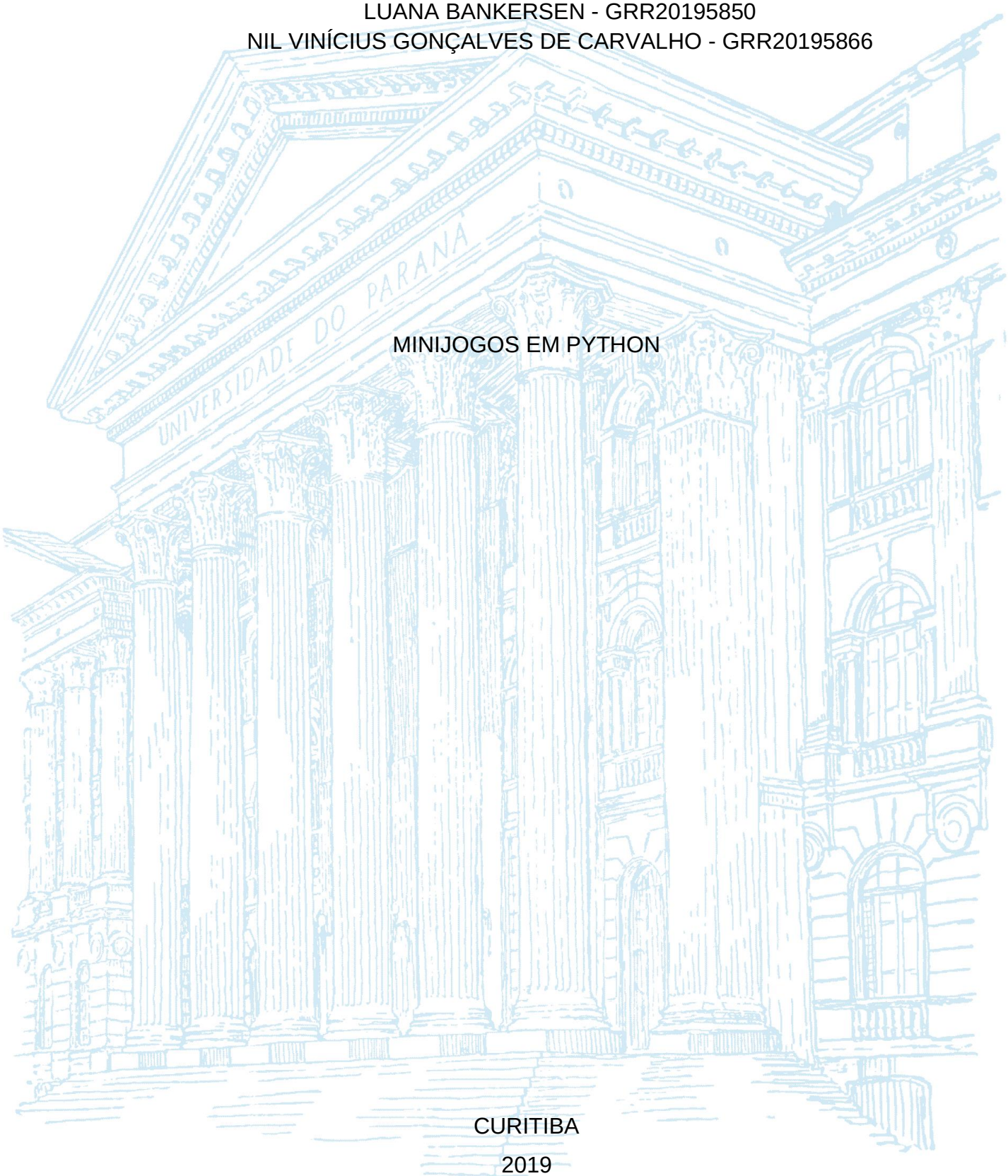
UNIVERSIDADE FEDERAL DO PARANÁ

CARLOS EDUARDO STIPP - GRR20195829
LUANA BANKERSEN - GRR20195850
NIL VINÍCIUS GONÇALVES DE CARVALHO - GRR20195866

MINIJOGOS EM PYTHON



CURITIBA
2019

CARLOS EDUARDO STIPP - GRR20195829
LUANA BANKERSEN - GRR20195850
NIL VINÍCIUS GONÇALVES DE CARVALHO - GRR20195866

MINIJOGOS EM PYTHON

Relatório apresentado à disciplina Fundamentos de Programação de Computadores do Curso de graduação em Matemática da Universidade Federal do Paraná

Orientador: Prof. Jackson Antônio do Prado Lima

CURITIBA

2019

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Este relatório tem como objetivo descrever detalhadamente a construção dos seguintes minijogos, utilizando Python 3: Genius, Jogo da Memória, Quiz e Snake. Os jogos citados estão reunidos em um menu principal, que possibilita ao jogador escolher qual pretende iniciar.

1.1 MOTIVAÇÃO

A motivação para a produção desta série de minijogos advém da maneira simples e divertida com a qual foi possível treinar os conhecimentos adquiridos durante o semestre, de modo a explorar as aplicações práticas do Python 3.

1.2 RESUMO

1.2.1 Minjogos

Este programa detém os quatro jogos já citados anteriormente. A seguir, cada um destes terá seu resumo e explicação.

1.2.1.1 Genius

O jogo Genius tem como base uma matriz 2x2, na qual cada elemento é representado por uma cor (amarelo, vermelho, verde e azul). Essas cores brilham em uma sequência que cresce conforme as rodadas passam. O objetivo é que o jogador memorize a sequência e refaça-a, clicando nas respetivas cores até a 10ª rodada – ou seja, até o momento em que haja uma sequência em que ocorrem 10 acendimentos dessas cores. Caso o jogador consiga acertar todas as 10 sequências, será mostrada uma tela de "Vitória", com a possibilidade de começar um novo jogo ou de voltar ao menu principal. Entretanto, basta um erro para que o jogador perca o jogo, sendo assim levado a uma tela de "Game Over", com as mesmas opções da tela de "Vitória".

1.2.1.2 Jogo da Memória

O Jogo da Memória consiste basicamente em uma determinada quantidade de peças (ou cartas) com símbolos, sendo que cada um destes tem pelo menos um par de peças. As cartas são colocadas para baixo na disposição escolhida pelo jogador, com a condição de que a disposição dos símbolos seja feita aleatoriamente – sem que o jogador saiba distingui-las –, com o intuito de que ele apenas passe a saber onde está cada símbolo ao virar a peça em questão.

Após as peças serem dispostas para baixo de maneira aleatória, o jogador deve virar duas delas de cada vez e, se os símbolos das cartas escolhidas forem iguais, deve mantê-las viradas para cima. Caso as peças escolhidas sejam diferentes, o jogador perde uma vida das 10 que possui, e deve voltar a virar as duas cartas para baixo, na mesma disposição em que estavam anteriormente. Dessa forma, deve-se repetir este movimento com o intuito de fazer com que todos os símbolos fiquem virados para cima. O jogo termina quando todas as cartas estiverem viradas para cima (vitória) ou com o esgotamento das vidas (derrota); No código, essas condições levam à uma tela de "Vitória" e de "Game Over", respectivamente. Nesta versão do jogo, há 18 peças, com 9 símbolos (números) distintos; a disposição das peças é feita num retângulo 6x3.

