Jogo do Semáforo

LEI | Laboratório de Programação

Projeto realizado por:

AI78474 | Francisco Ruano

AI78686 | Pedro Pereira

AI78282 |Rui Pedro Madureira

Índice

1. Introdução

Este relatório discute os detalhes do desenvolvimento de um jogo em Python, abordando as principais questões técnicas, problemas encontrados e soluções encontradas. O jogo em questão foi desenvolvido com o objetivo de desenvolver habilidades de programação e demonstrar a capacidade de aplicar conceitos de lógica, estrutura de dados e algoritmos no desenvolvimento de software.

Foi-nos proporcionada uma lista de projetos/jogos como opções. Dentro destas decidimos escolher o jogo “Semáforo” devido a acharmos uma boa opção para explorarmos e testarmos o nosso conhecimento no desenvolvimento de jogos utilizando a linguagem de programação *Python* e as suas bibliotecas relacionadas ao desenvolvimento de software de jogos como por exemplo o *Pygame*.

Após uma leitura cuidadosa do protocolo fornecido pelos professores da UC, examinamos as instruções detalhadas sobre o que deveríamos fazer, bem como os procedimentos necessários e as regras do jogo. Como este jogo não era conhecido por nenhum dos membros do grupo de maneira a familiarizarmo-nos instalamos nos nossos dispositivos móveis aplicações com este jogo (Jogo do Semáforo, *PlayStore*).

Depois de conhecermos o jogo e saber as suas regras começamos por pensar em como iriamos criar o código e distribuímos tarefas entre os membros do grupo. Pesquisamos também no *Youtube* diversos vídeos sobre a biblioteca *Pygame* para aprendermos como a utilizar de maneira eficiente e prática de maneira a podermos utilizá-la no código do jogo.

2. Primeira abordagem

Após uma leitura cuidadosa do protocolo fornecido pelos professores da UC, examinamos as instruções detalhadas sobre o que deveríamos fazer, bem como os procedimentos necessários e as regras do jogo. Como este jogo não era conhecido por nenhum dos membros do grupo de maneira a familiarizarmo-nos instalamos nos nossos dispositivos móveis aplicações com este jogo (Jogo do Semáforo, *PlayStore*).

Depois de conhecermos o jogo e saber as suas regras começamos por pensar em como iriamos criar o código e distribuímos tarefas entre os membros do grupo. Pesquisamos também no *Youtube* diversos vídeos sobre a biblioteca *Pygame* para aprendermos como a utilizar de maneira eficiente e prática de maneira a podermos utiliza-la no código do jogo.

3. Interface Gráfica

Utilizamos imagens geradas por uma *AI* que acedemos através do site *Lexica* (Lexica.art) para utilizarmos de fundo nas diversas telas de fundo utilizadas no jogo.

A biblioteca *Pygame* foi bastante útil para a renderização destas imagens através da função *pygame.image.load*, para a redimensionar as imagens à medida da tela (1280 x 720) usamos a função *pygame.transform.scale*. Usamos também a função *screen.blit()* para renderizar na tela diversos elementos em coordenadas específicas.

De maneira adicionarmos os elementos à tela do jogo usamos a função *screen = pygame.display.set_mode((width, height))*.

No final de cada função, usamos a função pygame.display.update() para atualizar toda a área da tela com todos os ajustes feitos desde a atualização mais recente.

4. Bibliotecas utilizadas

Utilizamos 3 bibliotecas importadas na realização deste jogo sendo estas: *pygame*, *random* e *time*.

```
import pygame
import random
import time
```

5. Menu

pygame.init(): Inicializa os módulos necessários do Pygame.

screen = pygame.display.set_mode((width, height)): Cria uma janela de exibição com a largura e altura especificadas.

pygame.display.set_caption(title): Define o título da janela do jogo.

fundo = pygame.image.load('imagem/bg.png').convert_alpha(): Carrega uma imagem de fundo do arquivo "bg.png" e converte-a para o formato adequado para o Pygame.

fundo = pygame.transform.scale(fundo, (width, height)): Redimensiona a imagem de fundo para o tamanho da janela de exibição.

rodando = True: Variável que controla a execução contínua do loop principal.

fonte_botoes = pygame.font.Font(None, 35): Carrega uma fonte para os botões do jogo. O parâmetro "None" indica que será usada a fonte padrão do sistema com tamanho

sair = False: Variável que será usada para controlar o retorno da função.

