UNIVERSIDADE FEDERAL DO PARANÁ

CURSO DE AGRONOMIA

Cleison Patrick

Diogo Zerbini

Guilherme de S. Sotsek

SNAKE (jogo da cobrinha)

Curitiba

2018

Cleison Patrick (GRR: 20181852)

Diogo Zerbini (GRR: 20182318)

Guilherme de S. Sotsek (GRR: 20180824)

SNAKE

Relatório apresentado à disciplina de Fundamentos de programação de computadores do Curso de Graduação em Agronomia da Universidade Federal do Paraná

Orientador: Prof. Jackson Antônio do Prado Lima

Curitiba, novembro de 2018

SUMÁRIO

1. INTRODUÇÃO

Esse relatório apresenta informações relativas ao trabalho realizado em Python sobre o jogo SNAKE, comum em telefones móveis. Rodando no script “.py”.

Python é uma linguagem de programação de alto nível, interpretada, de script, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991

2. OBJETIVOS

2.1. OBJETIVO GERAL

O objetivo é que o jogador utilize os comandos necessários para movimentar a cobra, “comer” as maçãs e aumentar seu tamanho gradativamente a cada maçã “comida”.

3. DESENVOLVIMENTO

3.1 O TRABALHO

O programa conta com três (3) funções afim de serem executadas pela biblioteca pygame. Essa biblioteca é utilizada para o desenvolvimento do jogo com interface gráfica na linguagem Python. Portanto da biblioteca pygame, iremos importar todas as funções, representado pelo comando “ * ”.

```
import pygame, random
from pygame.locals import *
```

Figura 1 – Bibliotecas usadas.

A primeira função vai gerar uma posição aleatória entre os números indicados. Radom int (randint) gera a posição aleatória da maçã, números de no mínimo 10, e no máximo 580. Retorna divisões inteiras para gerar uma posição que seja um múltiplo de 10, ou seja, assim alinhando a posição da cobra e da maçã.

```
def aleatorio():
    x = random.randint(10, 580)
    y = random.randint(10, 580)
    return (x//10 * 10, y//10 * 10)
```

Figura 2 – Função cria posição aleatória

A segunda função indica se duas informações ocupam o mesmo espaço. É utilizada para analisar a colisão entre células (cabeça da cobra e maçã), e retorna uma colisão caso aconteça.

```
def colisao(c1, c2):
    return (c1[0] == c2[0]) and (c1[1] == c2[1])
```

Figura 3 – Função para leitura de espaço

São criadas variáveis para indicar os possíveis movimentos que o usuário poderá realizar.

```
UP = 0
RIGHT = 1
DOWN = 2
LEFT = 3
```

Figura 4 – Variáveis de cada movimento

O pygame.init() representa a inicialização do jogo. Na variável criada, TELA, está o tamanho em pixels de 600X600 (uma tupla imutável). Pygame.display.set_caption() coloca o nome do programa na janela do programa.

```
pygame.init()
tela = pygame.display.set_mode((600, 600))
pygame.display.set_caption('Snake')
```

Figura 5 – Criação da tela do jogo

A cobra (snake) é representada por uma tupla, como posicionamento. A surface (superfície) é criada para delimitar o tamanho da cobra em pixels. Usamos .fill() para indicar a cor da cobra e da maçã (Fig. 7).

```
cobrinha = [(200, 200), (210, 200), (220, 200)]
cobrinha_cor = pygame.Surface((10, 10))
cobrinha_cor.fill((255, 255, 255))
```

Figura 6 – Posição inicial da cobra, tamanho e preenchimento

```
maca_pos = aleatorio()
maca = pygame.Surface((10, 10))
maca.fill((255, 0, 0))
```

Figura 7 – Posição de criação da maçã, tamanho e cor

A cobra começa indo para a esquerda (comando LEFT). O clock utilizado controla o tempo de movimentação da cobra.

```
meu_sentido = LEFT

clock = pygame.time.Clock()
```

Figura 8 – Sentido inicial e frames por segundo

Com o laço infinito (while true), capturamos os eventos (a entrada do jogo) – pygame.event.get() – com algumas condições dentro do laço. O evento tipo KEYDOWN analisa as teclas digitadas para direcionar o movimento segundo condições (if). Clock.tick() determina o fps do programa.

```
while True:
    clock.tick(10)
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()

        if event.type == KEYDOWN:
            if event.key == K_UP:
                meu_sentido = UP
            if event.key == K_DOWN:
                meu_sentido = DOWN
            if event.key == K_LEFT:
                meu_sentido = LEFT
            if event.key == K_RIGHT:
                meu_sentido = RIGHT
```

Figura 9 – Laço infinito capturando os eventos do jogo

Se ocorrer a colisão entre a cobra e a maçã, será gerada uma nova posição para a maçã.

```
if colisao(cobrinha[0], maca_pos):
    maca_pos = aleatorio()
    cobrinha.append((0, 0))
```

Figura 10 – Colisão com a maçã

O laço de repetição “ for “ é utilizado como um recurso para movimentar o corpo da cobra, e as condições aninhadas são responsáveis pela movimentação da cobra.

```
for i in range(len(cobrinha) - 1, 0, -1):
    cobrinha[i] = (cobrinha[i-1][0], cobrinha[i-1][1])

if meu_sentido == UP:
    cobrinha[0] = (cobrinha[0][0], cobrinha[0][1] - 10)
if meu_sentido == DOWN:
    cobrinha[0] = (cobrinha[0][0], cobrinha[0][1] + 10)
if meu_sentido == RIGHT:
    cobrinha[0] = (cobrinha[0][0] + 10, cobrinha[0][1])
if meu_sentido == LEFT:
    cobrinha[0] = (cobrinha[0][0] - 10, cobrinha[0][1])
```

Figura 11 – Parâmetros para movimentação da cobra

A função tela.fill() gera uma tela limpa para o jogo rodar. Enquanto que .blit() imprime o ele mento na tela em branco.

Enquanto a condição colisão cria a condição da colisão da cobra com ela mesma, finalizando o laço de repetição e terminando o jogo.

```
tela.fill((0, 0, 0))
tela.blit(maca, maca_pos)
for pos in cobrinha:
    tela.blit(cobrinha_cor, pos)

if colisao(cobrinha[0], pos):
    break
```

Figura 12 – Cor da tela, representação da cobra e maçã na tela

Pygame.display.update() atualiza as informações da tela conforme o programa cria gera novos eventos.

```
pygame.display.update()
```

Figura 13 – Atualização de tela

4. CONCLUSÃO

A mudança de tema se decorreu de uma grande dificuldade que nosso grupo encontrou com as matrizes. Esse trabalho foi mais simples, porém utiliza o conteúdo aprendido na P1 e P2. A interface gráfica é desafiadora e precisou de pesquisas e testes para funcionar. Ainda existe muito para ser aperfeiçoado neste código, como delimitação das paredes, e jogo funcionar em "*looping"* quando o jogo acabar.

REFERENCIAS

http://www.dainf.ct.utfpr.edu.br/petcoce/wp-content/uploads/2013/09/document.pdf

https://junglejf.github.io/Damas/

https://www.portugal-a-programar.pt/forums/topic/46370-snake-em-pygame/