1.2.1.3 Quiz

O jogo Quiz se resume a uma série de perguntas que o jogador se compromete a responder. Nesta versão, o jogo é composto por 34 questões, sendo elas dos seguintes temas: Animais, Filmes/Séries/Livros, Jogos, Pessoas, Lugares e Época. Cada jogada do Quiz é composta por 10 perguntas aleatoriamente sorteadas e que não se repetem, bastando errar uma só questão para que o jogador perca o jogo e seja levado à uma tela de “Game Over”, na qual encontrará as opções de recomeçar ou sair para o menu principal. Entretanto, caso todas as 10 perguntas sejam respondidas corretamente, o jogador vence a rodada do Quiz – sendo levado para uma tela de “Vitória”, na qual encontrará as mesmas opções da tela de “Game Over”.

1.2.1.4 Snake

No jogo Snake, o jogador controla uma estrutura verde (a cobra) na tela. Seu objetivo é comer o maior número de frutas vermelhas, passando por elas. Cada vez que a cobra se alimenta de uma fruta, a próxima aparece em uma posição aleatória da tela. A dificuldade do jogo aumenta conforme a estrutura verde se alimenta das frutas vermelhas, visto que o comprimento dela aumenta em um bloco sempre que isso ocorre. O jogo acaba quando a cobra colide com seu próprio corpo, o que leva o jogador a uma tela de “Game Over”, que possibilita a ele que recomece ou volte ao menu principal. Além disso, a tela de fim de jogo mostra a pontuação obtida pelo jogador.

2 DESENVOLVIMENTO

2.1 METODOLOGIA

2.1.1 Bibliotecas utilizadas

Para a construção dos minijogos, do menu principal e das telas de “Game Over” e “Vitória”, foram utilizadas as bibliotecas: pygame e random.

2.1.1.1 Pygame

Desta biblioteca foram utilizadas as seguintes funções:

- **display.update**(retângulo): Atualiza a porção retângulo da tela. Caso não sejam passados parâmetros, a tela é atualizada como um todo.
- **draw. rect**(superfície, corRGB, (x, y, l, h)): Desenha um retângulo em uma superfície de cor igual a corRGB, a partir das coordenadas x,y de largura l e altura h.
- **draw.circle**(superfície, corRGB, (x,y), r): Desenha um círculo em uma superfície de cor igual a corRGB, com centro em x,y e raio = e.
- **event.get**(tipo_evento): Detecta os eventos do tipo tipo_evento de uma lista de possíveis eventos e os retorna.

```
QUIT            none
ACTIVEEVENT       gain, state
KEYDOWN          key, mod, unicode, scancode
KEYUP           key, mod
MOUSEMOTION       pos, rel, buttons
MOUSEBUTTONUP      pos, button
MOUSEBUTTONDOWN    pos, button
JOYAXISMOTION      joy, axis, value
JOYBALLMOTION      joy, ball, rel
JOYHATMOTION       joy, hat, value
JOYBUTTONUP        joy, button
JOYBUTTONDOWN      joy, button
VIDEORESIZE        size, w, h
VIDEOEXPOSE        none
USEREVENT        code
```

Figura 1 – Lista de possíveis eventos

- **event.type**():
- **font.SysFont**(fonte_escolhida, tamanho, negrito, itálico): Gera uma fonte de fonte_escolhida, com tamanho = tamanho. Negrito e itálico tem como padrão o valor lógico "False" - caso seja atribuído outro valor a eles, o texto será negrito ou itálico, respectivamente.
- **mouse.get_pos**(): Retorna as coordenadas x e y do cursor.
- **Surface.blit**(sup ,(x,y)): Desenha a superfície sup na superfície Surface a partir das coordenadas x e y.
- **time.Clock**(): Cria um objeto com a utilidade de marcar o tempo.
- **time.delay**(t): Pausa o programa por um período de t milissegundos.
- **pygame.init**(): Inicializa todos os módulos de pygame importados.
- **pygame.quit**(): Desliga todos os módulos de pygame que foram inicializados anteriormente.

2.1.1.2 Random

Desta biblioteca foram utilizadas as seguintes funções:

- **randrange**(começo, fim, passo): Gera um elemento aleatório entre começo e fim (inclusive), de passo a passo.
- **randint**(começo, fim): Gera um número inteiro aleatório entre começo e fim.
- **sample**(x,k): Gera uma sequência de k elementos únicos da sequência k.
- **shuffle**(x): Embaralha os elementos da sequência x.

2.1.2 Funções utilizadas

A seguir serão mostradas as funções utilizadas e seu funcionamento nos minijogos e no menu principal dentro do qual eles estão contidos.

- texto(mensag, cor, tamtexto, x, y)

De forma mais ampla, essa função é responsável por gerar um texto que será mostrado na tela. Tem como parâmetros a frase, a cor, o tamanho e as coordenadas do texto a ser gerado.

```
def texto(mensag, cor, tamtexto, x, y):
    font = pygame.font.SysFont(None, tamtexto)
    textol = font.render(mensag, True, cor)
    fundo.blit(textol, [x,y])
```

Figura 2 – Função texto()

2.1.2.1 Menu Principal

Na função principal do programa é definido e desenhado o menu principal, o qual espera o clique do usuário na opção de jogo desejada.

```
sair = False
while not(sair):
    clique = False
    fundo.fill(preto)
    pygame.draw.rect(fundo, branco, [0, 30, largura, 140])
    pygame.draw.rect(fundo, branco, [0, 200, largura, 70])
    pygame.draw.rect(fundo, branco, [0, 300, largura, 70])
    pygame.draw.rect(fundo, branco, [0, 400, largura, 70])
    pygame.draw.rect(fundo, branco, [0, 500, largura, 70])
    texto("MINI JOGOS", preto, 150, 200, 50)
    texto("Genius", verde, 100, 370, 205)
    texto("Quiz", agua, 100, 410, 300)
    texto("Memória", vermelho, 100, 350, 405)
    texto("Snake", verde, 100, 390, 505)
    pygame.display.update()
    while not(clique):
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:                    # reconhece onde o jogador está clicando com o mouse
                mousex = pygame.mouse.get_pos()[0]
                mousey = pygame.mouse.get_pos()[1]
                if mousey > 200 and mousey < 280:
                    reiniciar = jogo_genius()
                    while reiniciar:
                        reiniciar = jogo_genius()
                    clique = True
                if mousey > 300 and mousey < 380:
                    jogo_quiz()
                    clique = True
                if mousey > 400 and mousey < 480:
                    jogo_memoria()
                    clique = True
                if mousey > 500 and mousey < 580:
                    jogo_cobra()
                    clique = True
            if event.type == pygame.QUIT:
                sair = True
                clique = True

pygame.quit()
exit()
```

Figura 3 – Menu Principal

2.1.2.2 Genius

Para o jogo Genius foram utilizadas as seguintes funções:

- tela_apagada()

