STEM²D

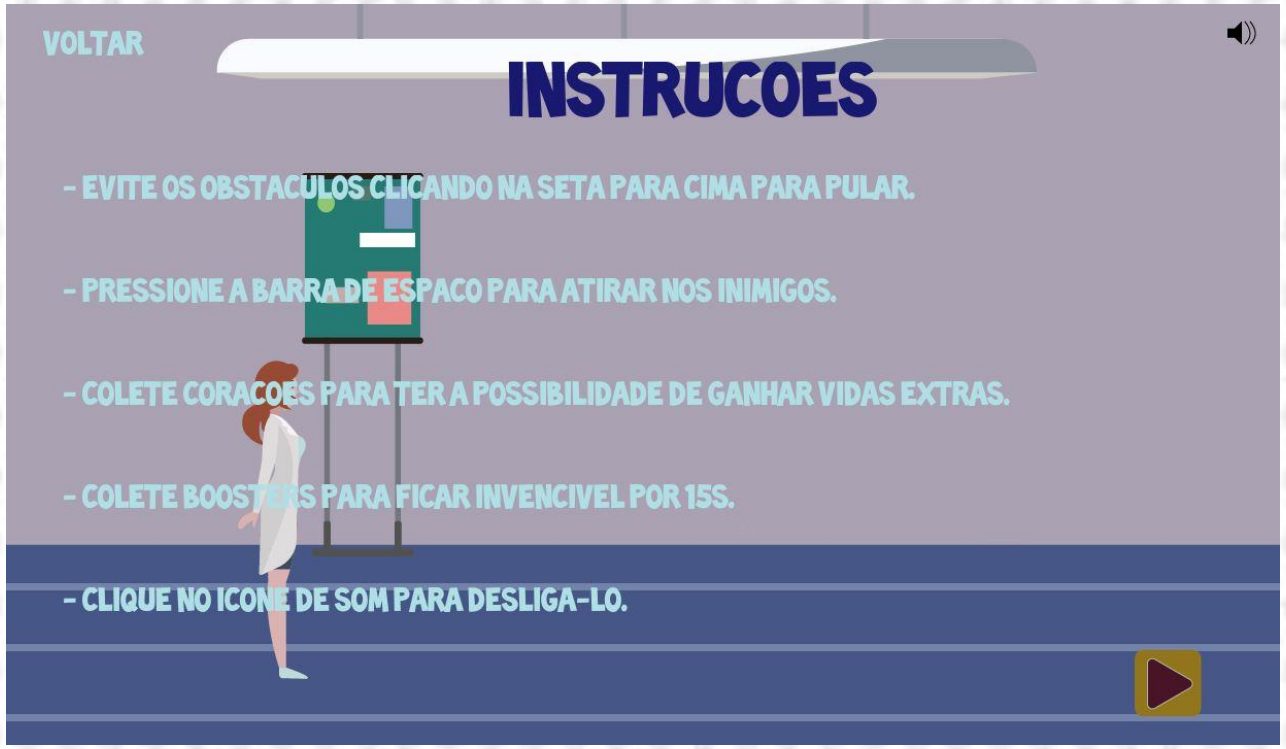
Go girls!

CODE

like a girl

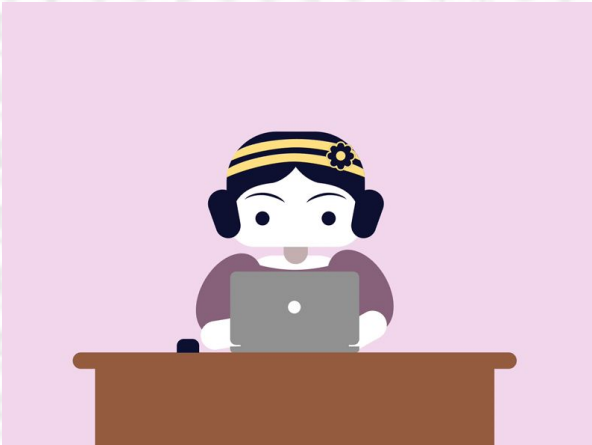
TELA DE INSTRUÇÕES

- Nessa etapa, vamos criar uma tela de instruções de jogo, para auxiliar o usuário a entender o funcionamento do jogo.



