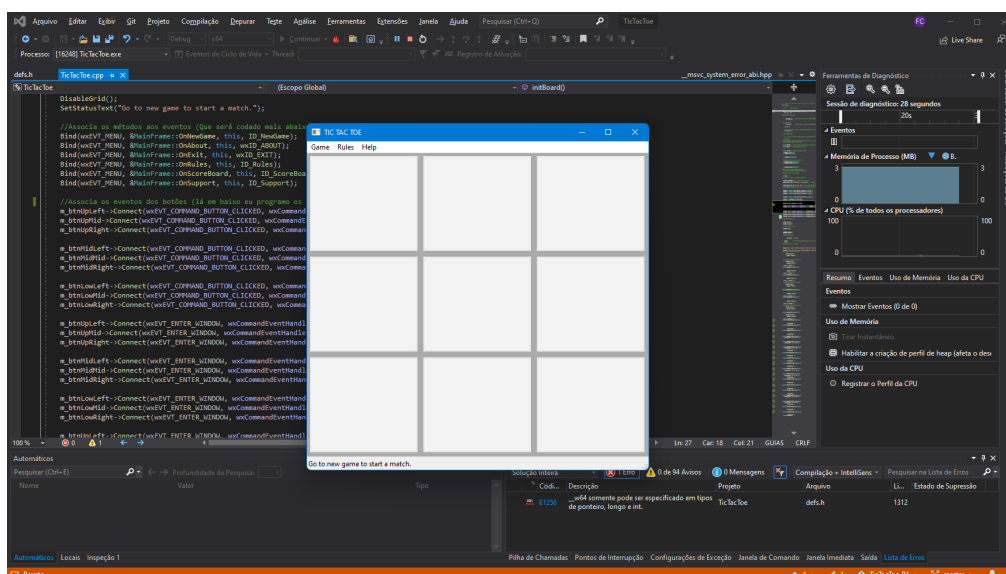


DOCUMENTAÇÃO EXPLICANDO O CÓDIGO COM SCREEN CAPTURE DO JOGO.

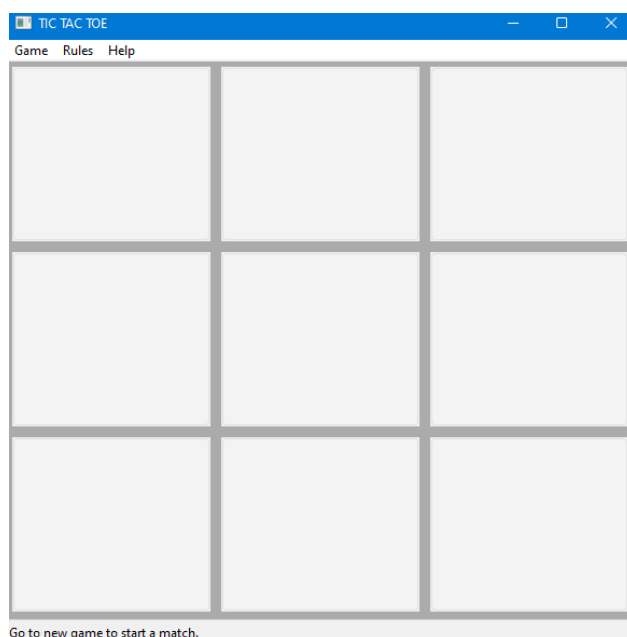
No "README.txt" incluso na pasta do jogo há a explicação das regras do jogo e explicações básicas sobre o trabalho, mas em resumo:

- O jogo foi criado em C, mas para fins gráficos importei uma biblioteca de C++, e, apenas para ela, usei POO (herança e polimorfismo) de C++, mas como visto durante todo o código, me absteve a utilizar apenas a linguagem C, visto que essa é a linguagem aprendida no período;
- O jogo da velha consiste em: O jogador que fizer uma sequência de 3 'X' ou 'O' seguidos, verticalmente, horizontalmente ou diagonalmente ganhará o jogo, caso ninguém faça isso o jogo será considerado empate, ou, como é popularmente conhecido no Brasil, o jogo dá "velha";
- Cada trecho do código eu explico o que venho fazendo no mesmo, para que servirá aquele trecho do código, e qual sua função no mesmo.



Acima está a tela inicial do jogo, coloquei amplamente para mostrar, na lateral, a quantidade de memória que o jogo está utilizando.

Abaixo apenas a tela do jogo:



Ao lado a tela inicial do jogo, o menu acima mostra funcionalidades do jogo, cada um com sub-abas, na aba "Games" há as seguintes abas:

- New Games (Ctrl + N);
- Scoreboard (Ctrl + S);
- Quit.

Já na aba "Rules", temos:

- Rules.

