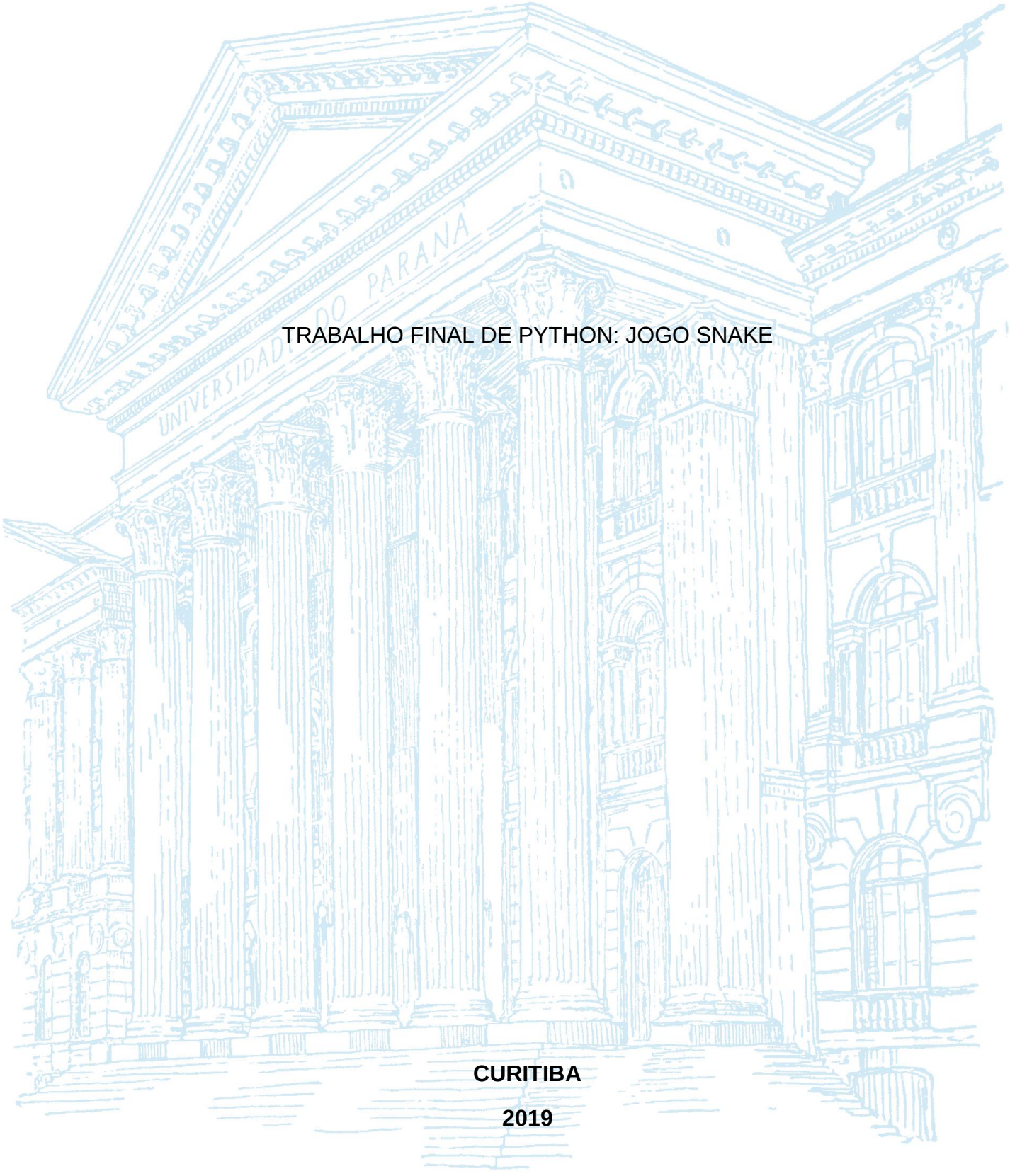
UNIVERSIDADE FEDERAL DO PARANÁ

ISABELE AYUMI MIYAWAKI



TRABALHO FINAL DE PYTHON: JOGO SNAKE

CURITIBA

2019

ISABELE AYUMI MIYAWAKI

GRR 20193439

TRABALHO FINAL: JOGO “SNAKE” (“EVIL COBRINHA”)

Relatório do trabalho final da disciplina de Fundamentos de Programação dos cursos de Matemática e Medicina da Universidade Federal do Paraná.

