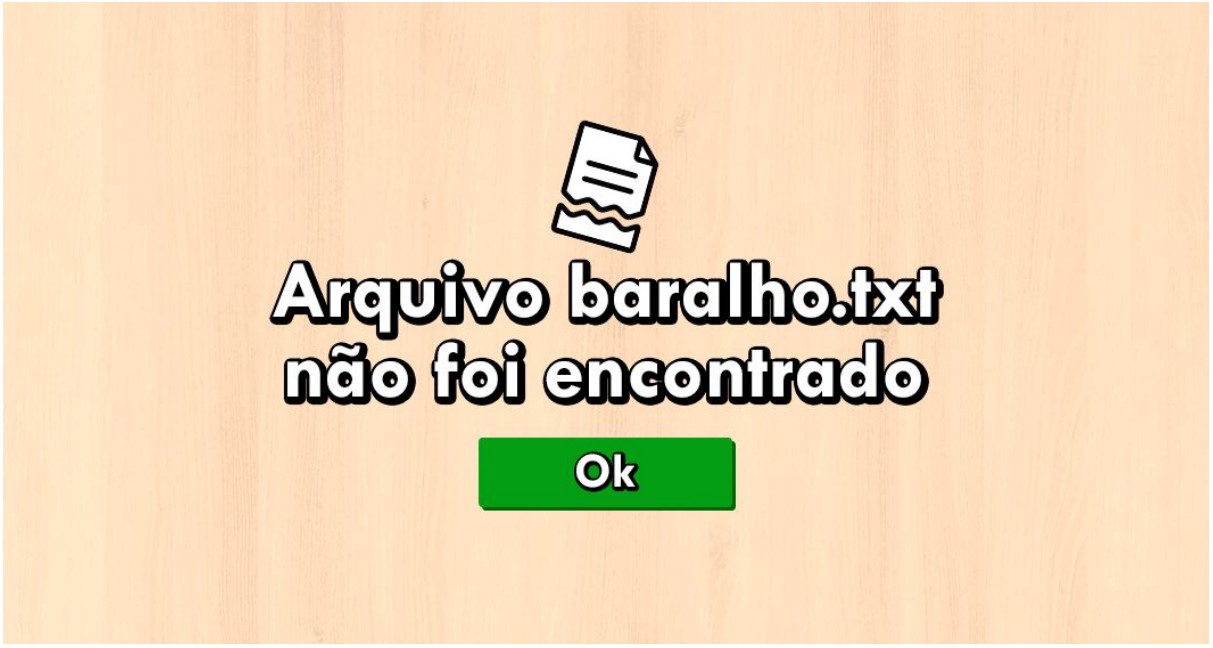

Figura 2. Tela de arquivo não encontrado

Após esses passos será exibida a com o Título do programa, nela deverá ser feita a escolha da quantidade de jogadores.

Figura 3. Escolha da quantidade de jogadores.

Após a escolha da quantidade de jogadores, será exibida a tela de boas vindas ao usuário.

Figura 4. Tela de boas vindas

A organização da tela durante a partida será da seguinte maneira:

- À esquerda da tela são exibidos os jogadores assim como aquele que fará a jogada;
- Na parte inferior da tela será exibida a mão do jogador;
- A parte central da tela é a mesa, lugar em que ocorrerá a criação dos grupos.