Função responsável por gerar a tela a ser usada no jogo Genius como estado inicial das cores, ou seja, com as quatro cores apagadas.

```
def tela_apagada():
    fundo.fill(preto)
    pygame.draw.rect(fundo, amareloescuro, [0, 0, largura/2 - 2, altura/2 - 2])
    pygame.draw.rect(fundo, vermelhoescuro, [largura/2 + 2, 0, largura/2 - 2, altura/2 - 2])
    pygame.draw.rect(fundo, verdeescuro, [0, altura/2 + 2 , largura/2 - 2, altura/2 - 2])
    pygame.draw.rect(fundo, azulescuro, [largura/2 + 2, altura/2 + 2, largura/2 - 2, altura/2 - 2])
    pygame.display.update()
```

Figura 4 – Função tela_apagada()

- jogo_genius()

De início, geramos um vetor de dez elementos, sendo que cada um corresponde a um número aleatório entre 1 e 4 (inclusive), tendo em vista que cada número representará uma das cores da tela de jogo. Um *for* é responsável pelo prosseguimento das 10 rodadas, e um *for* interior a este é responsável por mostrar na tela a sequência de cores de número de elementos igual a rodada na qual o jogador se encontra. No *for* mais exterior também é onde se encontra a verificação da sequência feita pelo jogador. Caso haja erro, o programa chama a tela de "Game Over". Depois de mostrar a sequência ao jogador, o programa espera que este tente repeti-la. Caso ele faça a sequência errada, uma tela de "Game Over" aparecerá, possibilitando o recomeço do jogo ou a volta ao menu. Caso acerte a sequência, a próxima rodada seguirá, e assim por diante até a 10a rodada - na qual, se atingida, mostrará a tela de "Vitória".

```
def jogo_genius():
    jogar = True
    while jogar:
        rodadas = 10
        sequencia = []
        for i in range(rodadas):
            cor = randint(1,4)
            sequencia.append(cor)

        rod = 1
        erro = 0
        sair = False
        up = False
        fim = False


        for i in range(1,rodadas+1):
            fundo.fill(branco)
            texto("RODADA " + str(i), preto, 50, largura/3, altura/3)
            pygame.display.update()
            pygame.time.delay(500)
            for j in range(i):

                for event in pygame.event.get():
                   if event.type == pygame.QUIT:
                       sair = True
                       j = i
                tela_apagada()
                pygame.time.delay(1000)
                if sequencia[j] == 1:
                    pygame.draw.rect(fundo, amarelo, [0,0,largura/2 - 2,altura/2 - 2])
                elif sequencia[j] == 2:
                    pygame.draw.rect(fundo, vermelho, [largura/2 + 2,0,largura/2 - 2,altura/2 - 2])
                elif sequencia[j] == 3:
                    pygame.draw.rect(fundo, verde, [0,altura/2 + 2,largura/2 - 2,altura/2 - 2])
                elif sequencia[j] == 4:
                    pygame.draw.rect(fundo, azul, [largura/2 + 2,altura/2 + 2,largura/2 - 2,altura/2 - 2])
                pygame.display.update()
                pygame.time.delay(500)
```

Figura 5 – Função jogo_genius() (Parte 1)

```
        jogador = []
        certo = 0
        while not(certo):
            tela_apagada()
            clique = False
            while not(clique):
                for event in pygame.event.get():
                    if event.type == pygame.MOUSEBUTTONDOWN:
                        mousex = pygame.mouse.get_pos()[0]
                        mousey = pygame.mouse.get_pos()[1]
                        if mousex < (largura/2 - 2) and mousey < (altura/2 - 2):
                            jogador.append(1)
                            pygame.draw.rect(fundo, amarelo, [0,0,largura/2 - 2,altura/2 - 2])
                            pygame.display.update()
                            pygame.time.delay(500)
                            clique = True
                        elif mousex > (largura/2 + 2) and mousex < largura and mousey < (altura/2 - 2):
                            jogador.append(2)
                            pygame.draw.rect(fundo, vermelho, [largura/2 + 2,0,largura/2 - 2,altura/2 - 2])
                            pygame.display.update()
                            pygame.time.delay(500)
                            clique = True
                        elif mousex < (largura/2 - 2) and mousey > (altura/2 + 2) and  mousey < altura:
                            jogador.append(3)
                            pygame.draw.rect(fundo, verde, [0,altura/2 + 2,largura/2 - 2,altura/2 - 2])
                            pygame.display.update()
                            pygame.time.delay(500)
                            clique = True
                        elif mousex > (largura/2 + 2) and mousex < largura and mousey > (altura/2 + 2) and  mousey < altura:
                            jogador.append(4)
                            pygame.draw.rect(fundo, azul, [largura/2 + 2,altura/2 + 2,largura/2 - 2,altura/2 - 2])
                            pygame.display.update()
                            pygame.time.delay(500)
                            clique = True
                    if event.type == pygame.QUIT:
                        clique = True
                        pygame.quit()
                        exit()
```

Figura 6 – Função jogo_genius() (Parte 2)

```
if len(jogador) == len(sequencia[:j+1]):
    if jogador == sequencia[:j+1]:
        certo = 1
    else:
        fundo.fill(tomate)
        texto("GAME OVER!", preto, 200, largura - 950, altura - 400)
        pygame.draw.rect(fundo, preto, [0, 7*(altura/8), largura, altura/8])
        pygame.draw.rect(fundo, branco, [largura/2 -1, 7*(altura/8), 2, altura/8])
        texto("Reiniciar", branco, 100, 90, 7*altura/8 + 8)
        texto("Sair", branco, 100, largura/2 + 180, 7*altura/8 + 8)
        pygame.display.update()
        clique = False
        while not(clique):
            for event in pygame.event.get():
                if event.type == pygame.MOUSEBUTTONDOWN:                    # reconhece onde o jogador está clicando com o mouse
                        mousex = pygame.mouse.get_pos()[0]
                        mousey = pygame.mouse.get_pos()[1]
                        if  mousex < largura/2 - 1 and mousey > 7*altura/8:
                            return 1
                        if mousex > largura/2 + 1 and mousey > 7*altura/8:
                            return 0
                if event.type == pygame.QUIT:
                    clique = True
                    pygame.quit()
                    sys.exit()
```

Figura 7 – Função jogo_genius() (Parte 3)

2.1.2.3 Jogo da Memória

Para o Jogo da Memória, utilizamos a seguinte função:

- jogo_memoria()

Essa função é responsável por tornar a tela jogável com o tamanho de 640 x 480 pixels, desenhar as cartas e as vidas com “pygame.draw” na mesma, embaralhar as cartas com shuffle (função da biblioteca random) e definir a quantidade de vidas (10). Deste modo, a função computará quais foram a primeira e a segunda cartas que o usuário clicou, e analisará se as cartas têm valores iguais. Caso sim, as peças ficarão viradas para cima, senão, elas voltarão a ficar para baixo. Quando o usuário perde ou ganha, a função irá retornar para uma tela de 1000 x 600 pixels, indicando uma tela de “Game Over” ou de "Vitória", respectivamente.

```
def jogo_memoria():
    #JOGO DA MEMÓRIA

    #as cartas sao divididas em tres linhas e 6 colunas
    #cada linha é representada pela letra da variavel e cada coluna pelo numero desta
    #o valor das variaveis com letras em maiusculo representam o status da carta naquela posicao
    #virada pra baixo=1
    #virada pra cima=2
    #pares certos=3
    #o valor das varievaies com letras em minusculo representam o numero daquela carta
    while True:
        A1=1
        A2=1
        A3=1
        A4=1
        A5=1
        A6=1
        B1=1
        B2=1
        B3=1
        B4=1
        B5=1
        B6=1
        C1=1
        C2=1
        C3=1
        C4=1
        C5=1
        C6=1
```

