utad

Relatório Lab II

Senet

Realizado por:
Carolina Machado Al79359
David Fidalgo Al79881
Leonardo Salazar Al78473

Índice

Introdução

O objetivo deste projeto era criar um jogo. Das opções fornecidas pelo professor escolhemos o Senet.

Senet é um jogo Egito e dos mais antigos conhecidos da história. Curiosamente, nunca foram encontradas regras para o Senet, seja em papiros ou pintadas em paredes de túmulos. Acredita-se que pelo jogo ter sido tão popular, foi ensinado exclusivamente de um jogador para outro, porque de qualquer maneira quase todo mundo sabia como jogá-lo. Ainda assim, com base em pinturas do jogo em túmulos, nas referências feitas a ele na escrita egípcia e olhando para os seus descendentes modernos, historiadores criaram o que se acredita ser a mais próxima reconstrução das regras de Senet.

O jogo utiliza quatro varas de arremesso que têm a forma de meio cilindro, com a superfície arredondada pintada de preto e a superfície plana pintada de branco. O objetivo do jogo é lançar as varas e contar o número de brancos obtidos.(um resultado de "nenhum" é contado como cinco). O tabuleiro possui 6 casas especiais que dificultam o objetivo do jogo, que é retirar as suas peças do tabuleiro mais rápido que o adversário.

Bibliotecas utilizadas

Para este trabalho utilizamos as seguintes bibliotecas para nos auxiliarem:

- Tkinter
- Random
- Pygame

A biblioteca Tkinter e a pygame são duas bibliotecas gráficas que nos foram uteis na realização da interface. Já a biblioteca Random ajudou-nos na logica do jogo.

Estrutura do código

Ficheiro “jogo.py”

```
def menu():
    global estado_jogo
    pygame.init()
    imagemfundo = pygame.image.load("imagemfundo.jpeg")
    compri = imagemfundo.get_width()
    larg = imagemfundo.get_height()
    janela = pygame.display.set_mode((compri - 1, larg - 1))
    BRANCO = (255, 255, 255)
    PRETO = (0, 0, 0)
    CINZA = (200, 200, 200)
    VERDE = (0, 255, 0)

    def criar_botao(texto, cor_normal, cor_hover, posicao, largura, altura, acao):
        retangulo = pygame.Rect(posicao[0], posicao[1], largura, altura)

        cor = cor_normal
        mouse_pos = pygame.mouse.get_pos()
        if retangulo.collidepoint(mouse_pos):
            cor = cor_hover
            if pygame.mouse.get_pressed()[0] == 1:
                acao()

        pygame.draw.rect(janela, cor, retangulo)
        texto_surface = pygame.font.Font(None, 40).render(texto, True, PRETO)
        texto_rect = texto_surface.get_rect(center=retangulo.center)
        janela.blit(texto_surface, texto_rect)
```

```
def novojogo():
    def intrnome():
        def segundoplay(entrada):
            def fechar():
                jogadoresd['nome2'] = entrada.get()
                jogadores.append(jogadoresd)
                janela_nome2.destroy()

                global estado_jogo
                estado_jogo = False


            jogadoresd['nome1'] = entrada.get()

            janela_nome1.destroy()
            janela_nome2 = tk.Tk()
            janela_nome2.title("JOGADOR 2")
            janela_nome2.geometry("400x200")
            label = tk.Label(janela_nome2, text ="  Introduza o seu nome:  ")
            label.pack()
            entrada = tk.Entry(janela_nome2)
            entrada.pack()
            botao = tk.Button(janela_nome2, text ="   OK   ", command=fechar)
            botao.pack()
            label_nome = tk.Label(janela_nome2, text ="")
            label_nome.pack()
            janela_nome2.mainloop()
```