Aqui temos o loop principal:

while rodando:: O loop continuará a ser executado enquanto a variável rodando for verdadeira.

for event in pygame.event.get():: Obtém todos os eventos que ocorreram desde a última iteração do loop.

if event.type == pygame.QUIT:: Verifica se o evento atual é o de fechar a janela do jogo. Se for, o Pygame é encerrado (pygame.quit()) e a variável sair é definida como True, indicando que o jogo deve ser encerrado corretamente.

Verifica se o evento é um clique do rato (event.type == pygame.MOUSEBUTTONDOWN) e se o botão pressionado é o botão esquerdo (event.button == 1).

posicao_mouse = pygame.mouse.get_pos(): Obtém a posição atual do rato.

Verifica se o botão "Iniciar" foi pressionado (botao_iniciar.collidepoint(posicao_mouse)). Se for, são executadas diferentes ações dependendo do tipo de jogador selecionado:

Se o tipo de jogador for 1, é solicitado o nome do jogador 1 (obter_nome_jogador) e do jogador 2 (obter_nome_jogador). Em seguida, é chamada a função jogar_semaforo para iniciar o jogo com os nomes dos jogadores.

Se o tipo de jogador for 2, o jogador 2 é definido como "Bot" e a função jogar_semaforo_bot é chamada para iniciar o jogo contra um bot.

Se o tipo de jogador for 3, a função jogar_bot_medio é chamada para iniciar o jogo contra um bot de nível médio.

Se o tipo de jogador for 4, a função jogar_bot_dificil é chamada para iniciar o jogo contra um bot de nível difícil.

6. Escolher_tipo_jogador

A função escolher_tipo_jogador permite ao jogador selecionar o tipo de jogo desejado.

A função escolher_tipo_jogador recebe três parâmetros: width (largura da tela), height (altura da tela) e screen (objeto da tela do jogo).

Carrega uma imagem de fundo a partir do arquivo 'imagem/bg1.png' e converte para o formato apropriado (convert_alpha). Em seguida, redimensiona a imagem para a largura e altura especificadas pelos parâmetros width e height, respectivamente.

Inicializa uma variável booleana rodando como True. Esta variável será usada para controlar o loop principal do programa.

Inicializa a variável tipo_jogador como None. Essa variável será usada para armazenar o tipo de jogador selecionado pelo usuário.

Define a fonte do texto que será exibido na tela. Neste caso, a fonte padrão será usada com um tamanho de 36 pixels.

Define a área dos botões na tela. Estes botões serão usados para permitir que o jogador escolha o tipo de jogo.

Inicia o loop principal do programa. Este loop é responsável por lidar com os eventos (como cliques do rato) e atualizar o estado do jogo.

Verifica se algum evento ocorreu. Se o evento for do tipo QUIT (o usuário fechou a janela do jogo), o programa é encerrado.

A função também verifica se ocorreu um clique do rato com o botão esquerdo (MOUSEBUTTONDOWN com event.button == 1). Se ocorreu, a posição do rato é obtida (pygame.mouse.get_pos()).

Verifica se a posição do clique está dentro da área de algum dos botões. Se o clique ocorreu dentro do botão botao_jogador1vsjogador2, o valor de tipo_jogador é definido como 1. Se ocorreu dentro do botão botao_jogador1vsbot, tipo_jogador é definido como 2. Se ocorreu dentro do

botão botao_bot_medio, tipo_jogador é definido como 3. Se ocorreu dentro do botão botao_bot_dificil, tipo_jogador é definido como 4.