Orientador: Prof. Jackson Antônio do Prado Lima

CURITIBA

2019

SUMÁRIO

1 INTRODUÇÃO

1.1 SOBRE A LINGUAGEM PYTHON E A BIBLIOTECA PYGAME

Gerida pela PSF (Python Software Foundation) a linguagem de programação Python é extremamente simples e robusta. Sua arquitetura bem projetada, fornece um bom desempenho e legibilidade do código. Esta linguagem Revista Tecnologias em Projeção, v10, n°1, ano 2019. p.56 está presente em várias aplicações do cotidiano, dando suporte as demais tarefas. A linguagem possui ampla versatilidade, podendo atuar bem na área comercial, ou em áreas mais específicas como por exemplo: no desenvolvimento científico, geoprocessamento e em aplicações mobile, tanto isoladamente ou integrada a outras ferramentas, o que expande ainda mais o mundo Pythônico.

Dentro do espectro Pythônico, o Pygame é uma biblioteca muito utilizada para o desenvolvimento de jogos multiplataforma (independente de sistema operacional) feita para ser utilizada em conjunto com a linguagem de programação Python. O seu nome tem origem da junção de Py, proveniente de Python e Game, que significa Jogo, ou seja, Jogos em Python. Pygame se baseia na ideia de que as tarefas mais intensivas a nível computacional em um jogo podem ser abstraídas separadamente da lógica principal, ou seja, o uso de memória e CPU (úteis para processar imagens e sons) são tratados pelo próprio código do Pygame e não pelo código do seu jogo. Assim, torna-se possível utilizar uma linguagem de alto nível, como Perl, Lua ou Python para organizar a estrutura do jogo em si.

1.2 SOBRE O JOGO ‘SNAKE’

Serpente (*Snake*, também conhecido como "jogo da cobrinha") é um jogo que ficou conhecido por diversas versões cuja versão inicial começou com o jogo Blockade de 1976, sendo fitas, desde então, várias imitações em videogame e computadores.

O jogo consiste em um retângulo que representa a ‘snake’ que é movida pelo jogador através das teclas ‘sobe, desce, direita e esquerda’. Também há um pequeno retângulo que representa a maçã que a cobrinha objetiva colidir contra ou ‘comer’. Dessa forma, o jogador manipula a cobrinha de forma que ele maximize as vezes que colida com a maçã, uma vez que a cada vez que ele consegue comer a maçã ganha um ponto no placar. O jogador deve, entretanto, evitar encostar nas bordas da tela, já que ao fazê-lo, perderá o jogo.

2 OBJETIVO

O trabalho teve como objetivo o desenvolvimento do jogo ‘Snake’ utilizando a biblioteca Pygame, principal biblioteca utilizada para desenvolvimento de jogos na linguagem Python, de tal forma que os conhecimentos aprendidos na disciplina ‘Fundamentos de Progamação’ pudessem ser aplicados de maneira prática.

3 DESENVOLVIMENTO

3.1 BIBLIOTECAS IMPORTADAS

```
import pygame
from random import randrange
```

De início, importei duas bibliotecas: pygame e randrange da biblioteca random. O pygame foi importado para dar toda a funcionalidade de jogo ao código e foi escolhido por ser uma ferramenta de fácil manipulação e altamente eficiente. O randrange foi importado para obterem-se coordenadas randômicas e inteiras das imagens da cobrinha e da maçã.