Figura 8 – Função jogo_memoria() (Parte 1)

```
        #sorteia o numero de cada carta
        Pecas=[1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9]
        shuffle(Pecas)
        a1=Pecas[0]
        a2=Pecas[1]
        a3=Pecas[2]
        a4=Pecas[3]
        a5=Pecas[4]
        a6=Pecas[5]
        b1=Pecas[6]
        b2=Pecas[7]
        b3=Pecas[8]
        b4=Pecas[9]
        b5=Pecas[10]
        b6=Pecas[11]
        c1=Pecas[12]
        c2=Pecas[13]
        c3=Pecas[14]
        c4=Pecas[15]
        c5=Pecas[16]
        c6=Pecas[17]

        #largura e altura da tela
        larg=640
        alt=480
        fundo = pygame.display.set_mode((larg,alt))
        fundo.fill(preto)                           #torna o fundo preto

        vidas=10
        cartavirada1=0          #numero da primeira carta virada
        cartavirada2=0          #numero da segunda carta virada
        cartaviradastatus1=0    #indica o status da primeira carta virada, somente virada para cima ou com par feito(2 ou 3 respectivamente)
        cartaviradastatus2=0    #indica o status da segunda carta virada, somente virada para cima ou com par feito(2 ou 3 respectivamente)
```

Figura 9 – Função jogo_memoria() (Parte 2)

```
    #Desenhos dos retangulos das cartas
    pygame.draw.rect(fundo,branco,[(larg*0.109375)-30,(alt*0.158333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.265625)-30,(alt*0.158333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.421875)-30,(alt*0.158333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.578125)-30,(alt*0.158333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.734375)-30,(alt*0.158333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.890625)-30,(alt*0.158333)-50,60,100])

    pygame.draw.rect(fundo,branco,[(larg*0.109375)-30,(alt*0.420833)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.265625)-30,(alt*0.420833)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.421875)-30,(alt*0.420833)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.578125)-30,(alt*0.420833)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.734375)-30,(alt*0.420833)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.890625)-30,(alt*0.420833)-50,60,100])

    pygame.draw.rect(fundo,branco,[(larg*0.109375)-30,(alt*0.683333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.265625)-30,(alt*0.683333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.421875)-30,(alt*0.683333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.578125)-30,(alt*0.683333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.734375)-30,(alt*0.683333)-50,60,100])
    pygame.draw.rect(fundo,branco,[(larg*0.890625)-30,(alt*0.683333)-50,60,100])

    #desenhos dos circulos das vidas
    pygame.draw.circle(fundo,vermelho,[11+35,429],25)
    pygame.draw.circle(fundo,vermelho,[72+35,429],25)
    pygame.draw.circle(fundo,vermelho,[133+35,429],25)
    pygame.draw.circle(fundo,vermelho,[194+35,429],25)
    pygame.draw.circle(fundo,vermelho,[255+35,429],25)
    pygame.draw.circle(fundo,vermelho,[316+35,429],25)
    pygame.draw.circle(fundo,vermelho,[377+35,429],25)
    pygame.draw.circle(fundo,vermelho,[438+35,429],25)
    pygame.draw.circle(fundo,vermelho,[499+35,429],25)
    pygame.draw.circle(fundo,vermelho,[560+35,429],25)

    #atualiza a tela
    pygame.display.update()
```

Figura 10 – Função jogo_memoria() (Parte 3)

```
#laco de repeticão que se finaliza quando o usuario perde, ganha ou fecha o jogo
while vidas!=0 and (A1+A2+A3+A4+A5+A6+B1+B2+B3+B4+B5+B6+C1+C2+C3+C4+C5+C6) != 54:
    #laco que define qual carta foi clicada e a vira para cima
    while cartaviradal==0 and cartaviradastatusl==0:
        #comando que reconhece eventos durante a execução (clique do mouse, comando como alt+f4, etc
        for event in pygame.event.get():
            if event.type==pygame.QUIT:                         #verifica se foi clicado para fechar a aba
                pygame.quit()
            elif event.type == pygame.MOUSEBUTTONDOWN:          #comando que reconhece o clique do mouse
                mousex=pygame.mouse.get_pos()[0]                #armazena a coordenada x em que o mouse clicou
                mousey=pygame.mouse.get_pos()[1]                #armazena a coordenada y em que o mouse clicou
                #condicional que determinara qual a primeira carta que devera ser virada para cima baseado na posicao que o jogador clicou
                if (mousex >40 and mousex < 100 and mousey > 26 and mousey < 126) and A1==1:
                    texto(str(a1),amarelo,100,40+10,26+20)
                    A1=2
                    cartaviradal=a1
                    cartaviradastatusl=A1
                elif (mousex >140 and mousex < 200 and mousey > 26 and mousey < 126) and A2==1:
                    texto(str(a2),amarelo,100,140+10,26+20)
                    A2=2
                    cartaviradal=a2
                    cartaviradastatusl=A2
                elif (mousex >240 and mousex < 300 and mousey > 26 and mousey < 126) and A3==1:
                    texto(str(a3),amarelo,100,240+10,26+20)
                    A3=2
                    cartaviradal=a3
                    cartaviradastatusl=A3
                elif (mousex >340 and mousex < 400 and mousey > 26 and mousey < 126) and A4==1:
                    texto(str(a4),amarelo,100,340+10,26+20)
                    A4=2
                    cartaviradal=a4
                    cartaviradastatusl=A4
                elif (mousex >440 and mousex < 500 and mousey > 26 and mousey < 126) and A5==1:
                    texto(str(a5),amarelo,100,440+10,26+20)
                    A5=2
                    cartaviradal=a5
                    cartaviradastatusl=A5
```