TELA DE INSTRUÇÕES

- Primeiramente, inicializaremos os atributos (variáveis do tipo **self**) da classe **TelaDeInstrucoes** (arquivo **TelaDeInstrucoes.py**) que guardarão as fontes utilizadas, os textos das instruções, o botão de **play** e o botão de voltar.



TELA DE INSTRUÇÕES

- A inicialização da classe **TelaDeInstrucoes** (arquivo **TelaDeInstrucoes.py**) é realizada com a declaração do método **__init__** , chamado de construtor. Nele, devemos:
  - Declarar o nome desta tela;
  - Carregar as fontes que serão utilizadas nos textos;
  - Declarar o texto do título;
  - Declarar o texto do botão de voltar;
  - Declarar a imagem do botão de jogar.

TELA DE INSTRUÇÕES

```
class TelaDeInstrucoes(Tela):
   def __init__(self):
        # declaracao  do  construtor  da classe -pai  Tela
       super().__init__()

       # definindo o nome da tela
       self.name = "Tela de Instrucoes"

       # carregando  as  fontes
       self.fonte1 = self.font = pygame.font.Font(os.path.join('Fontes', 'TOONISH.ttf'), 70)
       self.fonte2 = self.font = pygame.font.Font(os.path.join('Fontes', 'TOONISH.ttf'), 30)

       # carregando o titulo da tela
       self.titulo = fonte1.render('INSTRUCOES', True, NAVY)

       # carregando a mensagem 'VOLTAR' para o botao voltar
       self.botaoVoltar = self.fonte2.render('VOLTAR', True, AZULBB)

       # carregando a imagem do botao play
       self.botaoPlay = self.play = pygame.image.load(os.path.join('Imagens', 'play_1.png'))
```

TELA DE INSTRUÇÕES

- Implementaremos um método auxiliar para imprimir os textos das instruções de forma igualmente espaçada.

- Este método será denominado **imprimirInstrucoes** e terá como argumentos o administrador do jogo (**game**), o número da instrução (**num**) e o texto da instrução (**tex**).

TELA DE INSTRUÇÕES

- Este código deve ser transcrito para o seu jogo!

```
def imprimirInstrucoes(self, game, num, text):
    # carrega a instrucao com a fonte 2 e a cor azul bebe
    instrucao = self.fonte2.render(text, True, AZULBB)

    # posiciona a instrucao na tela
    game.janela.blit(instrucao, (70, 170 + 100*(num - 1)))
```

- Note que a instrução é posicionada com a sua coordenada x (posição horizontal) valendo 70 *pixels*.
- Além disso, a sua coordenada y (posição vertical) varia de acordo com a seguinte função: **170 + 100*(num - 1)**.

TELA DE INSTRUÇÕES

- Perceba que, utilizando esta função, a primeira instrução (**num = 1**) será posicionada com coordenada y igual a **170 + 100*(1 - 1) = 170**, já a segunda instrução (**num = 2**) será posicionada com coordenada y igual a **170 + 100*(2 - 1) = 170 + 100 = 270**, e assim por diante.

- Como usamos sempre o mesmo tamanho de fonte, as instruções aparecerão na tela igualmente espaçadas na vertical.

TELA DE INSTRUÇÕES

- É importante lembrar que não basta implementar o método **imprimirInstrucoes**, é preciso chamá-lo em algum lugar para que ele, efetivamente, funcione.

