****

**UNIVERSIDADE PAULISTA**

**CIÊNCIA DA COMPUTAÇÃO**

**CAMPUS PARAÍSO**

|  |  |
| --- | --- |
| FELIPE SANTOS MORAES | **RA:** N4039G6 |
|  |  |
|  |  |
|  |  |

**Logica de Programação Orientada a Objetos**

**São Paulo - SP**

**Abril/2022**

FELIPE SANTOS MORAES

**Jogo**

Trabalho de aprovação do Semestre referente a disciplina Atividades Práticas Supervisionadas.

Apresentado à Universidade Paulista – UNIP.

**São Paulo - SP**

**2022**

1. **Objetivo e motivação do trabalho**
2. **Introdução**
3. **Regras do jogo (conceitos gerais)**
4. **Plano de desenvolvimento do jogo (elementos e ferramentas que serão utilizadas)**
5. **Projeto (estrutura e módulos que serão desenvolvidos) do programa**
6. **Relatório com as linhas de código do programa**
7. **Bibliografia**
8. **Ficha de Atividades Práticas Supervisionadas**
9. **Objetivo e Motivação do Trabalho**

Objetivo do trabalho é criar um jogo com interface gráfica utilizando a linguagem de programação Python fazendo uso da programação orientada a objetos.

Foi necessário a utilização do Python por achar mais fácil fazer a criação de um jogo utilizando a biblioteca Pygame, e por ter mais familiaridade com a linguagem e sintaxe.

Foi idealizado um jogo de conscientização ambiental voltado as tartarugas e os canudos onde foi utilizado como base um jogo muito famoso chamado Flappy Bird.

O projeto deste jogo, não tem somente a finalidade para este propósito, mas também podendo visar futuros projetos de desenvolvimento de jogos e demostrando seu foco de desenvolvimento, e como esta área da computação está atualmente situada no mercado de trabalho atual.

Pygame é uma biblioteca escrita em Python e baseada em SDL

Voltada para o desenvolvimento de games e interfaces gráficas, o Pygame fornece acesso a áudios, teclados, controles, mouses e hardwares gráficos via OpenGL e Direct3D.

Por serem multiplataformas, tanto a SDL quanto o Pygame, podem rodar em quaisquer sistemas operacionais com alterações mínimas de código no funcionamento de um ou outro.

**2. Introdução**

Nos dias de hoje, a programação não está largamente vinculada somente a criação de softwares corporativos, mas também no desenvolvimento de jogos com fins interativos como, por exemplo, a nível educacional para ampliar o aprendizado ou até mesmo por diversão.

Atualmente, é possível realizar a conexão de informações já adquiridas no passado e reintroduzi-las na era tecnológica, buscando mais aprimoramento na forma de transmitir as mesmas informações de uma maneira mais concisa, podendo assim gerar, mais interesse do público em questão sobre tais assuntos, com técnicas de aperfeiçoamento através da programação de computadores.

A intenção do trabalho semestral desenvolvido em questão é, apresentar ao usuário uma experiência de interação com o meio ambiente e uma possível conscientização ecológica, com a finalidade de passar informações de forma intuitiva sobre assuntos como recursos naturais, sustentabilidade e reciclagem. Visando o efeito positivo de cada um na sociedade e a interação humana sobre cada respectivo assunto.

**3. Regras do jogo (conceitos gerais)**

As regras são do jogo são simples, você deverá apenas utilizar da key space para movimentar o personagem (Tartaruga), e caso você deixe cair ou encoste em um dos canudos terá um game over, já que o objetivo seria passar entre os canudos que tem uma certa abertura para que caiba o personagem.

Utilizou class para a criação dos ambientes tal como base, background e a própria tartaruga. Com a biblioteca Pygame foi possível criar o movimento e todo aspecto parecido com o Flappy Bird. As imagens foram retiradas do google imagens já que não há conhecimento em criação de base através de alguma ferramenta gráfica.

Foi-se adequado o máximo possível para que o jogo tivesse um resultado agradável e interativo.

A finalidade é para mostrar que os canudos matam as tartarugas e tentar conscientizar a todos sobre esse aspecto que é tão importante para o meio ambiente.

**4. Plano de desenvolvimento do jogo (elementos e ferramentas que serão utilizadas).**

Foi desenvoldido em Python e utilizado a biblioteca com a finalidade de transmitir algo muito importante que seria o alerta que temos para a não utilização de canudos plásticos nas praias para que possamos conservar os mares e não atingir os animais marinhos indefesos que acabam por comer esses plásticos que fazem mal e podem causar a morte. O jogo tem a finalidade de ser meramente educativo não trazendo finalidade competitiva ou de vicio para conseguir pontuação máxima.

**5. Projeto (estrutura e módulos que serão desenvolvidos) do programa**

Padrão do plano de fundo

Descrição gerada automaticamente

Tela inicial do jogo, onde temos que utilizar a barra de espaço para movimentar o personagem.

Padrão do plano de fundo

Descrição gerada automaticamente

Nossos canudos que ao se tocar pode se ter o game over do jogo.