Figura 11 – Função jogo_memoria() (Parte 4)

```
#laco que define qual a segunda carta escolhida
while cartavirada2==0 and cartaviradastatus2==0:
    #comando que reconhece eventos durante a execução (clique do mouse, comando como alt+f4, etc
    for event in pygame.event.get():
        if event.type==pygame.QUIT:                 #verifica se foi clicado para fechar a aba
            pygame.quit()
        elif (event.type == pygame.MOUSEBUTTONDOWN): #comando que reconhece o clique do mouse
            mousex=pygame.mouse.get_pos()[0]        #armazena a coordenada x em que o mouse clicou
            mousey=pygame.mouse.get_pos()[1]        #armazena a coordenada y em que o mouse clicou
            #condicional que determinara qual a primeira carta que devera ser virada para cima baseado na posicao do clique do jogador
            if (mousex >40 and mousex < 100 and mousey > 26 and mousey < 126) and A1==1:
                texto(str(a1),amarelo,100,40+10,26+20)
                A1=2
                cartavirada2=a1
                cartaviradastatus2=A1
            elif (mousex >140 and mousex < 200 and mousey > 26 and mousey < 126) and A2==1:
                texto(str(a2),amarelo,100,140+10,26+20)
                A2=2
                cartavirada2=a2
                cartaviradastatus2=A2
            elif (mousex >240 and mousex < 300 and mousey > 26 and mousey < 126) and A3==1:
                texto(str(a3),amarelo,100,240+10,26+20)
                A3=2
                cartavirada2=a3
                cartaviradastatus2=A3
            elif (mousex >340 and mousex < 400 and mousey > 26 and mousey < 126) and A4==1:
                texto(str(a4),amarelo,100,340+10,26+20)
                A4=2
                cartavirada2=a4
                cartaviradastatus2=A4
            elif (mousex >440 and mousex < 500 and mousey > 26 and mousey < 126) and A5==1:
                texto(str(a5),amarelo,100,440+10,26+20)
                A5=2
                cartavirada2=a5
                cartaviradastatus2=A5
            elif (mousex >540 and mousex < 600 and mousey > 26 and mousey < 126) and A6==1:
                texto(str(a6),amarelo,100,540+10,26+20)
                A6=2
                cartavirada2=a6
                cartaviradastatus2=A6
```

Figura 12 – Função jogo_memoria() (Parte 5)

```
            elif (mousex >240 and mousex < 300 and mousey > 278 and mousey < 378) and C3==1:
                texto(str(c3),amarelo,100,240+10,278+20)
                C3=2
                cartavirada2=c3
                cartaviradastatus2=C3
            elif (mousex >340 and mousex < 400 and mousey > 278 and mousey < 378) and C4==1:
                texto(str(c4),amarelo,100,340+10,278+20)
                C4=2
                cartavirada2=c4
                cartaviradastatus2=C4
            elif (mousex >440 and mousex < 500 and mousey > 278 and mousey < 378) and C5==1:
                texto(str(c5),amarelo,100,440+10,278+20)
                C5=2
                cartavirada2=c5
                cartaviradastatus2=C5
            elif (mousex >540 and mousex < 600 and mousey > 278 and mousey < 378) and C6==1:
                texto(str(c6),amarelo,100,540+10,278+20)
                C6=2
                cartavirada2=c6
                cartaviradastatus2=C6
    pygame.display.update()
    pygame.time.delay(1000)          #deixa as cartas escolhidas a mostra antes de vira-las para baixo novamente
```

Figura 13 – Função jogo_memoria() (Parte 6)

```
            if cartaviradal!=cartavirada2:
                if A1==2:
                    A1=1
                    pygame.draw.rect(fundo,branco,[(larg*0.109375)-30,(alt*0.158333)-50,60,100])
                if A2==2:
                    A2=1
                    pygame.draw.rect(fundo,branco,[(larg*0.265625)-30,(alt*0.158333)-50,60,100])
                if A3==2:
                    A3=1
                    pygame.draw.rect(fundo,branco,[(larg*0.421875)-30,(alt*0.158333)-50,60,100])
                if A4==2:
                    A4=1
                    pygame.draw.rect(fundo,branco,[(larg*0.578125)-30,(alt*0.158333)-50,60,100])
                if A5==2:
                    A5=1
                    pygame.draw.rect(fundo,branco,[(larg*0.734375)-30,(alt*0.158333)-50,60,100])
                if A6==2:
                    A6=1
                    pygame.draw.rect(fundo,branco,[(larg*0.890625)-30,(alt*0.158333)-50,60,100])

                if B1==2:
                    B1=1
                    pygame.draw.rect(fundo,branco,[(larg*0.109375)-30,(alt*0.420833)-50,60,100])
                if B2==2:
                    B2=1
                    pygame.draw.rect(fundo,branco,[(larg*0.265625)-30,(alt*0.420833)-50,60,100])
                if B3==2:
                    B3=1
                    pygame.draw.rect(fundo,branco,[(larg*0.421875)-30,(alt*0.420833)-50,60,100])
                if B4==2:
                    B4=1
                    pygame.draw.rect(fundo,branco,[(larg*0.578125)-30,(alt*0.420833)-50,60,100])
                if B5==2:
                    B5=1
                    pygame.draw.rect(fundo,branco,[(larg*0.734375)-30,(alt*0.420833)-50,60,100])
                if B6==2:
                    B6=1
                    pygame.draw.rect(fundo,branco,[(larg*0.890625)-30,(alt*0.420833)-50,60,100])
                if C1==2:
                    C1=1
                    pygame.draw.rect(fundo,branco,[(larg*0.109375)-30,(alt*0.683333)-50,60,100])
```

Figura 14 – Função jogo_memoria() (Parte 7)

```
                if C2==2:
                    C2=1
                    pygame.draw.rect(fundo,branco,[(larg*0.265625)-30,(alt*0.683333)-50,60,100])
                if C3==2:
                    C3=1
                    pygame.draw.rect(fundo,branco,[(larg*0.421875)-30,(alt*0.683333)-50,60,100])
                if C4==2:
                    C4=1
                    pygame.draw.rect(fundo,branco,[(larg*0.578125)-30,(alt*0.683333)-50,60,100])
                if C5==2:
                    C5=1
                    pygame.draw.rect(fundo,branco,[(larg*0.734375)-30,(alt*0.683333)-50,60,100])
                if C6==2:
                    C6=1
                    pygame.draw.rect(fundo,branco,[(larg*0.890625)-30,(alt*0.683333)-50,60,100])

                vidas = vidas -1
```

Figura 15 – Função jogo_memoria() (Parte 8)

```
elif cartaviradal==cartavirada2:
    if cartaviradastatus1==A1:
        A1=3
    elif cartaviradastatus1==A2:
        A2=3
    elif cartaviradastatus1==A3:
        A3=3
    elif cartaviradastatus1==A4:
        A4=3
    elif cartaviradastatus1==A5:
        A5=3
    elif cartaviradastatus1==A6:
        A6=3
    elif cartaviradastatus1==B1:
        B1=3
    elif cartaviradastatus1==B2:
        B2=3
    elif cartaviradastatus1==B3:
        B3=3
    elif cartaviradastatus1==B4:
        B4=3
    elif cartaviradastatus1==B5:
        B5=3
    elif cartaviradastatus1==B6:
        B6=3
    elif cartaviradastatus1==C1:
        C1=3
    elif cartaviradastatus1==C2:
        C2=3
    elif cartaviradastatus1==C3:
        C3=3
    elif cartaviradastatus1==C4:
        C4=3
    elif cartaviradastatus1==C5:
        C5=3
    elif cartaviradastatus1==C6:
        C6=3
    if cartaviradastatus2==A1:
        A1=3
    elif cartaviradastatus2==A2:
        A2=3
```