3.2 VARIÁVEIS FORA DAS FUNÇÕES

De modo que o jogo possua música de fundo, importei um arquivo mp3 à pasta do jogo, utilizando das documentações do pygame: pygame.mixer.init() que iniciará a música, pygame.mixer.music.load() que define o mp3 tocado e o pygame.mixer.music.play() que manterá a música tocando durante o jogo.

```
pygame.mixer.init()  # inicia a música
pygame.mixer.music.load('musica.mp3')
pygame.mixer.music.play()
```

De maneira que os fundos, as imagens e os textos no jogo sejam coloridos, defini algumas cores cujos códigos podem ser encontrados no site cloford.com.

```
rosa = (255, 62, 150)  # encontra as cores no site http://cloford.com/
azul = (151, 255, 255)
vermelho = (255, 48, 48)
verde = (24,252,0)
amarelo = (255,215,0)
preto = (41,36,33)
```

Para iniciar o jogo no pygame, insere-se o comando pygame.init() como tentativa e a frase “O jogo Evil Cobrinha não pôde ser inicializado!” como uma exceção.

```
try:
    pygame.init()  # inicia o jogo no pygame
```

```
except:
    print("O jogo Evil Cobrinha não pode ser inicializado!")
```

A variável relógio recebe a documentação do pygame de pygame.time.Clock() de maneira que defina a velocidade em que ocorrerá a atualização da página do jogo.

A variável “fundo” recebe a documentação pygame.display.set_mode de forma que defina a largura e a altura da tela, em pixels.

A documentação pygame.display.set_caption define o nome que aparecerá na aba do jogo (“Evil Cobrinha”).

```
relogio = pygame.time.Clock()  # define a velocidade em que ocorrerá o update de
tela
fundo = pygame.display.set_mode((largura, altura))  # tamanho da tela, 600 x 600
pixels
pygame.display.set_caption('Evil Cobrinha')  # nome que aparece na aba
```

3.3 FUNÇÕES

3.3.1 FUNÇÃO TEXTO:

```
def texto(msg, cor, size, x, y):
    font = pygame.font.SysFont(None, size)
    texto1 = font.render(msg, True, cor)
    fundo.blit(texto1, [x, y])  # insere o texto e define a posição em que se
encontra o texto (centro da tela)
```

A função texto irá definir a mensagem, a cor, o tamanho e as coordenadas x e y dos textos do jogo. É constituída pelas variáveis:

- font, que define a fonte do texto pela documentação pygame.font.SysFont () “create a Font object from the system fonts”
- texto1, que irá inserir a mensagem e a cor do texto pela documentação font.render ()
- fundo com a documentação blit para inserir o texto e definir as coordenadas x e y em que o texto aparece na tela

3.3.2 FUNÇÃO COBRINHA

```
def cobrinha(CobraXY):
    for XY in CobraXY:  # cria uma lista CobraXY
        pygame.draw.rect(fundo, rosa, [XY[0], XY[1], tamanho, tamanho])  # define
o fundo, a cor e as dimensões da cobrinha
```

Na função da cobra, cria-se uma lista CobraXY e define-se o fundo, a cor e as dimensões da cobra com a documentação pygame.draw.rect ().

3.3.3 FUNÇÃO RINGO (MAÇÃ)

```
def ringo(pos_x, pos_y):
    pygame.draw.rect(fundo, vermelho, [pos_x, pos_y, tamanho, tamanho])  # define
o fundo, a cor e as dimensões da maçã
```

Na função da maçã, define-se o fundo, a cor e as dimensões da maçã utilizado a documentação pygame.draw.rect ().

3.3.4 FUNÇÃO EVIL_COBRINHA (JOGO)

A função do jogo em si pode ser dividida, para fins elucidativos, em partes:

```
def evil_cobrinha():  # função do jogo
    sair = True
    fimdejogo = False
    pos_x = randrange(0, largura - tamanho, 10)  # faz com que a posição inicial x
da cobrinha seja aleatória e múltipla de 10
    pos_y = randrange(0, altura - tamanho-placar, 10)  # faz com que a posição
inicial y da cobrinha seja aleatória e múltipla de 10
    ringo_x = randrange(0, largura - tamanho, 10)  # faz com que a posição inicial
x da maçã seja aleatória e múltipla de 10
    ringo_y = randrange(0, altura - tamanho-placar, 10)  # faz com que a posição
inicial y da maçã seja aleatória e múltipla de 10
    velocidade_x = 0
    velocidade_y = 0
    CobraXY = []  # Cria uma lista da cobra XY
    CobraComp = 1  # Indica o comprimento inicial da cobra
    pontos = 0
```