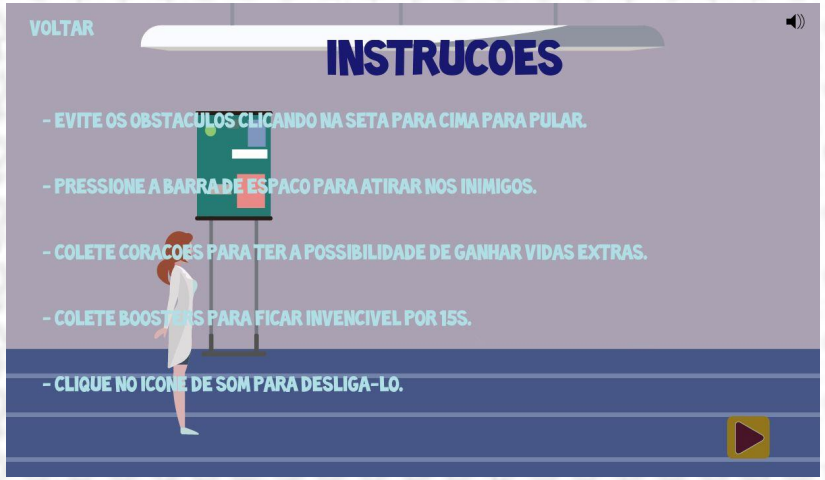
- Para imprimir as instruções na tela deveremos chamá-lo dentro do método **desenharTela**, da seguinte forma:

self.imprimirInstrucoes(game, <numero da instrucao>, <string com o texto da instrucao>**)**

```
# imprimindo a primeira instrucao do jogo com o auxilio do metodo imprimirInstrucoes
self.imprimirInstrucoes(game, 1, '- Evite os obstaculos clicando na seta para cima para pular.')
```

TELA DE INSTRUÇÕES

- O segundo método que devemos implementar na classe **TelaDeInstrucoes** é o **desenharTela** que consiste em desenhar os elementos que inicializamos no construtor da tela, isto é, desenhar o **botaoPlay**, o **botaoVoltar**, o **titulo** e os textos das instruções, com o auxílio do método **imprimirInstrucoes**.



EXERCÍCIO

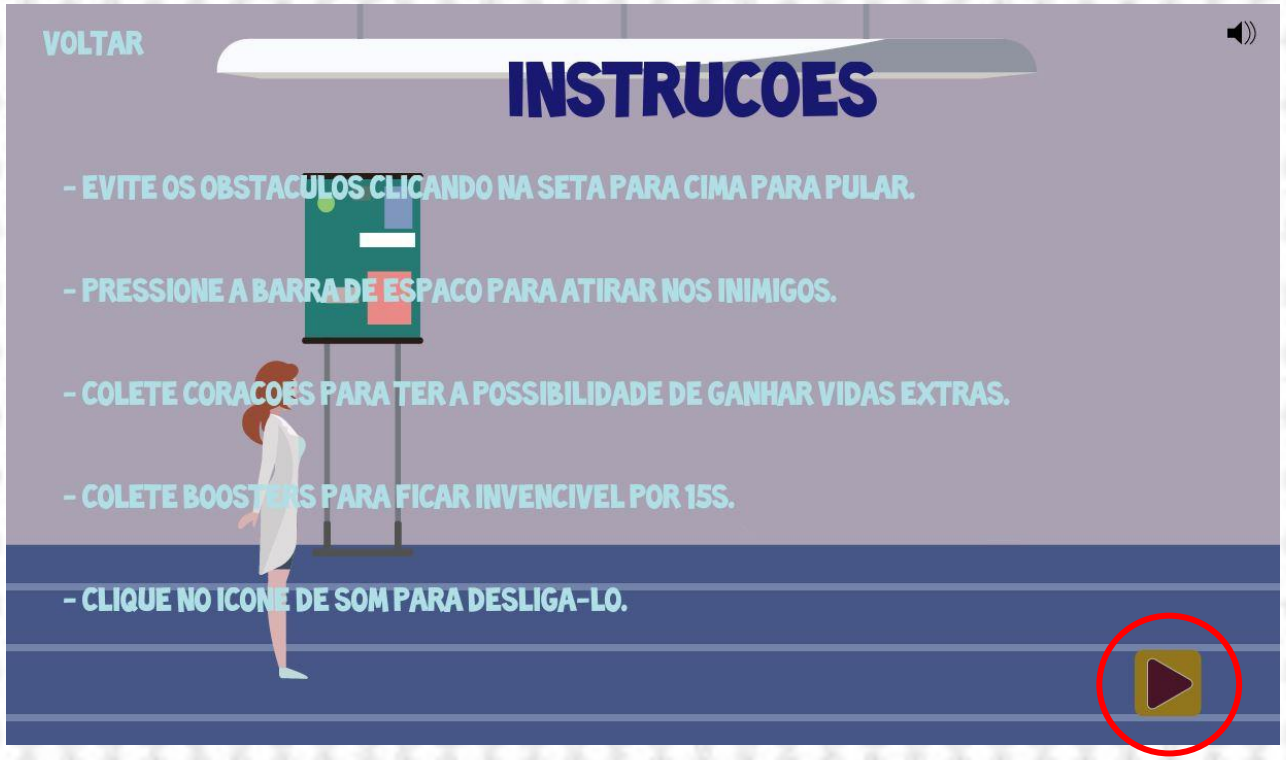
- Implemente o método **desenharTela(self, game)** da tela de instruções. Não esqueça de desenhar o **botaoPlay**, o **botaoVoltar**, o **titulo** e os textos das instruções, com o auxílio do método **imprimirInstrucoes**. **Dicas:**

```
def desenharTelaBasica(self, game):
    # adicionando o botão do som na tela utilizando a função blit que requer como argumentos o nome da variavel da imagem do botao, bem como as coordenadas (x, y) em que voce deseja posiciona-la
    game.janela.blit(self.botaoSom, (1200, 20))
```

```
# imprimindo a primeira instrucao do jogo com o auxilio do metodo imprimirInstrucoes
self.imprimirInstrucoes(game, 1, '- Evite os obstaculos clicando na seta para cima para pular.')
```