Figura 16 – Função jogo_memoria() (Parte 9)

```
        #condicional que retira as vidas da tela
        if vidas==9:
            pygame.draw.circle(fundo,preto,[560+35,429],25)
        if vidas==8:
            pygame.draw.circle(fundo,preto,[499+35,429],25)
        if vidas==7:
            pygame.draw.circle(fundo,preto,[438+35,429],25)
        if vidas==6:
            pygame.draw.circle(fundo,preto,[377+35,429],25)
        if vidas==5:
            pygame.draw.circle(fundo,preto,[316+35,429],25)
        if vidas==4:
            pygame.draw.circle(fundo,preto,[255+35,429],25)
        if vidas==3:
            pygame.draw.circle(fundo,preto,[194+35,429],25)
        if vidas==2:
            pygame.draw.circle(fundo,preto,[133+35,429],25)
        if vidas==1:
            pygame.draw.circle(fundo,preto,[72+35,429],25)
        if vidas==0:
            pygame.draw.circle(fundo,preto,[11+35,429],25)
            pygame.display.update()
            pygame.time.delay(500)
        #reseta as variaveis para que o programa consiga ler novamente 2 novas cartas
        cartaviradastatus1=0
        cartaviradastatus2=0
        cartavirada1=0
        cartavirada2=0
        pygame.display.update()
```

Figura 17 – Função jogo_memoria (Parte 10)

```
#após o primeiro laco de repeticão acabar, o jogo é levado a uma das duas telas possiveis (vitoria ou derrota)
if vidas == 0:
    fundo = pygame.display.set_mode((largura,altura))
    fundo.fill(tomate)
    texto("GAME OVER!", preto, 200, largura - 950, altura - 400)
    pygame.draw.rect(fundo, preto, [0, 7*(altura/8), largura, altura/8])
    pygame.draw.rect(fundo, branco, [largura/2 -1, 7*(altura/8), 2, altura/8])
    texto("Reiniciar", branco, 100, 90, 7*altura/8 + 8)
    texto("Sair", branco, 100, largura/2 + 180, 7*altura/8 + 8)
    pygame.display.update()
    clique = False
    while not(clique):
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:                    # reconhece onde o jogador está clicando com o mouse
                    mousex = pygame.mouse.get_pos()[0]
                    mousey = pygame.mouse.get_pos()[1]
                    if  mousex < largura/2 - 1 and mousey > 7*altura/8:
                        clique = True
                    if mousex > largura/2 + 1 and mousey > 7*altura/8:
                        return 0
            if event.type == pygame.QUIT:
                clique = True
                pygame.quit()
                sys.exit()
```

Figura 18 – Função jogo_memoria (Parte 11)

```
else:
    fundo = pygame.display.set_mode((largura,altura))
    fundo.fill(agua)
    texto("VITORIA   :)", salmao, 200, largura - 950, altura - 400)
    pygame.draw.rect(fundo, preto, [0, 7*(altura/8), largura, altura/8])
    pygame.draw.rect(fundo, branco, [largura/2 -1, 7*(altura/8), 2, altura/8])
    texto("Reiniciar", branco, 100, 90, 7*altura/8 + 8)
    texto("Sair", branco, 100, largura/2 + 180, 7*altura/8 + 8)
    pygame.display.update()
    clique = False
    while not(clique):
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:                    # reconhece onde o jogador está clicando com o mouse
                    mousex = pygame.mouse.get_pos()[0]
                    mousey = pygame.mouse.get_pos()[1]
                    if  mousex < largura/2 - 1 and mousey > 7*altura/8:
                        clique = True
                    if mousex > largura/2 + 1 and mousey > 7*altura/8:
                        return 0
            if event.type == pygame.QUIT:
                clique = True
                pygame.quit()
                sys.exit()
```

Figura 19 – Função jogo_memoria (Parte 12)

2.1.2.4 Quiz

Para o jogo Quiz foram utilizadas as seguintes funções:

- gera_questao()

É chamada quando uma nova questão e suas alternativas devem ser mostradas na tela. Discerne qual será a questão impressa por uma série de condicionais e pelo seu parâmetro, que contém o número da questão sorteado na função jogo_quiz().

```
def gera_questao(questao_escolhida):                                # gera a questão aleatoriamente

##    for i in range(len(Q[questao_escolhida])):
##        texto(Q[questao_escolhida][i]


    if questao_escolhida == 1:
      texto(Q1, preto, 30, largura-950,altura-565)
      texto(Q1_1, preto, 30, largura-950,altura-535)
      texto(Q1_2, preto, 30, largura-950,altura-505)
      for i in range(5):
          texto(R1[i], preto, 30, largura-950, (altura-320)+70*(i-1))          # mostra as alternativas da questão

    elif questao_escolhida == 2:
      texto(Q2, preto, 30, largura-950,altura-565)
      texto(Q2_1, preto, 30, largura-950,altura-535)
      texto(Q2_2, preto, 30, largura-950,altura-505)
      texto(Q2_3, preto, 30, largura-950,altura-475)
      for i in range(5):
          texto(R2[i], preto, 30, largura-950, (altura-320)+70*(i-1))          # mostra as alternativas da questão

    elif questao_escolhida == 3:
      texto(Q3, preto, 30, largura-950,altura-565)
      texto(Q3_1, preto, 30, largura-950,altura-535)
      texto(Q3_2, preto, 30, largura-950,altura-505)
      for i in range(5):
          texto(R3[i], preto, 30, largura-950, (altura-320)+70*(i-1))          # mostra as alternativas da questão

    elif questao_escolhida == 4:
      texto(Q4, preto, 30, largura-950,altura-565)
      texto(Q4_1, preto, 30, largura-950,altura-535)
      for i in range(5):
          texto(R4[i], preto, 30, largura-950, (altura-320)+70*(i-1))          # mostra as alternativas da questão

    elif questao_escolhida == 5:
      texto(Q5, preto, 30, largura-950,altura-565)
      texto(Q5_1, preto, 30, largura-950,altura-535)
      for i in range(5):
          texto(R5[i], preto, 30, largura-950, (altura-320)+70*(i-1))          # mostra as alternativas da questão
```

Figura 20 – Função gera_questao() (Parte 1)

```
    elif questao_escolhida == 33:
      texto(Q33, preto, 30, largura-950,altura-565)
      for i in range(5):
          texto(R33[i], preto, 30, largura-950, (altura-320)+70*(i-1))                  # mostra as alternativas da questão

    elif questao_escolhida == 34:
      texto(Q34, preto, 30, largura-950,altura-565)
      texto(Q34_1, preto, 30, largura-950,altura-535)
      for i in range(5):
          texto(R34[i], preto, 30, largura-950, (altura-320)+70*(i-1))                  # mostra as alternativas da questão

    pygame.display.update()                                  # atualização da tela
    resposta = 'z'

    while resposta == 'z':
      for event in pygame.event.get():
        if event.type == pygame.MOUSEBUTTONDOWN:                      # reconhece onde o jogador está clicando com o mouse
            mousex = pygame.mouse.get_pos()[0]
            mousey = pygame.mouse.get_pos()[1]
            if mousex > 30 and mousex < 930 and mousey > 200 and mousey < 250:       # reconhece quando o jogador clica na alternativa 'a'
                resposta = 'a'
            if mousex > 30 and mousex < 930 and mousey > 270 and mousey < 320:       # reconhece quando o jogador clica na alternativa 'b'
                resposta = 'b'
            if mousex > 30 and mousex < 930 and mousey > 340 and mousey < 390:       # reconhece quando o jogador clica na alternativa 'c'
                resposta = 'c'
            if mousex > 30 and mousex < 930 and mousey > 410 and mousey < 460:       # reconhece quando o jogador clica na alternativa 'd'
                resposta = 'd'
            if mousex > 30 and mousex < 930 and mousey > 480 and mousey < 530:       # reconhece quando o jogador clica na alternativa 'e'
                resposta = 'e'

        if event.type == pygame.QUIT:
            rodada = 11
            resposta = 'y'
            pygame.quit()

    return resposta, correta[questao_escolhida]
```

