Trabalho de FIA
Turma 2023-2

Feito por: Gabriel Gonzaga Seabra Câmara
Matrícula: 21102526

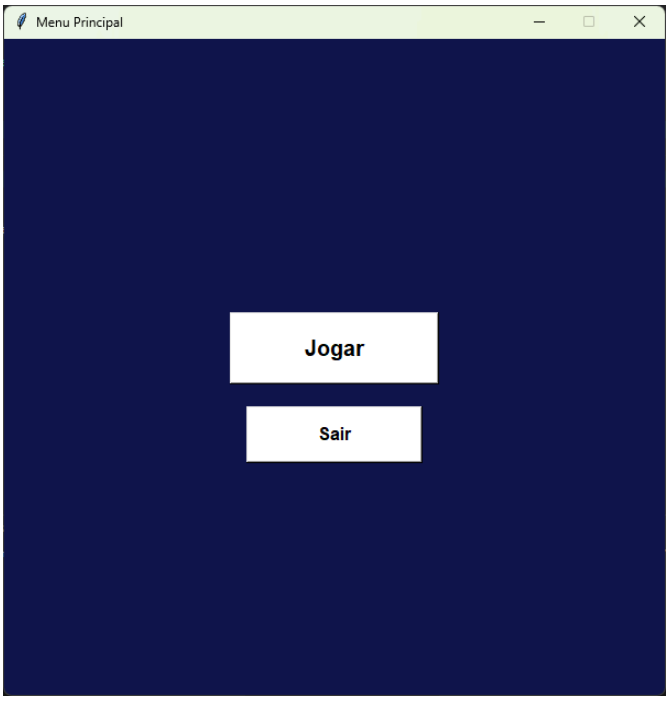
- ## Proposta

O trabalho proposto é uma representação do jogo de tabuleiro de 1965 BreakThru, entretanto com alterações em sua estrutura e regras:
-O tabuleiro, originalmente 11x11, foi adaptado para 7x7;
-As peças podem se mover somente 1 célula por turno;
-Só pode mover uma peça por vez;
-A posição inicial das peças no tabuleiro é fixa.

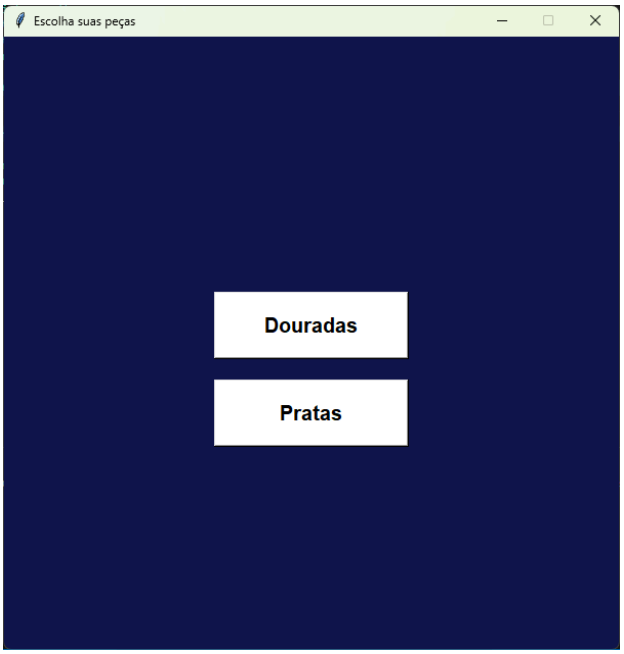
Além disso, foi solicitado que fosse implementado o algoritmo de busca MiniMax, com poda Alpha Beta para aumentar sua eficiência. Foi solicitado também que utilizasse duas diferentes heurísticas, colocando uma contra a outra para comparar suas eficiências.

- ## Código entregue

O código entregue cumpre com todos os requisitos solicitados acima. Ele é composto por somente um arquivo com todo o executável, portanto só precisa executá-lo diretamente em seu editor de texto. Sua tela inicial é um menu com 2 botões que dão ao usuário o direito de avançar no jogo ou encerrar o programa.



