Implementação de jogos com PyGames: Proposta de Desenvolvimento de Jogo da Serpente

Luis Ricardo De Souza[1], **Ayslan Trevizan Possebom**[1]

[1]Instituto Federal do Paraná (IFPR) campus Paranavaí
Rua José Felipe Tequinha, 1400. 87703-536. Jardim das Nações.
Paranavaí – PR – Brasil.

luisricardouser@gmail.com, ayslan.possebom@ifpr.edu.br

Abstract. *This project was developed with the aim of presenting the PyGame library and proposing the development of the snake game. For that, the Python language and some functions within the PyGame library for the development of this application are briefly presented. Subsequently, the development environment is prepared and some concepts necessary for the implementation and development of this application are presented. It deals with how the snake and the apple are created and positioned, as well as how to check the collision between game elements and its scenario.*

Resumo. *Esse projeto foi desenvolvido com o intuito de apresentar a biblioteca PyGame e propor o desenvolvimento do jogo da serpente. Para isso apresenta-se brevemente a linguagem Python e algumas funções dentro da biblioteca PyGame para o desenvolvimento desta aplicação. Posteriormente, o ambiente de desenvolvimento é preparado e alguns conceitos necessários para a implementação e desenvolvimento desta aplicação são apresentados. É tratado sobre como a a serpente e a maçã são criados e posicionados, além de como efetuar a verificação de colisão entre os elementos do jogo e seu cenário.*

1. Introdução

Em 1976, o jogo Blockade foi desenvolvido e lançado pela Gremlin Interactive para os fliperamas. Era monocromático, multi-jogador e com o personagem movido pelo teclado. Tinha o objetivo de "comer" o que aparecia na tela e evitar colisões contra as paredes do cenário. O jogador que "comia" mais e evitava as colisões por mais tempo vencia. Fez grande sucesso e serviu de inspiração para desenvolvedores concorrentes [Wikipedia 2022a]. Ganhou uma versão de único jogador em 1982 chamada Nibbler pela empresa Rock-ola, que posteriormente o lançou para computadores domésticos [Wikipedia 2022b].

Em 1997 ganhou uma versão mobile intitulada de Snake, desenvolvido pelo engenheiro Taneli Armanto para os celulares Nokia. O jogo da serpente se popularizou rapidamente, visto que podia ser jogado em qualquer local. Posteriormente ganhou novas versões com contagem de pontos e visual mais detalhado [Wikipedia 2022c]. Este jogo conta com diferentes versões que podem ser jogadas online nos dias de hoje.

Atualmente há diversas versões deste jogo com objetivos diferentes e gráficos mais atuais e com a lógica programática mais difícil. Um exemplo é o jogo *slither.io*

que contém cerca de 67 milhões de jogadores, com um limite de 500 jogadores em cada partida e com o objetivo de se tornar a maior cobra da partida.

Diante deste cenário, este trabalho visa apresentar a proposta de uma aplicação do jogo da serpente clássico, apresentando uma lógica programática simples e de fácil entendimento, utilizando a biblioteca Pygames em Python.

O trabalho está organizado da seguinte maneira: a seção 2 trata da fundamentação teórica; a seção 3 traz alguns conceitos importantes, enquanto que proposta de desenvolvimento de um jogo é apresentada na seção 4. Por fim, a conclusão é apresentada na seção 5.

2. Fundamentação Teórica

Este trabalho é um estudo com o intuito de apresentar ferramentas e conceitos de programação presentes na biblioteca PyGames para o desenvolvimento de jogos. Portanto, é importante ressaltar que trata-se de uma pesquisa da linguagem de programação Python e sua extensão PyGames. Os temas a serem abordados serão Python e PyGames.

2.1. Linguagem Python

A empresa responsável por esta linguagem fez um breve resumo apresentando-a como “Python é uma linguagem de programação orientada a objeto, interativa e imperativa [...] Python combina um poder notável com uma sintaxe muito clara” [PYGAME ORG 2022b].

Atualmente a Python Software Foundation (PSF) detém os direitos das versões 2.1 em diante. A PSF é uma organização independente e sem fins lucrativos, com o principal objetivo de melhorar e divulgar a tecnologia de código aberto Python [PYGAME ORG 2022b].