Figura 21 – Função gera_questão (Parte 2)

- jogo_quiz()

Em primeiro lugar, a função gera uma tela que espera o clique só usuário. Em seguida, sorteia as dez questões que serão mostradas ao jogador, a partir deste momento a função gera_questao é chamada para mostrar a pergunta na tela e verifica se a resposta dada pelo o usuário foi correta. Caso a resposta tenha esteja certa, a questão seguinte é mostrada, e assim por diante até a décima rodada - na qual, em caso de acerto, mostra a tela de "Vitória" para o usuário. Entretanto, em caso de erro (em qualquer momento do jogo) a tela de "Game Over" será impressa.

```
def jogo_quiz():
    fundo.fill(verde)
    texto("  Quiz  ", preto, 400, largura - 1000, altura - 500)
    texto("Clique para continuar", preto, 100, 120, altura - 80)

    clique = False
    pygame.display.update()
    while not(clique):
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:
                clique = True
            if event.type == pygame.QUIT:
                pygame.quit()

    while True:
        rodada = 0
        erro = 0
        lista_questoes = sample([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34], 10)

        while erro == 0 and rodada < 10:
            print(rodada)
            fundo.fill(branco)
            pygame.draw.rect(fundo, salmao, [largura-970, altura-580, 930, 150])       # cria o retângulo das perguntas
            pygame.draw.rect(fundo, agua, [largura-970, altura-400, 930, 50])          # cria o retângulo da alternativa 'a'
            pygame.draw.rect(fundo, agua, [largura-970, altura-330, 930, 50])          # cria o retângulo da alternativa 'b'
            pygame.draw.rect(fundo, agua, [largura-970, altura-260, 930, 50])          # cria o retângulo da alternativa 'c'
            pygame.draw.rect(fundo, agua, [largura-970, altura-190, 930, 50])          # cria o retângulo da alternativa 'd'
            pygame.draw.rect(fundo, agua, [largura-970, altura-120, 930, 50])          # cria o retângulo da alternativa 'e'

            resposta_dada, resposta_correta = gera_questao(lista_questoes[rodada])
            if resposta_dada == resposta_correta:                   # caso a resposta esteja correta, uma nova pergunta é gerada para o jogador
                rodada += 1
                print("aumentoRodada")
            else:                                                   # caso a resposta esteja incorreta, o jogo fecha
                erro = 1
```

Figura 22 – Função jogo_quiz() (Parte 1)

```
    if erro:
        fundo.fill(tomate)
        texto("GAME OVER!", preto, 200, largura - 950, altura - 400)
        pygame.draw.rect(fundo, preto, [0, 7*(altura/8), largura, altura/8])
        pygame.draw.rect(fundo, branco, [largura/2 -1, 7*(altura/8), 2, altura/8])
        texto("Reiniciar", branco, 100, 90, 7*altura/8 + 8)
        texto("Sair", branco, 100, largura/2 + 180, 7*altura/8 + 8)
        pygame.display.update()
        clique = False
        while not(clique):
            for event in pygame.event.get():
                if event.type == pygame.MOUSEBUTTONDOWN:                    # reconhece onde o jogador está clicando com o mouse
                        mousex = pygame.mouse.get_pos()[0]
                        mousey = pygame.mouse.get_pos()[1]
                        if  mousex < largura/2 - 1 and mousey > 7*altura/8:
                            clique = True
                        if mousex > largura/2 + 1 and mousey > 7*altura/8:
                            return 0
                if event.type == pygame.QUIT:
                    clique = True
                    pygame.quit()
                    sys.exit()
```

Figura 23 – Função jogo_quiz() (Parte 2)

```
    else:
        fundo.fill(agua)
        texto("VITORIA   :)", salmao, 200, largura - 950, altura - 400)
        pygame.draw.rect(fundo, preto, [0, 7*(altura/8), largura, altura/8])
        pygame.draw.rect(fundo, branco, [largura/2 -1, 7*(altura/8), 2, altura/8])
        texto("Reiniciar", branco, 100, 90, 7*altura/8 + 8)
        texto("Sair", branco, 100, largura/2 + 180, 7*altura/8 + 8)
        pygame.display.update()
        clique = False
        while not(clique):
            for event in pygame.event.get():
                if event.type == pygame.MOUSEBUTTONDOWN:                     # reconhece onde o jogador está clicando com o mouse
                        mousex = pygame.mouse.get_pos()[0]
                        mousey = pygame.mouse.get_pos()[1]
                        if  mousex < largura/2 - 1 and mousey > 7*altura/8:
                            clique = True
                        if mousex > largura/2 + 1 and mousey > 7*altura/8:
                            return 0
                if event.type == pygame.QUIT:
                    clique = True
                    pygame.quit()
                    sys.exit()
```

Figura 24 – Função jogo_quiz() (Parte 3)

2.1.2.5 Snake

Para o jogo Snake foram utilizadas as seguintes funções:

- cobra(CobraXY)

Essa função serve para fazer o desenho do retângulo verde usado como cabeça e corpo da cobra. CobraXY, seu parâmetro, é uma lista com as coordenadas x e y, de onde o retângulo deve ser desenhado.

```
def cobra(CobraXY):
    for XY in CobraXY:
        pygame.draw.rect(fundo, verde, [XY[0], XY[1],tamanho, tamanho])
```

Figura 25 – Função cobra(CobraXY)

- maca(macax, macay)

Essa função serve para desenhar as maçãs usadas no Snake. Tem como parâmetros as coordenadas em que a maçã deve ser desenhada, e usa da variável global *tamanho* para determinar o tamanho do retângulo a ser desenhado.

```
def maca(macax, macay):
    pygame.draw.rect(fundo,vermelho, [macax, macay,tamanho, tamanho])
```

Figura 26 – Função maca(macax, macay)

- jogo_cobra()