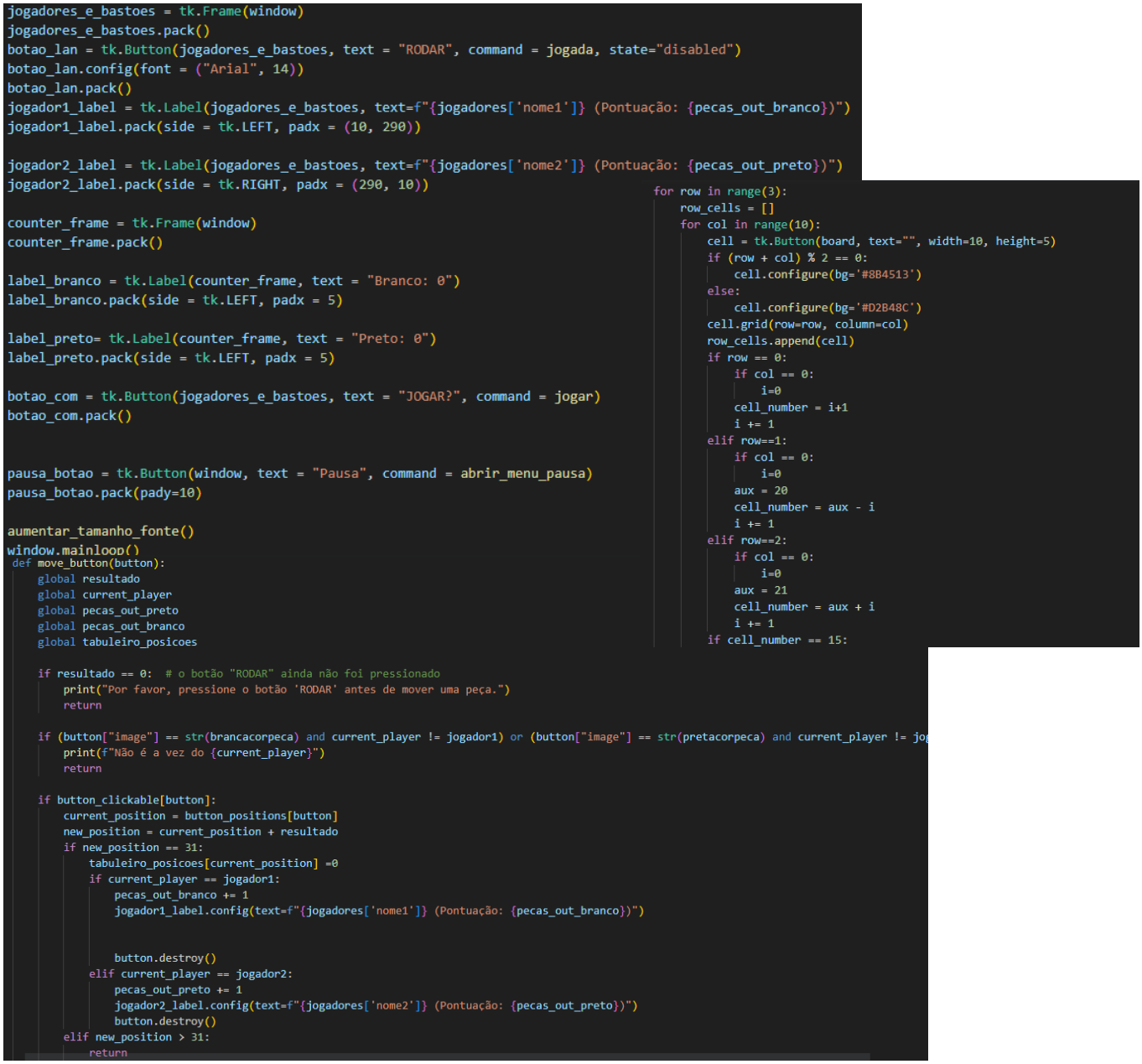
```
def regrasjogo():
    def fechajanela():
        janela_regras.destroy()
    janela_regras = tk.Tk()
    janela_regras.title ("Regras SENET")
    janela_regras.geometry ("1280x1024")
    frase1 = tk.Label (janela_regras, text = "O jogo utiliza quatro varas de arremesso que têm a forma de meio cilindro, com a \n superfície arr
    frase1.pack()
    frase2 = tk.Label (janela_regras, text = "\nQuando não é possível mover nenhuma peça em qualquer direção, o jogador deve \n retroceder 5 cas
    frase2.pack()
    frase3 = tk.Label (janela_regras, text = "\nO objetivo final é ser o primeiro jogador a retirar todas as suas peças do tabuleiro.", font = (
    frase3.pack()
    frasef= tk.Label (janela_regras, text = "")
    frasef.pack()
    botaoregra = tk.Button(janela_regras, text = "          OK          ", command = fechajanela)
    botaoregra.pack ()
    janela_regras.mainloop()
```