2.2. Biblioteca PyGame

Com o passar do tempo, a popularidade dos jogos e a exigência dos jogadores aumentaram cada vez mais, levando ao desenvolvimento de novos recursos e ao aprimoramento dos jogos antigos, assim, ampliando as possibilidades para os desenvolvedores de jogos. O PyGame é um desses recursos, é uma extensão da linguagem Python, que faz as tarefas simples de maneira fácil e às complexas de maneira correta, segundo o seu criador [PYGAME ORG 2022g].

Pygame é uma biblioteca do Python, multiplataforma de código aberto [PYGAME ORG 2022b] , assim, facilitando o acesso a biblioteca Simple DirectMedia Layer (SDL). Esta biblioteca oferece recursos de entrada e saída muito utilizados nos computadores, como, teclado, mouse, placa gráfica e áudio (SIMPLE DIRECTMEDIA LAYER ORG), assim facilitando o desenvolvimento de jogos.

3. Proposta do jogo da serpente

Neste capítulo é apresentado alguns conceitos que serão necessários para o desenvolvimento do jogo da serpente clássico. Este jogo tem como objetivo preencher todo o espaço possível no cenário, evitando colisões contra os elementos e as paredes.

3.1. Preparação do ambiente de desenvolvimento

Para poder desenvolver essa aplicação, precisa-se preparar o ambiente de desenvolvimento, baixando e instalando o Python e a biblioteca PyGame. Para baixar e instalar o Python, deve-se acessar o site: (https://www.python.org/downloads/). Para baixar e instalar a biblioteca PyGame, deve-se acessar o link: (pygame-1.9.6.tar.gz).

3.2. Conceitos

Neste capítulo é apresentado alguns conceitos que serão necessários para o desenvolvimento do jogo da serpente clássico. Este jogo tem como objetivo preencher todo o espaço possível no cenário, evitando colisões contra a serpente e as paredes. Os conceitos a serem abordados são: Display, Surface, Font, Time, Event, Frames per Second e sistema RGB.

3.2.1. *Display*

É um módulo que permite criar e controlar a exibição da tela do pygame. Depois que uma tela for criada, esta deve ser tratada como uma superfície normal. As alterações feitas na superfície não serão imediatamente visíveis na tela visto que uma função deverá atualizar a exibição real. Uma tela criada usando o *display* funciona como uma matriz. A origem da sua exibição começa na parte superior esquerda da tela com a coordenada do eixo X = 0 e do eixo Y = 0. Ambos os eixos aumentam positivamente em direção ao canto inferior direito da tela [PYGAME ORG 2022a].

3.2.2. *Surface*

Possui um tamanho pré-definido e é usado para desenhar qualquer imagem na tela. Pode-se comparar uma *surface* a um quadro em branco. Ao iniciá-la, um novo objeto de imagem será criado e por padrão será preenchido em preto podendo ser alterado usando o sistema de cores RGB. Seu único argumento necessário é o tamanho [PYGAME ORG 2022e]. Para a aplicação, deve-se ter atenção ao configurar o tamanho da tela, visto que sempre que a serpente chegar no limite da tela, o jogador deve ser penalizado perdendo o jogo.

3.2.3. *Font*

Módulo para carregar e renderizar fontes que serão apresentadas na tela em objetos *Surface*. As *fonts* são construídas em cima da biblioteca SDL_ttf, que inclui todas as instalações normais do pygame [PYGAME ORG 2022d].

3.2.4. *Time*

Módulo para gerenciar o tempo dentro da aplicação, representados em milisegundo. Conta com funções capazes de controlar a taxa de *Frames per Second* (FPS) e pausar a execução da aplicação por um determinado tempo [PYGAME ORG 2022f].

3.2.5. *Event*

Módulo para interagir com eventos e filas dentro da aplicação. As entradas de eventos podem ser feitas diretamente com seus módulos que leem os valores direto do objeto especificado, como o cursor ou o teclado. Os eventos suportam comparações, o que pode ser usado para gerar ações dentro da aplicação [PYGAME ORG 2022c].

3.2.6. *Frames per Second (FPS)*

É a frequência com que uma quantidade de quadro de uma animação ou fotos sequenciais é reproduzida por segundo, criando a ilusão de movimento.