Nesta função está definido o funcionamento principal do jogo Snake, nela é feita a detecção de colisão entre a cobra e as maçãs, a detecção de colisão da cobra com si mesma e onde é feita a movientação da cobra na tela, onde são chamadas as funções de desenho da cobra e da maçã, onde a pontuação do jogador é contada, e onde é gerada a tela de "Game Over" quando o jogador perde.

```
def jogo_cobra():     # Nil fazer funcao para zerar parametros
    sair = False
    gameover = False

    coordx = randrange(0, largura - tamanho, 50)    #gera valores numa grade altura x largura de blocos de 50 pixels
    coordy = randrange(0, altura - tamanho, 50)
    macax = randrange(0, largura - tamanho, 50)
    macay = randrange(0, altura - tamanho, 50)
    macaXY = [0,0]
    while (coordx == macax and coordy == macay):
        macax = randrange(0, largura - tamanho, 50)
        macay = randrange(0, altura - tamanho, 50)

    velocx = 0          #veloc ded deslocamento nos eixos
    velocy = 0
    CobraXY = []        #posicoes dos blocos da cobra
    CobraComp = 1       #comp inicial da cobra
    pontos = 0          #pontuacao inicial

    while not(sair):
        relogio.tick(10)
        fundo.fill(preto)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sair = True
            if event.type == pygame.KEYDOWN:            #recebe a direcao q a cobra deve ir, sem poder inverter sua direcao
                if event.key == pygame.K_LEFT and velocx != tamanho and mexeu:
                    velocy = 0
                    velocx = -tamanho
                    mexeu = False
                elif event.key == pygame.K_RIGHT and velocx != -tamanho and mexeu:
                    velocy = 0
                    velocx = tamanho
                    mexeu = False
                elif event.key == pygame.K_UP and velocy != tamanho and mexeu:
                    velocy = -tamanho
                    velocx = 0
                    mexeu = False
```

Figura 27 – Função jogo_cobra() (Parte 1)

```
while gameover:
    fundo.fill(tomate)
    texto("GAME OVER!", preto, 200, largura - 950, altura - 400)
    texto("Pontuação: " +str(pontos), preto, 80, 280, 350)
    pygame.draw.rect(fundo, preto, [0, 7*(altura/8), largura, altura/8])
    pygame.draw.rect(fundo, branco, [largura/2 -1, 7*(altura/8), 2, altura/8])
    texto("Reiniciar", branco, 100, 90, 7*altura/8 + 8)
    texto("Sair", branco, 100, largura/2 + 180, 7*altura/8 + 8)
    pygame.display.update()
    clique = False
    while not(clique):
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN:                    # reconhece onde o jogador está clicando com o mouse
                    mousex = pygame.mouse.get_pos()[0]
                    mousey = pygame.mouse.get_pos()[1]
                    if  mousex < largura/2 - 1 and mousey > 7*altura/8:
                        clique = True
                        sair = False
                        gameover = False
                        coordx = randrange(0, largura - tamanho, 50)
                        coordy = randrange(0, altura - tamanho, 50)
                        macax = randrange(0, largura - tamanho, 50)
                        macay = randrange(0, altura - tamanho, 50)
                        velocx = 0
                        velocy = 0
                        CobraXY = []
                        CobraComp = 1
                        pontos = 0
                    if mousex > largura/2 + 1 and mousey > 7*altura/8:
                        return 0
            if event.type == pygame.QUIT:
                clique = True
                pygame.quit()
                exit()


    pygame.display.update()

pygame.display.update()
```

Figura 28 – Função jogo_cobra() (Parte 2)

```
        elif event.key == pygame.K_DOWN and velocy != -tamanho and mexeu:
            velocy = tamanho
            velocx = 0
            mexeu = False

coordx += velocx                            #define a posicao da cobra
coordy += velocy

if coordx > largura - 1:                    #tranposrta a cobra de um lado ao outro da tela quando aquela ultrapassa a borda desta
    coordx = 0
if coordx < 0:
    coordx = largura - tamanho
if coordy > altura  - 1:
    coordy = 0
if coordy < 0:
    coordy = altura - tamanho

CobraCabeca = []
CobraCabeca.append(coordx)
CobraCabeca.append(coordy)
CobraXY.append(CobraCabeca)

if len(CobraXY) > CobraComp:
    del CobraXY[0]
    mexeu = True
if any(CobraCorpo == CobraCabeca for CobraCorpo in CobraXY[:-1]):
        gameover = True

cobra(CobraXY)

maca(macax, macay)

if (coordx == macax) and (coordy == macay):
    macaXY[0] = macax
    macaXY[1] = macay
    while (coordx == macax and coordy == macay) or any(CobraCorpo == macaXY for CobraCorpo in CobraXY[:-1]):

        macax = randrange(0, largura - tamanho, 50)
        macay = randrange(0, altura - tamanho, 50)
    CobraComp += 1
    pontos += 1
```

Figura 29 – Função jogo_cobra() (Parte 3)

2.1.2.6 Tela de Game Over

Dentro das funções dos quatro jogos, quando ocorre a falha do jogador – como por exemplo, no Snake, quando a cobra toca em si mesma –, geramos uma tela de “Game Over” que possibilita o recomeço do jogo ou a volta ao menu principal.

Figura 30 – Tela Game Over (Padrão)

Figura 31 – Tela Game Over (Snake)

2.1.2.7 Tela de Vitória

Nas funções jogo_quiz, jogo_genius e jogo_memoria, usamos uma tela de “Vitória” para quando o jogador alcança o objetivo do jogo, sendo que nesta o usuário obtém as opções de voltar ao menu principal ou começar o jogo novamente.

Figura 32 – Tela de Vitória (Padrão)

2.2 RESULTADOS OBTIDOS

Foi possível concluir 4 jogos funcionais, sendo eles: o jogo Snake clássico; um jogo da memória de dimensões 6x3 (que acaba quando o usuário perder as 10 vidas ou vencer); um quiz com 34 perguntas diferentes, sendo que 10 serão perguntadas ao usuário aleatoriamente; o jogo Genius com 10 rodadas aleatórias; e uma tela inicial que relaciona todos eles.

2.3 DIFICULDADES ENCONTRADAS

Encontramos dificuldade na comunicação do grupo, tendo em vista que em alguns momentos um participante ou outro ficava ocioso por não termos dividido as atividades de uma maneira satisfatória. Houve uma curva de aprendizado um tanto quanto complicada quando se diz respeito à biblioteca pygame, principalmente quanto às funções relacionadas à contagem de tempo.

3 CONSIDERAÇÕES FINAIS

Este trabalho mostrou-se fundamental para o grupo, tendo em vista os desafios proporcionados por ele – prazos a serem cumpridos, múltiplas tarefas, etc – , que auxiliaram no amadurecimento acadêmico dos integrantes. Além disso, esta produção teve papel importante na preparação do grupo para futuras atividades semelhantes.

3.1 SUGESTÕES DE TRABALHOS

- Agenda virtual que entende letra cursiva (inteligência artificial)
- Programa que faz spams em páginas e sites que negam ciência
- Corretor ortográfico
- Calculadora científica com display (como a do Windows)
- Connect 4
- Jogo do dinossauro (estilo google)
- Duck_Hunt

REFERÊNCIA

[1] PYGAME DOCUMENTATION. Disponível em: https://pygame.org/docs/. Acessado em: 25/11/2019

[2] PYTHON FOR BEGINNERS. Disponível em: https://www.pythonforbeginners.com/random/how-to-use-the-random-module-in-python. Acessado em: 24/11/2019

[3] PYGAME – TUTORIAL SNAKE GAME. Disponível em: https://www.youtube.com/playlist?list=PLzn2mIpnKXEo0iiFrlqv-fFAbRd0cjDLe. Acessado em: 24/11/2019

[4] PYTHON DOCUMENTATION RANDOM. Disponível em: https://docs.python.org/3/library/random.html. Acessado em: 25/11/2019