Se algum botão foi clicado, a variável tipo_jogador é definida e o loop é interrompido, encerrando a função escolher_tipo_jogador.

Resumindo, esta função cria uma tela com botões para permitir que o jogador escolha entre diferentes tipos de jogadores para o jogo. Dependendo do botão clicado, o valor de tipo_jogador é definido para um número específico, indicando o tipo de jogo escolhido.

7. obter_nome_jogador

Desenha o texto na tela. Se o tipo_jogador for igual a 1 (Jogador vs Jogador), o texto será "Digite o nome do Jogador 1:", caso contrário, será "Digite o nome do Jogador:". O número do jogador é convertido em string e adicionado ao texto. O texto é renderizado com a fonte e cor especificadas, e desenhado na posição adequada na tela.

Desenha o campo de texto na tela. Um retângulo é desenhado na posição e tamanho especificados pela variável campo_texto, com uma cor de contorno branca.

Renderiza o texto digitado pelo jogador usando a fonte e cor especificadas. O texto é desenhado dentro do campo de texto, com uma margem de 5 pixels em relação às bordas do retângulo.

A tela é atualizada para exibir as mudanças feitas.

Retorna o valor da variável nome_jogador. Esse valor conterá o nome introduzido pelo jogador.

Em suma, esta função cria uma tela com um campo de texto onde o jogador pode digitar o seu nome. Quando o jogador pressiona a tecla Enter, o loop é interrompido e o nome digitado é retornado pela função.

8. fazer_jogada

A função fazer_jogada recebe três parâmetros: tabuleiro (uma matriz que representa o tabuleiro do jogo), linha (a linha em que o jogador deseja fazer a jogada) e coluna (a coluna em que o jogador deseja fazer a jogada).

A função verifica se a posição (linha, coluna) está dentro dos limites do tabuleiro. Se a posição estiver fora dos limites, imprime uma mensagem de erro e retorna False, indicando que a jogada é inválida.

Inicializa a variável booleana movimento_possível como False. Esta variável será usada para controlar o loop que verifica se o movimento é possível.

Inicia um loop que executa até que um movimento seja possível. O loop verifica o valor da posição (linha, coluna) no tabuleiro.

Se a posição contiver o valor 3, isso significa que a casa já não tem mais jogadas possíveis. Nesse caso, imprime uma mensagem de erro e retorna False, indicando que a jogada é inválida.

Se a posição contiver o valor 1, significa que é uma jogada válida do jogador 1. Altera o valor da posição para 2, indicando que o jogador 1 fez a jogada, e define movimento_possível como True para sair do loop.

Se a posição contiver o valor 2, significa que é uma jogada válida do jogador 2. Altera o valor da posição para 3, indicando que o jogador 2 fez a jogada, e define movimento_possível como True para sair do loop.

Se a posição contiver qualquer outro valor, assume-se que é uma jogada vazia. Altera o valor da posição para 1, indicando que o jogador 1 fez a jogada, e define movimento_possível como True para sair do loop.

Após sair do loop, a função retorna True, indicando que a jogada foi feita com sucesso.

Esta função permite que uo jogador faça uma jogada no tabuleiro, atualizando o estado do tabuleiro de acordo com as regras do jogo.

9. verificar_vencedor

A função verificar_vencedor verifica se há um vencedor no jogo representado pelo tabuleiro fornecido como argumento.

Itera sobre as linhas do tabuleiro (índices 0, 1 e 2). Verifica se há três valores iguais nessas linhas consecutivas, excluindo 0. Se essa condição for verdadeira, significa que um jogador venceu e a função retorna True

Itera sobre as linhas do tabuleiro (índices 0, 1 e 2) novamente. No entanto, agora verifica se há três valores iguais nessas linhas consecutivas, começando a partir do índice 1. Isso ocorre porque o código anterior já verificou a sequência de três valores consecutivos começando do índice 0. Novamente, se essa condição for verdadeira, um jogador venceu e a função retorna True.