3.2.7. *Sistema RGB*

É um sistema de cores aditivas de três tons de vermelho (*red*), verde (*green*) e azul (*blue*). Representa o método de cores utilizadas em monitores e dispositivos eletrônicos, fazendo referência aos tons vermelho, verde e azul para compor imagens coloridas nas telas.

4. Desenvolvimento

Para se desenvolver a aplicação, é preciso compreender a lógica do jogo da serpente, que consiste em fazer a serpente consumir a maior quantidade de maçãs possível, evitando colidir com os limites da tela ou com ela mesma.

4.1. Criando a tela do jogo

Para criar a tela do jogo com uma matriz 400x400, é necessário usar o módulo *display* e a função set_mode, que receberá o tamanho da tela da aplicação. É possível atribuir também o título “Jogo Da Serpente” à janela com a função set_caption presente no módulo *display*. A Figura 1 apresenta a tela gerada com o código abaixo.

```
tela = pygame.display.set_mode((400, 400))
pygame.display.set_caption('Jogo Da Serpente')
```

4.2. Criando os objetos

O primeiro objeto a ser criado será a serpente. Ela vai ser uma lista de tuplas que vai armazenar as suas coordenadas. É possível utilizar a função *surface* para estruturar que a serpente deve ser desenhada como um quadrado 20x20 na tela. A função *fill* é utilizada para atribuir a cor verde à serpente, através do sistema de cores RGB.

```
Serpente = [(200, 200), (220, 200), (240, 200)]
peleSerpente = pygame.Surface((20, 20))
peleSerpente.fill((0, 255, 0))
```

Para criar a maçã, utiliza-se a função *surface* para estruturar como ela será desenhada na tela como um quadrado 20x20. A cor vermelha é atribuída por meio da função *fill*. É também possível atribuir uma posição inicial informando a coordenada inicial com os eixos X=360 e Y=200 na tela.

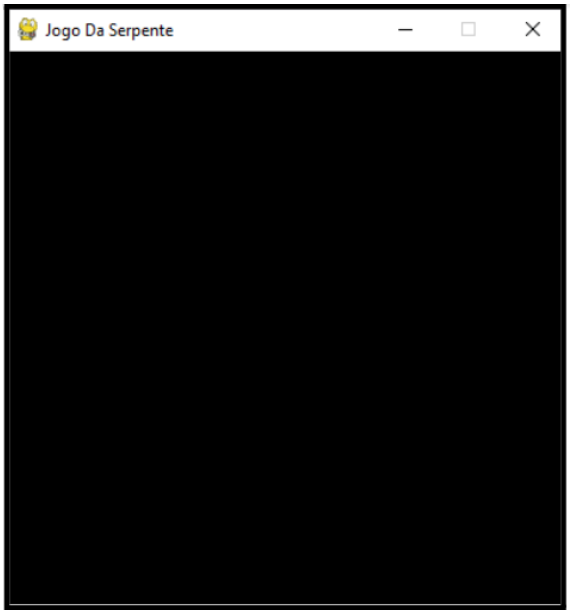

Figura 1. Tela gráfica da aplicação.

```
posicaoMaca = (360, 200)
maca = pygame.Surface((20, 20))
maca.fill((255, 0, 0))
```

Utiliza-se do módulo *time* e da função *clock* para criar um limitador de FPS da aplicação. Também é necessário a criação de uma variável que armazenará a pontuação do jogador sempre que a serpente atingir uma maçã.

```
fps = pygame.time.Clock()
pontos = 0
```

É preciso também algumas constantes que serão usadas para controlar a direção da serpente e as fontes usando o módulo *font* e a função *SysFont*. Estes itens recebem como parâmetro a fonte e o tamanho que se deseja para as fontes, configurando como as mensagens serão apresentadas na tela.

```
fontePontos = pygame.font.SysFont("arial", 20)
fonteFimJogo = pygame.font.SysFont("arial", 40)
CIMA, BAIXO, DIREITA, ESQUERDA = 0, 1, 2, 3
```

4.3. Funções

Pode-se utilizar funções nesta aplicação. As funções podem ser usadas para verificar a colisão da serpente com a maçã, para gerar valores aleatórios para a maçã depois que a serpente colidir com ela e para quando a serpente colida com ela mesma ou com os limites da tela.