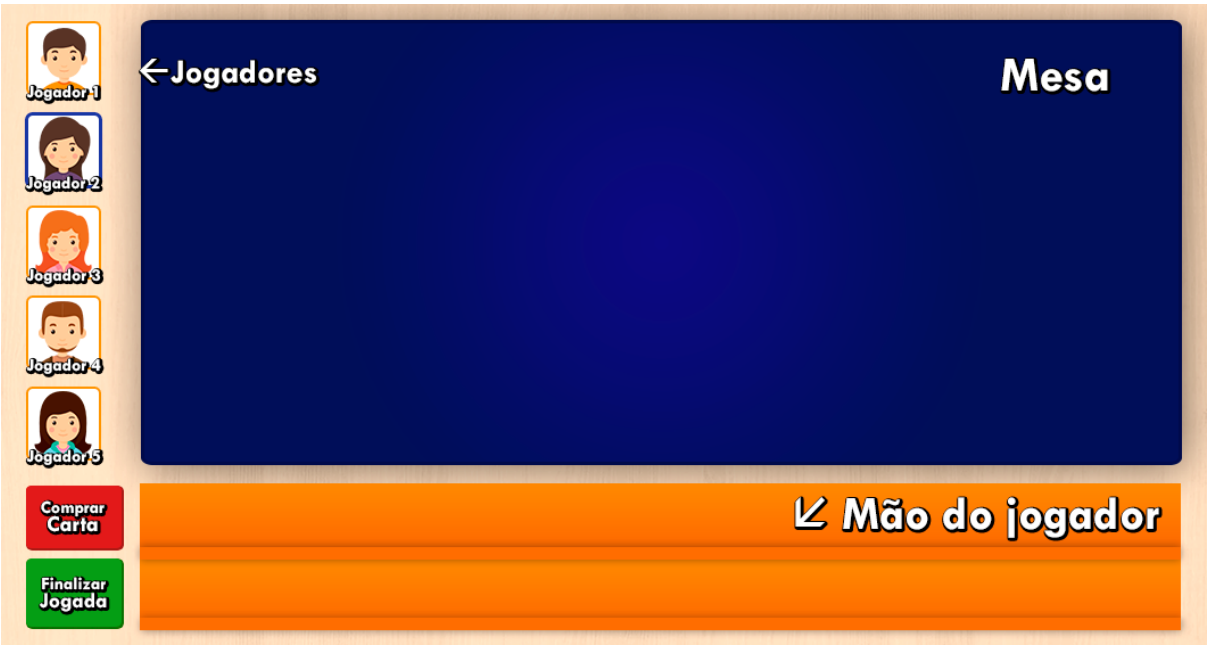

Figura 5 . Tela de jogo

Após o pressionamento do botão **Pronto**, será iniciado o jogo que funcionará baseado no mouse; o jogador deverá arrastar as cartas de seu baralho para a mesa ou remanejar as cartas da mesa a fim de fazer uma jogada válida. Após a realização da mesma, o usuário deverá clicar em **Finalizar jogada**; caso não possua cartas disponíveis deverá clicar em **Comprar carta.**

Figura 6. Exemplo de jogada

Caso o usuário tente realizar uma jogada inválida e clique em **Finalizar jogada**, o mesmo será impedido pelo programa e deverá refazer o movimento das cartas.

Figura 7. Exemplo de jogada inválida

Figura 8. Tela de jogada inválida

Assim que a jogada válida ocorrer, o programa não exibirá as cartas do próximo jogador a fim de não prejudicar as estratégias do jogo; em vez disso exibirá a seguinte tela.

Figura 8. Tela de transição de jogadas

O primeiro jogador a baixar todas as suas cartas para a mesa ganha o jogo.

Figura 9. Tela exemplo de vencedor do jogo

4. OUTRAS INFORMAÇÕES

4.1 LIMITAÇÕES

- Caso o arquivo de carregamento de cartas esteja com uma quantidade diferente das regras originisl, o programa lerá o arquivo como ele está, e o jogo pode não funcionar corretamente.

4.2 Lista encadeada

Como citado anteriormente, o programa usa uma lista encadeada para acessar os valores das cartas. Elas possuem a vantagem de não precisar de um número definido de elementos, ou seja, a lista aumenta à medida que o usuário precisa incluir um novo elemento nela.

> *As listas encadeadas são uma das estruturas de dados mais utilizadas no desenvolvimento de programas. São recursos que utilizam mecanismos de acesso encadeado, que armazenam seus dados somente quando necessário, fazendo a liberação da memória quando não mais utilizada. (MADEIRA; SIMÕES; MARTINS, 2004).*

No programa, cada informação do jogador, da carta e da mesa fica salvo dentro de uma *struct*.

5. Referências

MADEIRA, Maicon Francisco; SIMÕES, Priscyla Waleska Tagino de Azevedo; MARTINS, Paulo João. **ODIN: Ambiente Web de Apoio ao Ensino de Estruturas de Dados Lista Encadeada**. 2004. Disponível em: <http://periodicos.unesc.net/sulcomp/article/view/800>. Acesso em: 28 jun. 2018.

Gtk Change font to spin button. Stack Overflow. Disponível em <https://stackoverflow.com/questions/47083294/gtk-change-font-to-spin-button>

Styling gtk with css. The gnome journal. Disponível em <https://thegnomejournal.wordpress.com/2011/03/15/styling-gtk-with-css/>

Chap css overview. Developer gnome. Disponível em <https://developer.gnome.org/gtk3/stable/chap-css-overview.html>

Css background-image property on GtkButton. Mail Gnome. Disponível em <https://mail.gnome.org/archives/gtk-app-devel-list/2012-February/msg00011.html>

How to apply css to gtk code. Stack Overflow. Disponível em <https://stackoverflow.com/questions/47114306/how-to-apply-css-to-gtk-code>. Acesso: 02 jun. 2018