Repete sobre as colunas do tabuleiro (índices 0, 1, 2 e 3). Verifica se há três valores iguais nessas colunas consecutivas, excluindo 0. Se essa condição for verdadeira, um jogador venceu e a função retorna True.

Verifica a diagonal principal (da posição [0][0] à [2][2]). Se os valores nessas posições forem iguais e diferentes de 0, um jogador venceu e a função retorna True.

Verifica uma diagonal secundária (da posição [0][1] à [2][3]). Se os valores nessas posições forem iguais e diferentes de 0, um jogador venceu e a função retorna True.

Verifica a diagonal invertida da diagonal secundária (da posição [0][3] à [2][1]). Se os valores nessas posições forem iguais e diferentes de 0, um jogador venceu e a função retorna True.

Verifica a outra diagonal invertida da diagonal principal (da posição [0][2] à [2][0]). Se os valores nessas posições forem iguais e diferentes de 0, um jogador venceu e a função retorna True.

Verifica a última condição repetida (da posição [0][2] à [2][0]). Se os valores nessas posições forem iguais e diferentes de 0, um jogador venceu e a função retorna True.

Se nenhuma das condições acima for atendida, isso significa que não há um vencedor e a função retorna False.

10. pontos_a_zero

Define a variável pontos_1 como zero para representar os pontos do jogador 1.

Define a variável pontos_2 como zero para representar os pontos do jogador 2.

Retorna uma tupla contendo os valores de pontos_1 e pontos_2.

Esta função é usada para reiniciar os pontos dos jogadores, definindo-os como zero no início de um novo jogo ou em alguma outra situação necessária.

11. atualizar_pontuacao

texto_pontuacao1 é uma variável que armazena o texto renderizado da pontuação do jogador1. O texto é gerado usando a fonte_pontuacao fornecida como argumento, concatenando o nome do jogador (jogador1) e sua pontuação atual (pontos_1).

texto_pontuacao2 é uma variável que armazena o texto renderizado da pontuação do jogador2. O texto é gerado da mesma maneira que o texto_pontuacao1, mas usando o nome do jogador (jogador2) e sua pontuação atual (pontos_2).

Retorna uma tupla contendo os textos renderizados texto_pontuacao1 e texto_pontuacao2.

Essa função é usada para atualizar a exibição da pontuação na tela do jogo, com base nas pontuações dos jogadores. Os textos renderizados podem ser usados posteriormente para desenhar a pontuação atualizada na interface do jogo.

12. somar_pontos

A função somar_pontos recebe como entrada o número do jogador que marcou um ponto, bem como as pontuações atuais dos jogadores. Em seguida, ela incrementa a pontuação correspondente ao jogador e retorna as pontuações atualizadas.

Verifica se o jogador é igual a 1. Se for o caso, significa que o jogador 1 marcou um ponto.

Incrementa a variável pontos_1 em 1 para aumentar a pontuação do jogador 1.

Se o jogador não for igual a 1, verifica se é igual a 2. Se for o caso, significa que o jogador 2 marcou um ponto.

Incrementa a variável pontos_2 em 1 para aumentar a pontuação do jogador 2.

Retorna uma tupla contendo as pontuações atualizadas pontos_1 e pontos_2.

13. alternar_jogador

Verifica se o jogador_atual é igual a 1. Se for o caso, significa que o jogador atual é o jogador 1.

Retorna 2, indicando que o próximo jogador será o jogador 2.

Se o jogador_atual não for igual a 1, significa que o jogador atual é o jogador 2.

Retorna 1, indicando que o próximo jogador será o jogador 1.

Esta função é usada para alternar entre os jogadores durante o jogo, permitindo que cada jogador tenha a sua vez de fazer uma jogada. O número retornado representa o próximo jogador que deve realizar uma ação.

14. jogar_outra_vez

Define uma fonte para o texto que será exibido na tela.

Define as cores do texto e dos botões.