A função abaixo pode ser usada para verificar se as coordenadas do eixo X e eixo Y da serpente são iguais às coordenadas do eixo X e eixo Y da maçã.

```
def colisao(serp, mac):
    return (serp[0] == mac[0]) and (serp[1] == mac[1])
```

A função posicaoAleatoria() abaixo é usada para gerar números aleatórios utilizando o módulo *random* e a função *randint*. Estes números são usados para fornecer uma nova coordenada da maçã depois da colisão com a serpente.

```
def posicaoAleatoria():
    x = (random.randint(0, 380))
    y = (random.randint(0, 380))
    return (x//20)*20, (y//20)*20
```

A função fimDeJogo() abaixo será executada quando a serpente colidir com ela mesma ou com os limites da tela. Ela vai receber a pontuacaoFinal que o jogador conseguiu atingir e, usando da função *render*, vai renderizar a fonte e a mensagem em uma só e as apresentarão na tela. A função *blit* vai receber como parâmetro a mensagem já renderizada junto com a fonte e as coordenadas do eixo X e eixo Y que a mensagem deverá ser exibida. Depois disso, é encerrada a execução da aplicação por 5000 milisegundo, usando do módulo *time* e da função *wait*, e, só então, a aplicação é fechada.

```
def fimDeJogo(pontuacaoFinal):
   mensagemFimJogo = fonteFimJogo.render(
   "Fim De Jogo!", True, (255, 255, 0))
   ultimaPontuacao = fontePontos.render(
   f'Pontuação Total: {pontuacaoFinal}', True, (255, 255, 0))
   tela.blit(mensagemFimJogo, [80, 140])
   tela.blit(ultimaPontuacao, [120, 190])
   pygame.display.update()
   pygame.time.wait(5000)
   pygame.quit()
   exit()
```

4.4. Laço de Repetição Principal

Utiliza-se de um laço de repetição *while* para controlar a execução da aplicação. Os eventos, funções e verificações condicionais estarão presentes dentro deste laço de repetição.

4.4.1. Eventos

Geralmente utiliza-se de um laço *for* junto ao módulo *event* e a função *get* para buscar eventos como cliques ou teclas pressionadas durante a execução do laço principal.

```
for event in pygame.event.get():
```

Diante disso, ações são atribuídas aos eventos, tais como alterar as constantes que controlam a direção da serpente através do evento KEYDOWN.

```
if event.type == KEYDOWN:
```

Se alguma das teclas *w*, *s*, *a* ou *d* for pressionada no teclado e a direção da serpente for diferente da sua direção contrária, a variável direcaoSerpente, deverá receber um novo valor.

```
if event.key == K_w and direcaoSerpente != BAIXO:
   direcaoSerpente = CIMA
if event.key == K_s and direcaoSerpente != CIMA:
   direcaoSerpente = BAIXO
if event.key == K_a and direcaoSerpente != DIREITA:
   direcaoSerpente = ESQUERDA
if event.key == K_d and direcaoSerpente != ESQUERDA:
   direcaoSerpente = DIREITA
```

Com base no novo valor da variável direcaoSerpente proveniente dos eventos, as coordenadas da serpente aumentarão ou diminuirão em seu eixo X ou eixo Y, fazendo então com que a serpente seja desenhada em uma nova coordenada a cada repetição do laço principal.

```
if event.key == K_w and direcaoSerpente != BAIXO:
    direcaoSerpente = CIMA
if event.key == K_s and direcaoSerpente != CIMA:
    direcaoSerpente = BAIXO
if event.key == K_a and direcaoSerpente != DIREITA:
    direcaoSerpente = ESQUERDA
if event.key == K_d and direcaoSerpente != ESQUERDA:
    direcaoSerpente = DIREITA
```

4.4.2. Colisões

É preciso verificar se a coordenada do seu eixo X ou eixo Y são iguais aos limites da tela criada, ou menores que zero. Caso isso ocorra, deve-se retornar um valor verdadeiro e a função fimDeJogo() deve ser chamada, passando a pontuação do jogador até aquele momento.

```
if Serpente[0][0] == 400 or Serpente[0][1] == 400 or
   Serpente[0][0] < 0 or Serpente[0][1] < 0:
      fimDeJogo(pontos)
```