TELA DE INSTRUÇÕES

- Dentro da tela de instruções, há um botão **play**, para que o usuário possa jogar logo após ler as instruções.



TELA DE INSTRUÇÕES

- O botão **play** da tela de instruções funciona conforme o botão **play** da tela de início: quando o cursor do *mouse* estiver nos limites da imagem do botão, trocamos a imagem do botão para aquela em que ele possui brilho diferente (*replay_brilho_X.png*) e se o usuário clicar na imagem do botão, trocamos a tela atual para a Tela de Jogo.

EXERCÍCIO

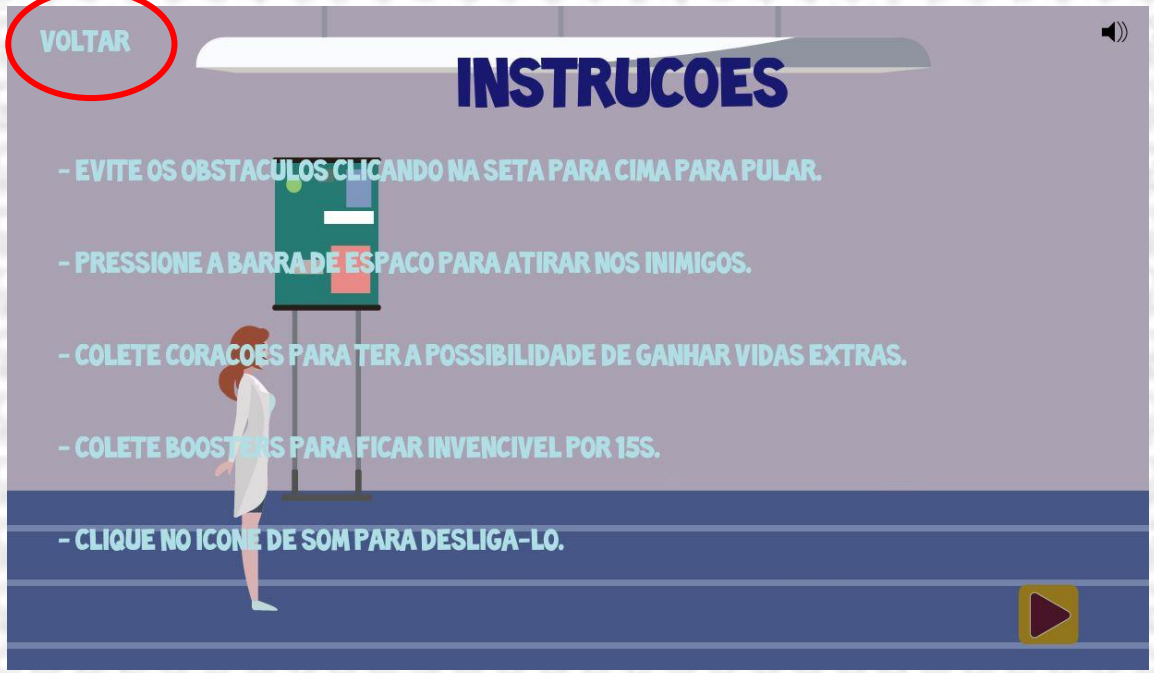
- Implemente o método **comportamentoBotaoDeJogar(self,game,evento,pos)** da tela de instruções. Utilize como base o código **comportamentoBotaoDeJogar** da tela de início:

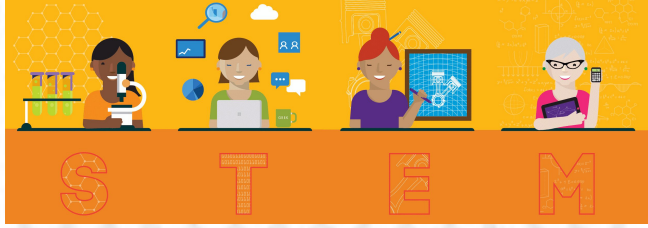
```
def comportamentoBotaoDeJogar(self, game, evento, pos):
    # verifica se o usuario esta com o cursor em cima da imagem do botao
    if pos[0]>550 and pos[0]<615 and pos[1]>365 and pos[1]<430:
        # verifica se o usuario clicou no botao
        if evento.type == pygame.MOUSEBUTTONDOWN:
            game.ultimaTela = 'Tela de Inicio'
            game.telaAtual = 'Tela de Jogo'
        # se o usuario nao clicou no botao, apenas muda a imagem do botao para outra com brilho diferente imagem
        else:
            self.botaoPlay = self.play = pygame.image.load(os.path.join('Imagens', 'play_brilho_4.png'))
    # se o usuario nao esta com o cursor em cima do botao, retorna a imagem do botao para a imagem regular
    else:
        self.botaoPlay = self.play = pygame.image.load(os.path.join('Imagens', 'play_4.png'))
```

CODE LIKE A GIRL

TELA DE INSTRUÇÕES

- Dentro da tela de instruções, há um botão **voltar**, representado pelo texto '**VOLTAR**', com o intuito de viabilizar que o usuário retorne para a tela de início do jogo.





TELA DE INSTRUÇÕES

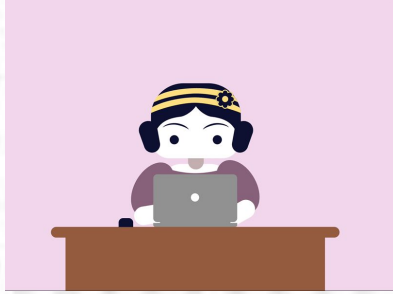
- O botão **voltar** da tela de instruções funciona da seguinte forma: quando o jogador colocar o cursor sobre a palavra '**VOLTAR**', essa palavra deve adquirir uma cor amarela (como fizemos com o botão de instruções da classe **TelaDeInicio**).

- Ademais, quando o cursor sair da área correspondente à palavra '**VOLTAR**', o ícone deve voltar a ter sua cor original. Por último, se o usuário clicar na palavra '**VOLTAR**', devemos redirecionar a tela atual do jogo para a Tela de Inicio.