Define as dimensões e posições dos botões "Sim" e "Não".

Entra em um loop infinito que processa os eventos da janela do jogo.

Verifica se o evento é do tipo "QUIT" (fechar a janela). Se for, encerra o jogo.

Verifica se o evento é do tipo "MOUSEBUTTONDOWN" (clique do rato).

Obtém a posição do clique do rato.

Verifica se o clique ocorreu dentro da área do botão "Sim". Se sim, retorna True para indicar que os jogadores desejam jogar novamente.

Verifica se o clique ocorreu dentro da área do botão "Não". Se sim, retorna False para indicar que os jogadores não desejam jogar novamente.

Limpa a tela preenchendo-a com a cor preta.

Desenha os botões "Sim" e "Não" na tela.

Desenha o texto nos botões.

Atualiza a tela para exibir os desenhos feitos.

O loop continua até que os jogadores façam uma escolha.

Esta função é usada para exibir uma tela de confirmação no final de uma partida, permitindo que os jogadores decidam se desejam jogar novamente ou encerrar o jogo. A resposta retornada é utilizada para determinar o próximo curso de ação no programa principal.

A função jogar_outra_vez exibe uma tela com dois botões ("Sim" e "Não") e espera pela resposta dos jogadores. Se o jogador clicar no botão "Sim", a função retorna True, indicando que eles desejam jogar novamente. Se o jogador clicar no botão "Não", a função retorna False, indicando que eles não desejam jogar novamente. A função continua em um loop até que os jogadores façam uma escolha.

15. quem_joga

Verifica se o jogador_atual é igual a 1. Se for o caso, significa que o jogador atual é o jogador 1.

Cria um texto renderizado usando a fonte, fonte_jogadores com o número do jogador atual.

Se o jogador_atual não for igual a 1, significa que o jogador atual é o jogador 2.

Cria um texto renderizado usando a fonte fonte_jogadores com o número do jogador atual.

Retorna o texto renderizado do jogador atual e a fonte de renderização dos jogadores.

Esta função é usada para obter o texto renderizado que representa o jogador atual. O número do jogador atual é usado para criar o texto renderizado. A fonte de renderização dos jogadores é retornada junto com o texto renderizado para ser usada em outros lugares do código.

16. desenhar_tabuleiro

screen.blit(fundo, (0, 0)): Desenha o fundo na tela.

screen.blit(texto_pontuacao1, (width - largura_texto_pontuacao1 - 10, 10)): Desenha o texto da pontuação do jogador 1 na tela.

screen.blit(texto_pontuacao2, (width - largura_texto_pontuacao2 - 10, 10 + altura_texto_pontuacao + 5)): Desenha o texto da pontuação do jogador 2 na tela.

screen.blit(texto_jogador1, (width - largura_texto_jogador1 - 10, 10 + altura_texto_pontuacao * 2)): Desenha o texto do jogador 1 na tela.

screen.blit(texto_jogador2, (width - largura_texto_jogador2 - 10, 10 + altura_texto_pontuacao * 3 + altura_texto_jogador)): Desenha o texto do jogador 2 na tela.

Um loop aninhado é usado para iterar sobre as células do tabuleiro.

A cor da célula é determinada com base no valor armazenado no tabuleiro. Se for 1, a cor será verde; se for 2, a cor será amarela; se for 3, a cor será vermelha.

pygame.draw.rect(screen, cor, (posicao_tabuleiro[0] + coluna * largura_celula, posicao_tabuleiro[1] + linha * largura_celula, largura_celula-1, largura_celula-1)): Desenha um retângulo representando a célula na tela.

pygame.display.flip(): Atualiza a tela com as alterações feitas.

Essa função é responsável por atualizar visualmente o estado do jogo, desenhando o tabuleiro, as células e as informações dos jogadores na tela.

17. Escolha

A função escolha utiliza a biblioteca random para gerar um número aleatório entre 0 e 1, inclusive, representando uma escolha aleatória. Em seguida, retorna o número gerado.