Pode-se verificar se o eixo X e o eixo Y da serpente estão ocupando a mesma coordenada do contador. Em caso positivo, a função fimDeJogo() deve ser chamada.

```
for contador in range(1, len(Serpente) - 1):
    if Serpente[0][0] == Serpente[contador][0] and
       Serpente[0][1] == Serpente[contador][1]:
         fimDeJogo(pontos)
```

4.4.3. Exibindo na tela

Para exibir a serpente na tela, cria-se um laço *for* que vai percorrer toda a serpente (visto que ela é uma lista) e a exibirá na tela (cada posição da lista) usando a função *blit* que recebe como parâmetro a superfície e a posição que é passada pelo próprio laço.

```
for pos in Serpente:
    tela.blit(peleSerpente, pos)
```

Para exibir a pontuação do jogador durante a execução da aplicação, deve-se usar a função *render* para renderizar a fonte e a mensagem da pontuação e então apresentá-la na tela usando a função *blit*. Nesta última função, geralmente é passado como parâmetro a mensagem renderizada e a coordenada que se deseja que a pontuação fique durante a execução da aplicação.

```
    pontuacao = fontePontos.render(f'Pontuação: {pontos}',
          True, (255, 255, 0))
    tela.blit(pontuacao, [0, 0])
```

Para exibir a maçã na tela, usa-se a função *blit*, passando a superfície da maçã e a coordenada que ela deve aparecer.

```
    conteúdo.tela.blit(maca, posicaoMaca)
```

Para auxiliar as exibições de objetos na tela, pode-se usar as funções *fill* e *update*. Estas funções limpam e atualizam a tela a cada repetição do laço principal, dando sentido de movimento aos objetos.

```
    tela.fill((0, 0, 0))
    pygame.display.update()
```

A Figura 2 demonstra a tela da aplicação em execução com base nos códigos acima.

Figura 2. Tela do jogo em execução (na esquerda) e tela de fim de jogo (direita).

5. Conclusão

Como resultado, obtém-se uma tela 400x400 de tamanho. Ao desenhar na tela, tem-se a serpente que pode mudar de direção a partir da entrada de dados do teclado e crescer sempre que colidir com a maçã. As maçãs são desenhadas em posições aleatórias sempre que for atingida pela serpente e a pontuação do jogador vai ser atualizada sempre que a serpente colidir com a maçã. A Figura 3 demonstra um exemplo de possível resultado. A figura apresenta o jogo e seus "personagens", a cobra (retângulo verde), a maçã (quadrado vermelho) e a pontuação do usuário. O código completo da implementação do jogo pode ser encontrada em `https://github.com/Luis356/Pygames`.

Figura 3. Exemplo de possível resultado

Referências

PYGAME ORG (2022a). Pygame display. https://www.pygame.org/docs/ref/display.html. Acessado em: 2022-10-10.

PYGAME ORG (2022b). Pygame documentation. http://www.pygame.org/docs/. Acessado em: 2022-10-10.

PYGAME ORG (2022c). Pygame event. https://www.pygame.org/docs/ref/event.html. Acessado em: 2022-10-10.

PYGAME ORG (2022d). Pygame font. https://www.pygame.org/docs/ref/font.html. Acessado em: 2022-10-10.

PYGAME ORG (2022e). Pygame surface. http://www.pygame.org/docs/ref/surface.html. Acessado em: 2022-10-10.

PYGAME ORG (2022f). Pygame time. https://www.pygame.org/docs/ref/time.html. Acessado em: 2022-10-10.

PYGAME ORG (2022g). Python pygame introduction. https://www.pygame.org/docs/tut/PygameIntro.html. Acessado em: 2022-10-10.

Wikipedia (2022a). Blockade (video game). https://en.wikipedia.org/wiki/Blockade_(video_game). Acessado em: 2022-10-10.

Wikipedia (2022b). Nibbler (video game). https://en.wikipedia.org/wiki/Nibbler_(video_game). Acessado em: 2022-10-10.

Wikipedia (2022c). Snake (video game genre). https://en.wikipedia.org/wiki/Snake_(video_game_genre). Acessado em: 2022-10-10.