EXERCÍCIO

- Implemente o método **comportamentoBotaoVoltar(self,game,evento,pos)** da tela de instruções. Utilize como base o código **comportamentoBotaoDeInstrucoes** da tela de início:

```
def comportamentoBotaoDeInstrucoes(self, game, evento, pos):
    # verifica se o usuario esta com o cursor em cima da imagem do botao
    if pos[0]>400 and pos[0]<760 and pos[1]>540 and pos[1]<650:
        # verifica se o usuario clicou no botao
        if evento.type == pygame.MOUSEBUTTONDOWN:
            game.ultimaTela = 'Tela de Inicio'
            game.telaAtual = 'Tela de Instrucoes'
        # se o usuario nao clicou no botao, apenas muda a cor da imagem do botao
        else:
            self.inst = self.fonte2.render ('INSTRUÇÕES', True, AMARELO)
    # se o usuario nao esta com o cursor em cima do botao, retorna a cor do botao para a original
    else:
        self.inst = self.fonte2.render ('INSTRUÇÕES', True, BRANCO)
```



ELEMENTOS DO CENÁRIO

- Todos os elementos do cenário do jogo são classes derivadas da classe **Cenario**. Essa classe possui três métodos comuns a todos os elementos do cenário do jogo. O primeiro método é o seu construtor que possui quatro argumentos:
  - **x**: coordenada x em que se deseja posicionar o elemento na tela
  - **y**: coordenada y em que se deseja posicionar o elemento na tela;
  - **imagem**: imagem do elemento;
  - **vel**: "velocidade" com que o elemento se moverá na tela;

#girlsgonna_

ELEMENTOS DO CENÁRIO

- O construtor também é responsável por calcular as variáveis **self.largura** e **self.altura** da imagem do personagem, bem como gerar o retângulo de colisões do elemento.

```
class Cenario:
    def __init__(self, x, y, imagem, vel):
        self.x = x
        self.y = y
        self.image = imagem
        self.largura = imagem.get_width()
        self.altura = imagem.get_height()
        self.rect = pygame.Rect(self.x, self.y, self.largura, self.altura) # retangulo de colisoes
        self.vel = vel  # velocidade de atualizacao horizontal na tel
```

ELEMENTOS DO CENÁRIO

- Para movimentar os elementos, a classe **Cenario** possui o método **atualizacaoBasica**, o qual é responsável por atualizar a posição horizontal dos elementos e de seus retângulos de colisão na tela.

- Este código deve ser transcrito para o seu jogo!

```
# atualizacaoBasica eh o metodo que descreve como o item do cenario tem sua posicao horizontal atualizada na tela
def atualizacaoBasica(self):
    if self.x > -self.largura:
        self.x -= self.vel
    self.rect.x = self.x
    self.rect.y = self.y
```

ELEMENTOS DO CENÁRIO

- Para desenhar na tela os elementos, a classe **Cenario** possui o método **desenhar**, o qual é responsável por adicionar o elemento do cenário na janela de jogo.

```
# desenha o item do cenario na tela
    def desenhar(self, game):
        game.janela.blit(self.image, (self.x, self.y))
        # pygame.draw.rect(game.janela, (255, 0, 0), self.rect, 2)
```

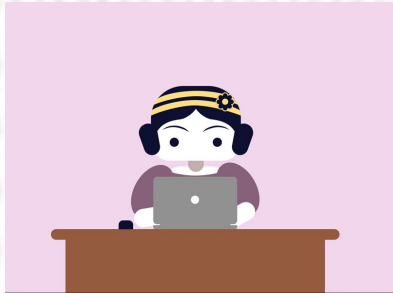
APARECIMENTO DO CENÁRIO

- Para implementar o aparecimento dos elementos do cenário do jogo, utilizaremos o método **criarCenario** da classe **TelaDeJogo**.

- O aparecimento de elementos no cenário será feito de forma aleatória, com o auxílio da função **random**.

APARECIMENTO DO CENÁRIO

- A ideia aqui será atribuir para uma variável um número aleatório entre 0 e determinado valor. Então, dependendo do valor que esta variável inteira assumir, por meio da utilização de expressões condicionais, será possível selecionar qual elemento do cenário será criado.