Figura 1: Parte do código do ficheiro “Jogo”

No ficheiro “Jogo” utilizamos duas bibliotecas o pygame e o tkinter. O pygame auxiliou-nos na criação da primeira janela do jogo que é o Menu onde temos 4 botões,

um para iniciar um novo jogo, “NOVO JOGO”, um para carregar um jogo já existente, “CARREGAR JOGO” que não esta disponível uma vez que não o conseguimos por a funcionar, “REGRAS” que quando clicamos aparece uma segunda janela com um texto a explicar o jogo para os utilizadores e por fim temos o botão “SAIR” para fecharmos o jogo. Já o tkinter foi o que nos auxiliou a criar todas as janelas segundarias do menu, como as das regras e a janela para digitar o nome dos jogadores(abre quando clicamos no botão “NOVO JOGO”).

Ficheiro “tabuleiro.py”



```
jogadores_e_bastoes = tk.Frame(window)
jogadores_e_bastoes.pack()
botao_lan = tk.Button(jogadores_e_bastoes, text = "RODAR", command = jogada, state="disabled")
botao_lan.config(font = ("Arial", 14))
botao_lan.pack()
jogador1_label = tk.Label(jogadores_e_bastoes, text=f"{jogadores['nome1']} (Pontuação: {pecas_out_branco})")
jogador1_label.pack(side = tk.LEFT, padx = (10, 290))

jogador2_label = tk.Label(jogadores_e_bastoes, text=f"{jogadores['nome2']} (Pontuação: {pecas_out_preto})")
jogador2_label.pack(side = tk.RIGHT, padx = (290, 10))

counter_frame = tk.Frame(window)
counter_frame.pack()

label_branco = tk.Label(counter_frame, text = "Branco: 0")
label_branco.pack(side = tk.LEFT, padx = 5)

label_preto= tk.Label(counter_frame, text = "Preto: 0")
label_preto.pack(side = tk.LEFT, padx = 5)

botao_com = tk.Button(jogadores_e_bastoes, text = "JOGAR?", command = jogar)
botao_com.pack()


pausa_botao = tk.Button(window, text = "Pausa", command = abrir_menu_pausa)
pausa_botao.pack(pady=10)

aumentar_tamanho_fonte()
window.mainloop()
```

```
for row in range(3):
    row_cells = []
    for col in range(10):
        cell = tk.Button(board, text="", width=10, height=5)
        if (row + col) % 2 == 0:
            cell.configure(bg='#8B4513')
        else:
            cell.configure(bg='#D2B48C')
        cell.grid(row=row, column=col)
        row_cells.append(cell)
        if row == 0:
            if col == 0:
                i=0
            cell_number = i+1
            i += 1
        elif row==1:
            if col == 0:
                i=0
            aux = 20
            cell_number = aux - i
            i += 1
        elif row==2:
            if col == 0:
                i=0
            aux = 21
            cell_number = aux + i
            i += 1
        if cell_number == 15:
```

```
def move_button(button):
    global resultado
    global current_player
    global pecas_out_preto
    global pecas_out_branco
    global tabuleiro_posicoes

    if resultado == 0:  # o botão "RODAR" ainda não foi pressionado
        print("Por favor, pressione o botão 'RODAR' antes de mover uma peça.")
        return

    if (button["image"] == str(brancacorpeca) and current_player != jogador1) or (button["image"] == str(pretacorpeca) and current_player != jog
        print(f"Não é a vez do {current_player}")
        return

    if button_clickable[button]:
        current_position = button_positions[button]
        new_position = current_position + resultado
        if new_position == 31:
            tabuleiro_posicoes[current_position] =0
            if current_player == jogador1:
                pecas_out_branco += 1
                jogador1_label.config(text=f"{jogadores['nome1']} (Pontuação: {pecas_out_branco})")


                button.destroy()
            elif current_player == jogador2:
                pecas_out_preto += 1
                jogador2_label.config(text=f"{jogadores['nome2']} (Pontuação: {pecas_out_preto})")
                button.destroy()
        elif new_position > 31:
            return
```

Figura 2: Parte do código do ficheiro “Tabuleiro”

No ficheiro “Tabuleiro” usamos a unicamente a biblioteca tkinter para criar o tabuleiro de jogo. Neste cada quadrado e cada peça são um botão, para alem deste temos mais dois botões, o “JOGAR?” para iniciar o jogo e o “RODAR” para rodar os bastões.