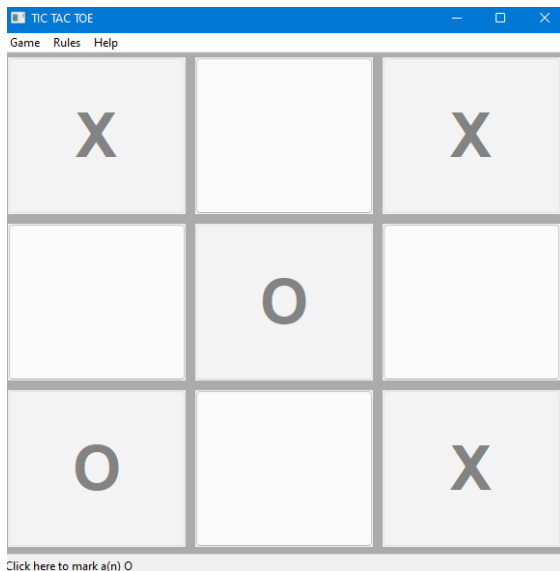
Na aba "Help", temos:

- About;
- Support.

O tabuleiro, no fundo, trabalha com números, 0 e 1, só que no front-end aparece o "X" e o "O".

Já abaixo aparece os eventos, no início ele fala para ir ao menu game e clicar no 'new game' para começar o jogo, quando se passa o mouse em cima dos sub-menus ele também aparece o que cada evento faz. Já quando começa o jogo, ele marca de quem é a vez, do "X" ou do "O".

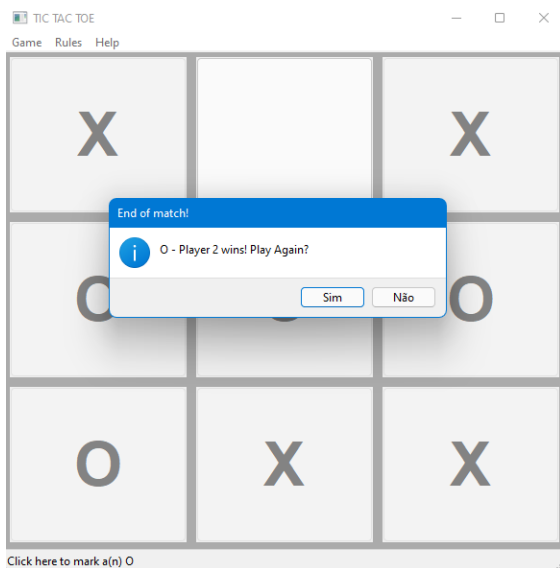
Assim fica o tabuleiro após algumas jogadas, visualmente falando:



Como dito acima, visualmente falando o tabuleiro aparece com "X" e "O", os botões que estão marcados e, conseqüentemente inativos, ficam mais escuros. No menu eventos abaixo, aparece de qual jogador é a vez.

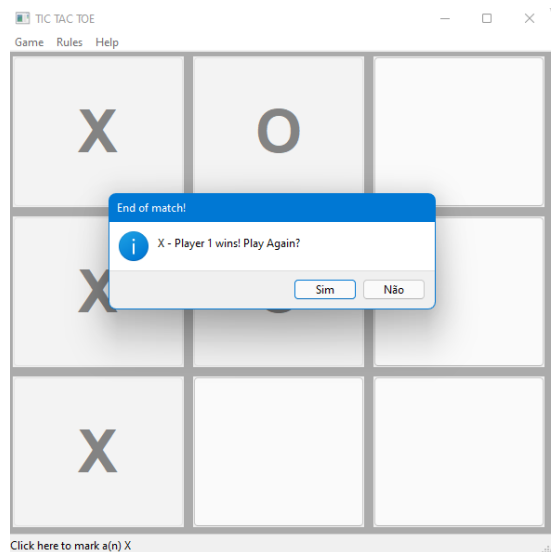
A cada rodada o código verifica se, em alguma parte do tabuleiro algum jogador ganha.

Abaixo as telas de vitória de cada jogador, com a opção de começar a partida de novo, caso os players queiram.

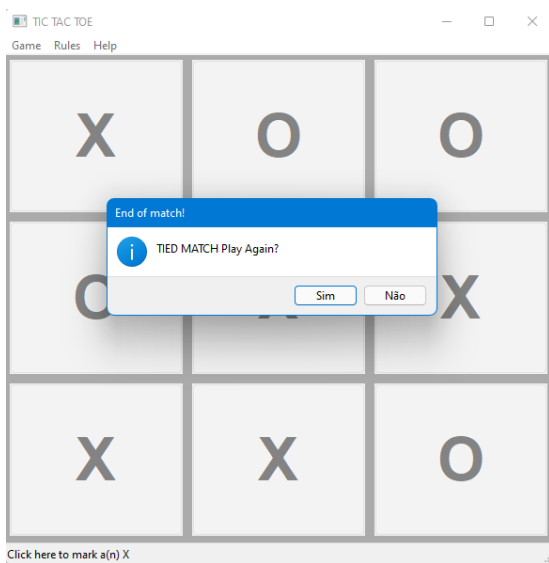


< Player
2 ganha

Player 1 >
ganha

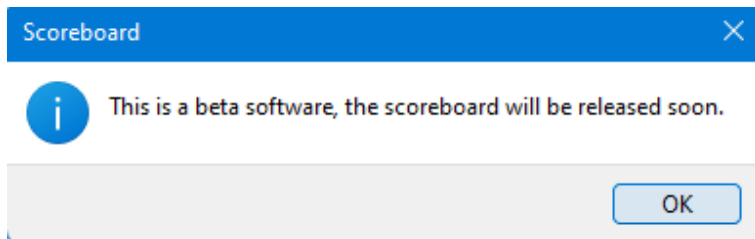


Em caso de empate:

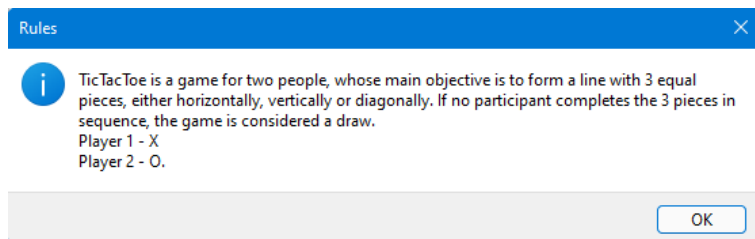


Diz que o jogo foi empate e pergunta se quer começar de novo.

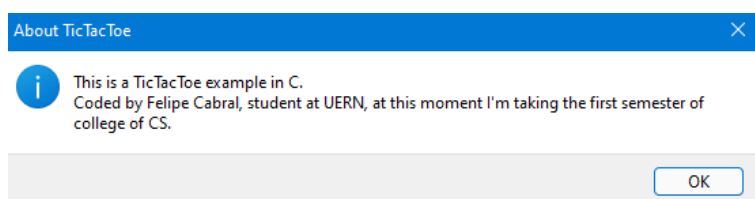
Abaixo os eventos dos sub-menus:



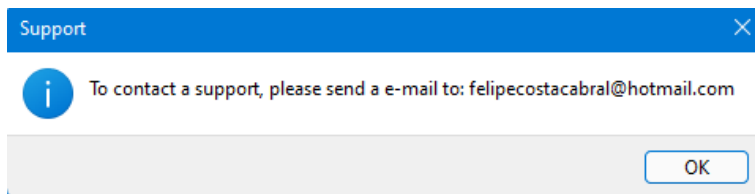
- Scoreboard: Deixei como beta, sem marcar, pois não consegui pensar em nenhuma forma de marcar uma pontuação.



- Rules: Explico as regras do jogo.



- About: Sobre o jogo, que é um jogo programado em C, e informações sobre o criador do jogo (eu).



- Support: Uma forma de entrar em contato caso o jogo apresente algum bug.

CONSIDERAÇÕES FINAIS:


Criei o jogo em inglês por conta do github, para melhorar a exposição futuramente. Mas, ao contrário do último jogo, não coloquei a explicação em inglês, senão o código ficaria ainda mais extenso. Mas, criarei uma documentação ainda mais extensa que o "README" futuramente, inteiramente em inglês e português.

Tive que aprender ainda mais sobre C e C++, entender toda a biblioteca do wxwidgets (que ainda não aprendi tudo) lendo a documentação, como importar e como ela servia. Foi duro no início, mas com o tempo fui aprendendo. Essa foi a parte do código que mais me trouxe dificuldade:

```
//Criando a parte que o usuário "enxerga" dos botões do tabuleiro
m_btnUpLeft = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnUpLeft->SetFont(font);
gBoard->Add(m_btnUpLeft, 0, wxALL | wxEXPAND, 5);
m_btnUpMid = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnUpMid->SetFont(font);
gBoard->Add(m_btnUpMid, 0, wxALL | wxEXPAND, 5);
m_btnUpRight = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnUpRight->SetFont(font);
gBoard->Add(m_btnUpRight, 0, wxALL | wxEXPAND, 5);

m_btnMidLeft = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnMidLeft->SetFont(font);
gBoard->Add(m_btnMidLeft, 0, wxALL | wxEXPAND, 5);
m_btnMidMid = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnMidMid->SetFont(font);
gBoard->Add(m_btnMidMid, 0, wxALL | wxEXPAND, 5);
m_btnMidRight = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnMidRight->SetFont(font);
gBoard->Add(m_btnMidRight, 0, wxALL | wxEXPAND, 5);

m_btnLowLeft = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnLowLeft->SetFont(font);
gBoard->Add(m_btnLowLeft, 0, wxALL | wxEXPAND, 5);
m_btnLowMid = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnLowMid->SetFont(font);
gBoard->Add(m_btnLowMid, 0, wxALL | wxEXPAND, 5);
m_btnLowRight = new wxButton(this, wxID_ANY, wxEmptyString, wxDefaultPosition, wxSize(40, 40), 0);
m_btnLowRight->SetFont(font);
gBoard->Add(m_btnLowRight, 0, wxALL | wxEXPAND, 5);
```

 `const wxPoint wxDefaultPosition`

[Pesquisar Online](#)