APARECIMENTO DO CENÁRIO

- Este código deve ser transcrito para o seu jogo!

```
# cria itens do cenario na tela
def criarCenario(self, game):
    # geracao de numero aleatorio
    # game.aparecimentoElementos eh uma constante que vale, inicialmente, 50
    r = random.randrange(0, game.aparecimentoElementos)
    # se o numero gerado for menor que 8
    if r < 8:
        if len(game.obstaculos) == 0 or (len(game.obstaculos) != 0 and game.obstaculos[-1].x +
game.obstaculos[-1].largura + self.tolerancia < LARGURA_DA_TELA):
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 5
             game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 5))
```

APARECIMENTO DO CENÁRIO

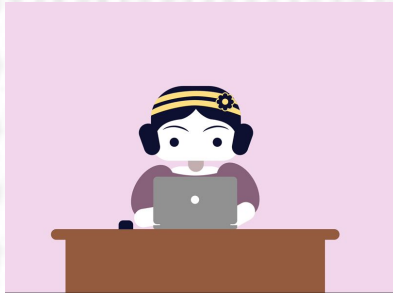
- Na linha 8, a primeira parte da condição verifica a tentativa de se inserir um obstáculo no jogo quando não há nenhum outro obstáculo.

- Se isso for verdade, é possível inserir o obstáculo no jogo, sem precisar se preocupar com a sobreposição de imagem deste com a de outro obstáculo.

APARECIMENTO DO CENÁRIO

- Além disso, na linha 8, a segunda parte da condição verifica a tentativa de se inserir um obstáculo no jogo quando já existem outros obstáculos e evita possíveis sobreposições de imagens.

- Isso porque os elementos do cenário são criados com coordenada x igual a **LARGURA_DA_TELA** e se o último obstáculo inserido não estiver completamente na tela, o outro obstáculo será criado em cima dele, sobrepondo as imagens, o que não é bom.



APARECIMENTO DO CENÁRIO

- A expressão apresentada logo após o **and** presente na linha 8, em que **game.obstaculos[-1]** acessa o último elemento do vetor de obstáculos do jogo, **game.obstaculos[-1].x + game.obstaculos[-1].largura** calcula a coordenada x da extremidade direita da imagem do obstáculo e **self.tolerancia** representa o mínimo espaçamento desejado entre um obstáculo e outro.

APARECIMENTO DO CENÁRIO

- Para deixar o jogo mais interessante, você pode implementar o aparecimento de um obstáculo que se move com alta velocidade quando a tela não possui nenhum outro obstáculo.

```
# cria itens do cenario na tela
def criarCenario(self, game):
    # geracao de numero aleatorio
    # game.aparecimentoElementos eh uma constante que vale, inicialmente, 50
    r = random.randrange(0, game.aparecimentoElementos)
    # se o numero gerado for menor que 8
    if r < 8:
        if len(game.obstaculos) == 0:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 10
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 10))
        elif game.obstaculos[-1].x + game.obstaculos[-1].largura + self.tolerancia < LARGURA_DA_TELA:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 5
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 5))
```