Temos também os nomes dos 2 jogares e a sua pontuação (neste contexto a pontuação diz-nos o número de peças de cada jogador que já estão fora do tabuleiro) por último colocamos a identificação do resultado do lançamento dos bastões, identificando o número de lados que saíram brancos e pretos.

Ficheiro “regras.py”

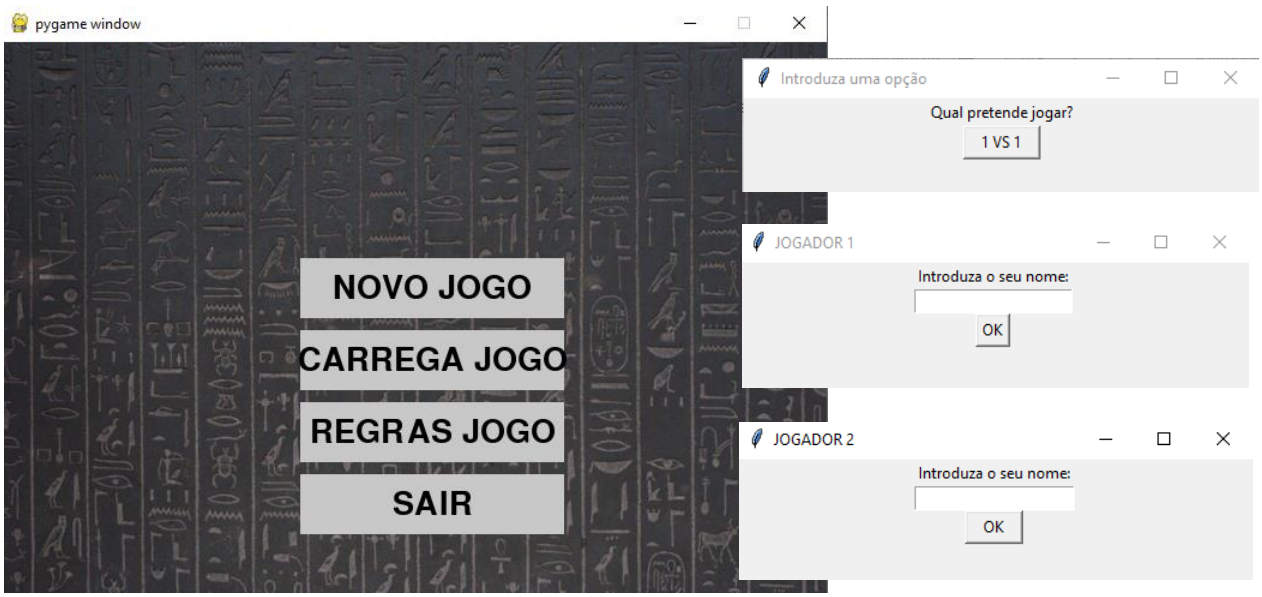
```
bastão = 0
ladobranco = 0
ladomadeira = 0
peao = 0
resultado=0
resultado_bastao = 0
def regras():
    global resultado
    bastão = 0
    ladobranco = 0
    ladomadeira = 0
    peao = 0

    for j in range(1, 5):
        bastão = random.randint(1, 2)
        if bastão == 1:
            print("bastao", j, "-> Lado branco")
            ladobranco = ladobranco + 1



        elif bastão == 2:
            print("bastao", j, "-> Lado de madeira")
            ladomadeira = ladomadeira + 1

    if ladobranco == 0:
        print("  *saiu 0 lados brancos")
        print("  *sairam 4 lados de madeira")
        print("Mova 5 quadrados e ganha uma jogada extra\n")
        peao += 5
    elif ladobranco == 1:
        print("  *saiu 1 lado branco")
        print("  *sairam 3 lados de madeira")
        print("Mova 1 quadrado e ganha uma jogada extra\n")
        peao += 1
    elif ladobranco == 2:
        print("  *sairam 2 lados brancos")
        print("  *sairam 2 lados de madeira")
        print("Mova 2 quadrados\n")
        peao += 2
    elif ladobranco == 3:
        print("  *sairam 3 lados brancos")
        print("  *sairam 1 lados de madeira")
        print("Mova 3 quadrados\n")
        peao += 3
    elif ladobranco == 4:
        print("  *saiu 4 lados brancos")
        print("  *saiu 0 lado de madeira\n")
        print("Mova 4 quadrados e ganha uma jogada extra")
        peao += 4

    if ladobranco == 0:
        resultado =5
        return ladobranco, ladomadeira, resultado
    else:
        resultado = ladobranco
        return ladobranco, ladomadeira, resultado


def verificapeca(dic_posicao, play, posicaopeca, lancamento, passo):
    if passo ==1:
        if play == "Jogador1" and posicaopeca+lancamento != 27:
            if dic_posicao[posicaopeca+lancamento] == 2:
                return -1
            elif dic_posicao[posicaopeca+lancamento] == 1:
                return 0
            elif dic_posicao[posicaopeca+lancamento] == 0:
                return 1
        if play == "Jogador2" and posicaopeca+lancamento != 27 :
            if dic_posicao[posicaopeca+lancamento] == 2:
                return 0
            elif dic_posicao[posicaopeca+lancamento] == 1:
                return -1
            elif dic_posicao[posicaopeca+lancamento] == 0:
                return 1
        if posicaopeca+lancamento == 27:
            return 2
    if passo == 2:
        if dic_posicao[15] == 1 or dic_posicao[15] == 2:
            return 2.1
        elif dic_posicao[15] == 0:
            return 2.2
```