Esta função pode ser usada para simular uma escolha aleatória entre duas opções, por exemplo, quando se precisa decidir aleatoriamente qual jogador começa um jogo ou qual caminho seguir em um jogo de lógica. O valor retornado será 0 ou 1, indicando a escolha feita.

18. jogar_semaforo

A função jogar_semaforo implementa a lógica do jogo do semáforo usando a biblioteca Pygame. Recebe como parâmetros os nomes dos jogadores (jogador1 e jogador2), a tela do jogo (screen), as dimensões da tela (width e height), e opcionalmente, os pontos iniciais de cada jogador (pontos_1 e pontos_2).

A função inicia o Pygame, carrega a imagem de fundo, define as fontes para os botões, inicializa as pontuações dos jogadores, e prepara as variáveis para exibir a pontuação e os nomes dos jogadores na tela.

Em seguida, cria o tabuleiro do jogo, define o tamanho das células do tabuleiro, e configura a posição do tabuleiro na tela. A função também configura a janela do Pygame com o título "Jogo do Semáforo" e desenha o tabuleiro inicial na tela.

O jogo é executado num loop principal que verifica eventos, como ação de fechar a janela ou cliques do rato. Se ocorrer um clique dentro do tabuleiro, a função fazer_jogada é chamada para realizar a jogada na posição correspondente e atualizar o tabuleiro. Em seguida, verifica-se se há um vencedor utilizando a função verificar_vencedor. Se houver um vencedor, a pontuação é atualizada, o tabuleiro é reiniciado e o jogador atual é alterado.

A função também exibe um botão "Sair" na tela, que encerra o jogo quando clicado.

No final, quando o loop principal termina, o Pygame é encerrado com pygame.quit().

Esta função pode ser usada para iniciar e executar o jogo do semáforo entre dois jogadores na interface gráfica do Pygame.

19. Jogar_semaforo_bot

Bastante semelhante à função *jogar_semáforo* mas no entanto chama o *bot* fácil para jogar. Este *bot* apenas joga de forma *random* na sua jogada.

20. jogar_bot_medio

Semelhante à função *jogar_semaforo_bot* mas no entanto esta chama o *bot_medio*.

21. bot_medio

Esta função analisa o tabuleiro e chama a função *fazer_jogada* na casa em que é possível ganhar.

22. jogar_bot_dificil

Semelhante à função *jogar_semaforo_bot* mas no entanto esta chama o *bot_dificil*.

23. bot_dificil

É semelhante a *bot* médio só que também analisa a jogada feita e faz uma jogada que torna impossível a vitoria ao jogador na jogada seguinte.

24. regras

A função regras exibe as regras do jogo na tela. Utiliza a biblioteca Pygame para criar uma janela e desenhar o texto e os botões.

25. Conclusão

O Jogo do Semáforo, um jogo desenvolvido em *Python*, foi descrito neste relatório. O objetivo era desenvolver um jogo com as regras do jogo do semáforo em que os jogadores competem para formar uma linha de três peças da mesma cor numa direção horizontal, vertical ou diagonal.

No processo de desenvolvimento, encontramos problemas como o tratamento de eventos do usuário, posicionamento de elementos na tela e carregamento de imagens. No entanto, conseguimos superar esses desafios e implementar o jogo do semáforo com sucesso, usando corretamente a documentação do *Pygame* e conceitos de programação.

Aprendemos mais sobre o desenvolvimento de jogos e melhoramos as nossas habilidades em *Python* durante o processo de criação. Aprendemos muito sobre como organizar o código, usar bibliotecas externas e interagir com o usuário para criar uma experiência de jogo agradável.

Em suma, criar este jogo em *Python* foi uma experiência difícil, mas gratificante. Foi emocionante assistir à evolução do projeto e ao resultado final. Estamos satisfeitos com os resultados e, no futuro, esperamos continuar a desenvolver e a melhor as nossas habilidades no desenvolvimento de jogos e na área de programação.