**6. Relatório com as linhas de código do Programa**

**import pygame, random**

**from pygame.locals import \***

**SCREEN\_WIDTH = 500**

**SCREEN\_HEIGHT = 800**

**SPEED = 10**

**GRAVITY = 1**

**GAME\_SPEED = 10**

**GROUND\_WIDTH = 2 \* SCREEN\_WIDTH**

**GROUND\_HEIGHT = 20**

**PIPE\_WIDTH = 80**

**PIPE\_HEIGHT = 500**

**PIPE\_GAP = 200**

**SCORE = 0**

**class Turtle(pygame.sprite.Sprite):**

**def \_\_init\_\_(self):**

**pygame.sprite.Sprite.\_\_init\_\_(self)**

**self.images = pygame.image.load('turtle.png').convert\_alpha(),**

**self.speed = SPEED**

**self.current\_image = 0**

**self.image = pygame.image.load('turtle.png').convert\_alpha()**

**self.mask = pygame.mask.from\_surface(self.image)**

**self.rect = self.image.get\_rect()**

**self.rect[0] = SCREEN\_WIDTH / 2**

**self.rect[1] = SCREEN\_HEIGHT / 2**

**def update(self):**

**self.image = self.images[ self.current\_image ]**

**self.speed += GRAVITY**

**# Update height**

**self.rect[1] += self.speed**

**def bump(self):**

**self.speed = -SPEED**

**class Pipe(pygame.sprite.Sprite):**

**def \_\_init\_\_(self, inverted, xpos, ysize):**

**pygame.sprite.Sprite.\_\_init\_\_(self)**

**self.image = pygame.image.load('canudo.png').convert\_alpha()**

**self.image = pygame.transform.scale(self.image, (PIPE\_WIDTH, PIPE\_HEIGHT))**

**self.rect = self.image.get\_rect()**

**self.rect[0] = xpos**

**if inverted:**

**self.image = pygame.transform.flip(self.image, False, True)**

**self.rect[1] = - (self.rect[3] - ysize)**

**else:**

**self.rect[1] = SCREEN\_HEIGHT - ysize**

**self.mask = pygame.mask.from\_surface(self.image)**

**def update(self):**

**self.rect[0] -= GAME\_SPEED**

**class Ground(pygame.sprite.Sprite):**

**def \_\_init\_\_(self, xpos):**

**pygame.sprite.Sprite.\_\_init\_\_(self)**

**self.image = pygame.image.load('base.png').convert\_alpha()**

**self.image = pygame.transform.scale(self.image, (GROUND\_WIDTH, GROUND\_HEIGHT))**

**self.mask = pygame.mask.from\_surface(self.image)**

**self.rect = self.image.get\_rect()**

**self.rect[0] = xpos**

**self.rect[1] = SCREEN\_HEIGHT - GROUND\_HEIGHT**

**def update(self):**

**self.rect[0] -= GAME\_SPEED**

**def is\_off\_screen(sprite):**

**return sprite.rect[0] < -(sprite.rect[2])**

**def get\_random\_pipes(xpos):**

**size = random.randint(100, 300)**

**pipe = Pipe(False, xpos, size)**

**pipe\_inverted = Pipe(True, xpos, SCREEN\_HEIGHT - size - PIPE\_GAP)**

**return (pipe, pipe\_inverted)**

**pygame.init()**

**screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT))**

**BACKGROUND = pygame.image.load('Background\_BlueOcean1.png')**

**BACKGROUND = pygame.transform.scale(BACKGROUND, (SCREEN\_WIDTH, SCREEN\_HEIGHT))**

**turtle\_group = pygame.sprite.Group()**

**turtle = Turtle()**

**turtle\_group.add(turtle)**

**ground\_group = pygame.sprite.Group()**

**for i in range(2):**

**ground = Ground(GROUND\_WIDTH \* i)**

**ground\_group.add(ground)**

**pipe\_group = pygame.sprite.Group()**

**for i in range(2):**

**pipes = get\_random\_pipes(SCREEN\_WIDTH \* i + 600)**

**pipe\_group.add(pipes[0])**

**pipe\_group.add(pipes[1])**

**pontos\_group = pygame.sprite.Group()**

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

**while True:**

**clock.tick(30)**

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

**if event.type == QUIT:**

**pygame.quit()**

**if event.type == KEYDOWN:**

**if event.key == K\_SPACE:**

**turtle.bump()**

**screen.blit(BACKGROUND, (0, 0))**

**if is\_off\_screen(ground\_group.sprites()[0]):**

**ground\_group.remove(ground\_group.sprites()[0])**

**new\_ground = Ground(GROUND\_WIDTH - 20)**

**ground\_group.add(new\_ground)**

**if is\_off\_screen(pipe\_group.sprites()[0]):**

**pipe\_group.remove(pipe\_group.sprites()[0])**

**pipe\_group.remove(pipe\_group.sprites()[0])**

**pipes = get\_random\_pipes(SCREEN\_WIDTH \* 2)**

**pipe\_group.add(pipes[0])**

**pipe\_group.add(pipes[1])**

**turtle\_group.update()**

**ground\_group.update()**

**pipe\_group.update()**

**turtle\_group.draw(screen)**

**pipe\_group.draw(screen)**

**ground\_group.draw(screen)**

**pygame.display.update()**

**if (pygame.sprite.groupcollide(turtle\_group, ground\_group, False, False, pygame.sprite.collide\_mask) or**

**pygame.sprite.groupcollide(turtle\_group, pipe\_group, False, False, pygame.sprite.collide\_mask)):**

**# Game Over**

**break**

**pygame.font.init()**

**7. Referências Bibliográficas**

<https://www.pygame.org/docs/ref/pygame.html>

<https://acervolima.com/como-fazer-o-jogo-flappy-bird-no-pygame/>

<https://github.com/filhoweuler/python-flappybird/blob/9e572fc3c141b530e74953bf3344f7d454d7deac/flappy.py#L8>

<https://github.com/pyladiesdf/flappy-tutorial/blob/master/flappy.py>

<https://kenzie.com.br/blog/pygame/>