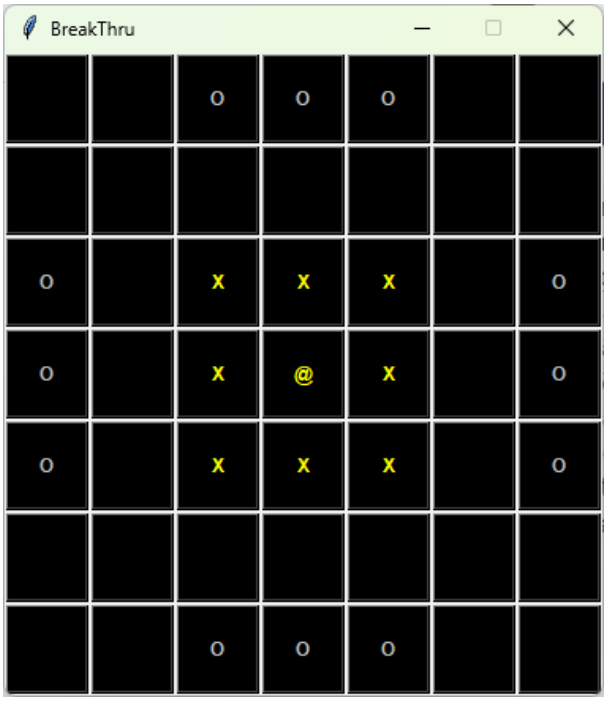
Em seguida, tendo o usuário clicado no botão “*Jogar*”, lhes será dado as opções de jogar com as peças pratas ou douradas, também representada por dois botões.



Independente da escolha, o tabuleiro se iniciará o mesmo.

No caso do usuário escolher as peças douradas, ele quem fará a primeira jogada. Para mover a peça, ele deve selecionar a peça que ele quer mover e em seguida a posição final que o usuário deseja. Caso seja feito uma tentativa de movimento ilegal, o movimento não será realizado e será notificado no terminal que o movimento não foi permitido.

No caso do usuário escolher representar as peças pratas, ele deve clicar em qualquer célula para iniciar o jogo (chamando o primeiro movimento da IA). Sua forma de movimentação será exatamente igual às peças douradas, apenas com objetivo e posições iniciais diferentes.



Na implementação, as heurísticas utilizadas foram a diferença na quantidade de peças e sua proximidade do flagship:

```
def number_piece(self):
    if self.game.Player_dourada:
            return len(__obj/self.game.posicoes_pratas)/self.INICIAL_SILVER - len(__obj/self.game.posicoes_douradas)/self.INICIAL_GOLD
    else:
            return len(__obj/self.game.posicoes_douradas)/self.INICIAL_GOLD - len(__obj/self.game.posicoes_pratas)/self.INICIAL_SILVER

def proximity_to_flagship(self):
    flagship_positions = [pos for pos in self.game.posicoes_douradas if self.game.tabuleiro[pos[0]][pos[1]]['text'] == "@"]
    min_distance = float(__x='inf')
    for silver_pos in self.game.posicoes_pratas:
            for flagship_pos in flagship_positions:
                distance = abs(__x/silver_pos[0] - flagship_pos[0]) + abs(__x/silver_pos[1] - flagship_pos[1])
                min_distance = min(__arg1/min_distance, __arg2/distance)
    if self.game.Player_dourada:
            return -min_distance
    else:
            return min_distance
```

Em ambos os casos, o objetivo da heurística é maximizar seu valor para a IA. No minimax aplicado eu não considerei “passos adiante” pois tive dificuldade na implementação de passar uma cópia do tabuleiro para fazer uma simulação dos movimentos previstos e optei por deixar de fora do código final da entrega. Entretanto, ele é capaz de olhar adiante dos seus próprios movimentos até a profundidade descrita na chamada da função “*turno_IA*”.

- Bibliografia

- https://www.youtube.com/watch?v=l-hh51ncgDI&t=557s&pp=ugMICgJwdBABGAHKBRptaW5pbWF4IGFscGhhIGJldGEgcHJ1bmluZw%3D%3D
- https://chat.openai.com
- https://en.wikipedia.org/wiki/Breakthru_(board_game)
- https://web.archive.org/web/20160303230420/http://www.gamepile.com/details.php?id=3
-