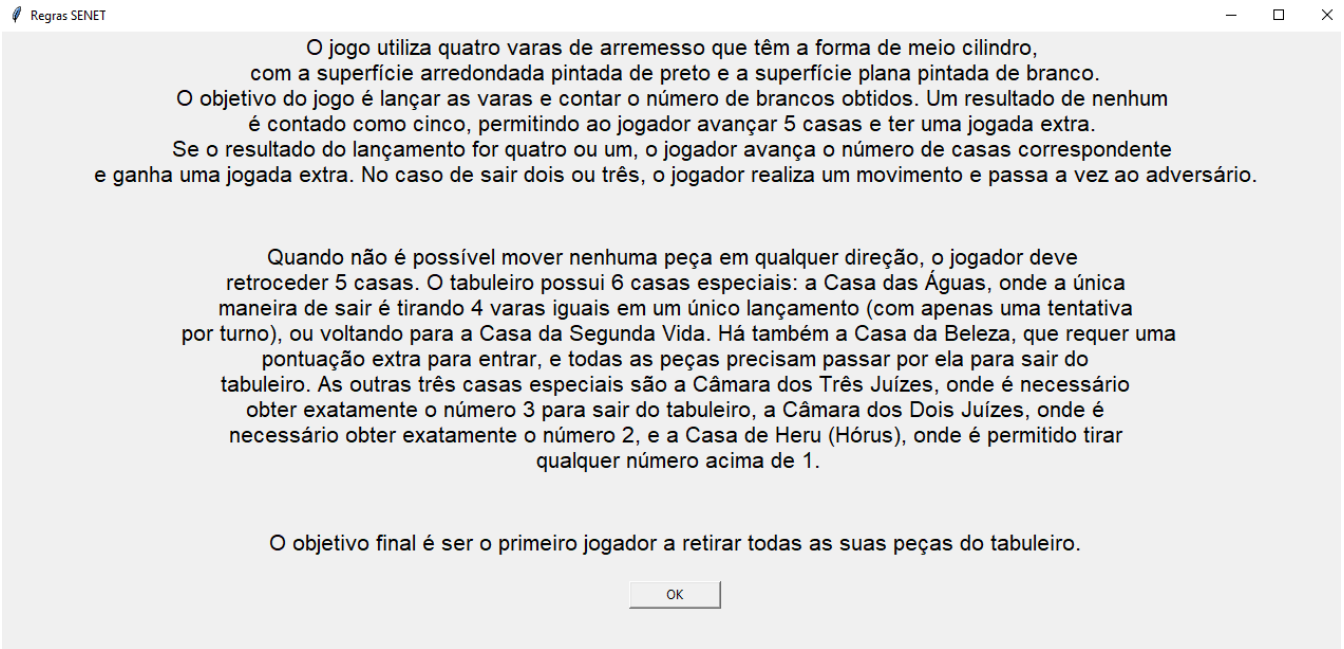
Figura 3: Código do ficheiro “Regras”

No ficheiro “Regras” foi onde colocamos a logica do jogo, utilizamos a biblioteca Random para os valores dos bastões, depois foi onde também criamos função para conseguirmos mover os peões tendo em conta a posição dos outros no tabuleiro.