- Variáveis da função:

  - ✓ Estabele-se que “sair” recebe True e “fimdejogo” recebe False para, posteriormente colocar os comandos do jogo e de saída em um laço de repetição
  - ✓ Define-se as posições x e y iniciais da cobrinha e da maçã como pos_x, pos_y, ringo_x, ringo_y, respectivamente, de tal forma que as posições sejam randômicas e inteiras
  - ✓ Define-se as velocidades nos planos x e y recebendo 0 para posteriormente, determinar a velocidade dos objetos
  - ✓ Em Cobra XY define-se o caráter de lista da variável
  - ✓ Em CobraComp define-se o comprimento inicial da cobra (1 unidade)
  - ✓ Os pontos do placar estão inicialmente zerados, definido pela variável ponto

- Laço de repetição e o primeiro for:

O primeiro for, com a documentação pygame.event.get(), define uma sequência de eventos.

Primeiramente, define-se a possibilidade de sair do jogo com a documentação do pygame de event.type e pygame.QUIT.

Também se define os diferentes comandos dependendo da tecla apertada pelo jogador, sendo que se o jogador apertar a tecla c (definido pela documentação pygame.K_c) todas as variáveis da função continuarão valendo e se o jogador apertar a tecla s (definido pela documentação do pygame.K_s) ele conseguirá sair do jogo.

```
while sair:
    while fimdejogo:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sair = False
                fimdejogo = False
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_c:
                    sair = True
                    fimdejogo = False
                    pos_x = randrange(0, largura - tamanho, 10)  # faz com que a
posição inicial x da cobrinha seja aleatória e múltipla de 10
                    pos_y = randrange(0, altura - tamanho-placar, 10)  # faz com
que a posição inicial y da cobrinha seja aleatória e múltipla de 10
                    ringo_x = randrange(0, largura - tamanho, 10)  # faz com que a
posição inicial x da maçã seja aleatória e múltipla de 10
                    ringo_y = randrange(0, altura - tamanho-placar, 10)  # faz com
que a posição inicial y da maçã seja aleatória e múltipla de 10
                    velocidade_x = 0
                    velocidade_y = 0
                    CobraXY = []  # Cria uma lista da cobra XY
                    CobraComp = 1  # Indica o comprimento inicial da cobra
                    pontos = 0
                elif event.key == pygame.K_s:
                    sair = False
                    fimdejogo = False
```

- Definição da tela de “Game Over”

Ainda dentro da função do jogo (“evil_cobrinha”) se definem alguns comandos da tela do jogo “Game over”, que terá cor vermelha pela documentação fundo.fill() e 4 textos diferentes (definidos na função de texto).

```
fundo.fill(vermelho)
texto('Oh, Jeez, Rick - Game Over!', azul, 50, 65, 30)
texto('Sua pontuação: '+str(pontos), preto, 40, 70, 80)
pygame.draw.rect(fundo, preto, [45, 120, 210, 50])
texto('Continuar(C)', amarelo, 35, 50, 125)
pygame.draw.rect(fundo, preto, [300, 120, 150, 50])
texto('Sair(S)', amarelo, 35, 310, 125)
pygame.display.update()
```

- O segundo “for”

No segundo for define-se uma sequências de eventos com o pygame.event.get() em que os comandos são definidos pelas teclas de “cima, baixo, direita e esquerda”, sendo que a imagem (tanto a cobrinha quanto a maçã) irá se movimentar na direção apontada pela tecla mas não conseguirá se mover para a direção oposta da qual está se movimentando (se estou me movendo para a direita, não posso logo em seguida me mover para a esquerda), aumentando o grau de dificuldade do jogo.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        sair = False
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT and velocidade_x != tamanho:
            velocidade_y = 0
            velocidade_x = -tamanho
        elif event.key == pygame.K_RIGHT and velocidade_x != -tamanho:
            velocidade_y = 0
            velocidade_x = tamanho
        elif event.key == pygame.K_UP and velocidade_y != tamanho:
            velocidade_x = 0
            velocidade_y = -tamanho
        elif event.key == pygame.K_DOWN and velocidade_y != -tamanho:
            velocidade_x = 0
            velocidade_y = tamanho
```

- Outras definições

Em seguida, definem-se outros commandos tais como o preenchimento da tela do jogo de azul, o movimento das imagens ao aumentar sempre a posição (cujo valor cai dentro das funções de cobrinha e de maçã), define-se as posições da cobrinha como uma lista dentro de outra lista, de modo que o primeiro elemento da lista (o rabo da cobra) seja apagado na medida em que a cobra se move, define-se também o placar da pontuação.

```
fundo.fill(azul)
pos_x += velocidade_x
pos_y += velocidade_y

CobraInicio = [pos_x, pos_y]  # Cria uma outra lista da cabeça da cobra
CobraXY.append(CobraInicio)  # adiciona a lista CobraInicio à lista CobraXY

if len(CobraXY) > CobraComp:
    del CobraXY[0]  # deleta o primeiro elemento da lista, que é o rabo da cobra,
```

```
o novo elemento da lista

pygame.draw.rect(fundo, verde, [0, altura-placar, largura, placar])
texto('Pontuação: ' +str(pontos), preto, 25, 10, altura-50)
cobrinha(CobraXY)  # envia os valores de pos_x e pos_y em forma de lista
```

- Colisão, momento em que a cobrinha come a maçã

Quando as coordenadas da maçã e da cobrinha se igualam, define-se a colisão, com uma nova posição aleatória da maçã e com o aumento do comprimento da cobrinha em uma unidade. Também define-se o aumento do placar em uma unidade à cada vez em que a cobrinha come a maçã. Estabelece-se o “update”, a atualização da página a cada vez que a cobrinha come a maçã e também estabelece-se que o jogo terá “Game Over” se a cobrinha alcançar os limites da tela.

```
if pos_x == ringo_x and pos_y == ringo_y:  # se a houver a colisão da cobrinha com a maçã
    ringo_x = randrange(0, largura - tamanho, 10)  # faz com que a nova posição x da maçã seja aleatória e múltipla de 10
    ringo_y = randrange(0, altura - tamanho - placar, 10)  # faz com que a nova posição y da maçã seja aleatória e múltipla de 10
    CobraComp += 1  # faz com que o comprimento da cobra sempre cresça em 1 unidade
    pontos += 1

ringo(ringo_x, ringo_y)
pygame.display.update()
relogio.tick(15)
if pos_x > largura:
    fimdejogo = True
if pos_x < 0:
    fimdejogo = True
if pos_y > altura-placar:
    fimdejogo = True
if pos_y < 0:
    fimdejogo = True
```

3.4 FIM

Finaliza-se com a função do jogo e com a documentação do pygame.quit().

```
evil_cobrinha()
pygame.quit()
```

4 CONCLUSÃO

Através do desenvolvimento do jogo, pude aprender muitas funcionalidades novas sobre a biblioteca Pygame e pude praticar e aperfeiçoar diversas aplicações aprendidas durante o curso de Python.

REFERÊNCIAS BILIOGRÁFICAS

[1] SILVA, Igor Rodrigues. Revista tecnologias em projeção. **LINGUAGEM DE PROGRAMAÇÃO PYTHON**, Revista Tecnologias em Projeção, v. v10, ed. 1, p.55, 24 nov. 2019.

[2] BEGINNING Game Development with Python and Pygame: From Novice to Professional. [*S. l.*]: Will McGugan, 2007. ISBN 1590598725.

[3] PYGAME Documentation. [*S. l.*], 24 nov. 2019. Disponível em: https://www.pygame.org/docs/ref/display.html. Acesso em: 24 nov. 2019.

[4] PYGAME tutoriais. [*S. l.*], 20 ago. 2017. Disponível em: https://www.youtube.com/watch?v=Z-Q5DHPXfdg&list=PLzn2mIpnKXEo0iiFrIqv-fFAbRd0cjDLe. Acesso em: 23 nov. 2019.