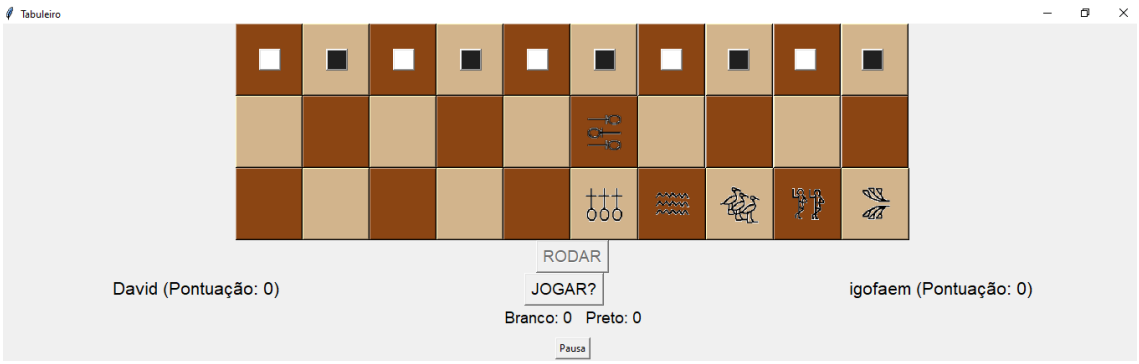
Resultados

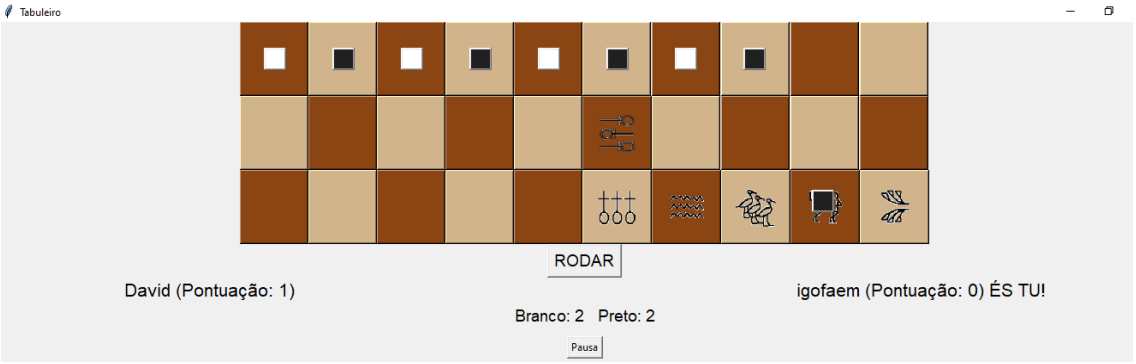
Ficheiro “jogo.py”





Ficheiro “tabuleiro.py”





Ficheiro “regras.py”

```
bastao 1 -> Lado branco
bastao 2 -> Lado branco
bastao 3 -> Lado de madeira
bastao 4 -> Lado de madeira
  *sairam 2 lados brancos
  *sairam 2 lados de madeira
Mova 2 quadrados
```

Conclusão

Por fim, este projeto ofereceu nos uma oportunidade de aplicar e aprimorar nosso conhecimento sobre desenvolvimento de software, especialmente no campo da criação de jogos. Desde a conceção do jogo, design e programação até a depuração e teste final. Foi necessário trabalhar em equipa e colaborar de forma eficaz, uma vez que cada membro do grupo de trabalho trouxe habilidades e perspetivas únicas para a mesa.

O jogo não ficou na totalidade dentro das nossas perspetivas, ambicionávamos um melhor design e as jogabilidades totalmente operáveis. As partes mais importantes do jogo então todas operacionais. Devido a alguma dificuldade para gerenciar o tempo dado, não foi conseguido alcançar todos os objetivos que nos foram propostos.