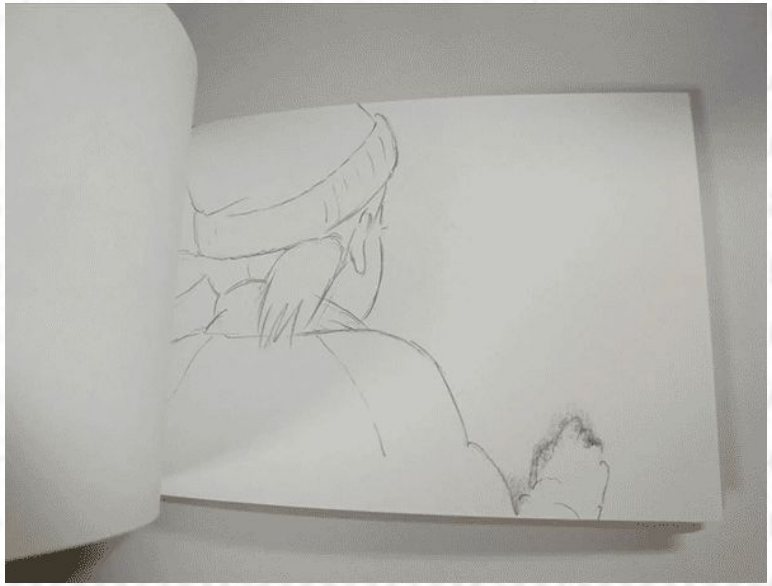
APARECIMENTO DO CENÁRIO

- Qualquer ação que visualizamos em filmes, vídeos ou jogos é composta por **quadros**.

- Eles são nada mais do que imagens sequenciais que, ao serem reproduzidas em velocidade, dão a sensação de movimento.

APARECIMENTO DO CENÁRIO

- É possível fazer um teste simples em casa para entender como funcionam os vídeos. Tente desenhar várias figuras similares em um bloco e em cada folha deslocá-las um pouco. Ao folhear o bloco, você terá impressão que a figura está em movimento.



APARECIMENTO DO CENÁRIO

- No nosso jogo, o **FPS** é de 60 quadros por segundo o que significa, efetivamente, que o código do jogo é executado 60 vezes por segundo. Essa taxa de **FPS** está declarada no arquivo **Configuracoes.py**.

APARECIMENTO DO CENÁRIO

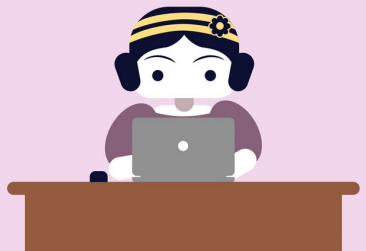
- Para que o aparecimento de obstáculos ocorra efetivamente, é preciso chamar o método **criarCenario** no método que efetivamente executa a tela de jogo, isto é, no método **run** da classe **TelaDeJogo**.

- Não obstante, é preciso lembrar que o método **run** é executado conforme a taxa de quadros por segundo do jogo. Como a taxa **FPS** é de 60 quadros por segundo, o código é executado 60 vezes por segundo.

APARECIMENTO DO CENÁRIO

- Perceba, todavia, que se chamarmos o método **criarCenario** 60 vezes por segundo, a taxa de aparecimento de objetos será muito grande.

- Por causa disso, a chamada deste método deve estar dentro de uma condição que só é satisfeita poucas vezes por segundo



APARECIMENTO DO CENÁRIO

- Este código deve ser transcrito para o seu jogo!

```
def run(self, game):
    self.tempo = 1

    while game.telaAtual == self.name and not game.usuarioSaiu:
        if self.tempo % 30 == 0:
            self.criarCenario(game)

        self.interpretarEventos(game)
        self.checarColisoes(game)
        self.atualizar(game)
        self.desenhar(game)

        self.tempo += 1
```

EXERCÍCIO

- Implemente o aparecimento do segundo tipo de obstáculo (**obstaculo_X_2.png**) no método **criarCenario** da classe **TelaDeJogo**. **Dica:**

```
# cria itens do cenario na tela
def criarCenario(self, game):
    # geracao de numero aleatorio
    # game.aparecimentoElementos eh uma constante que vale, inicialmente, 50
    r = random.randrange(0, game.aparecimentoElementos)
    # se o numero gerado for menor que 8
    if r < 8:
        if len(game.obstaculos) == 0:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 10
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 10))
        elif game.obstaculos[-1].x + game.obstaculos[-1].largura + self.tolerancia < LARGURA_DA_TELA:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 5
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 5))
```

APARECIMENTO DO CENÁRIO

- Para tornar o aparecimento de elementos no jogo ainda mais interessante, podemos **aumentar a velocidade com a qual os objetos são inicializados** ao longo do tempo, bem como **aumentar a taxa de aparecimento de obstáculos**.

APARECIMENTO DO CENÁRIO

- Para aumentar a taxa de aparecimento de obstáculos, decrementaresmo a variável **game.aparecimentoElementos** no método **run**, de tempos em tempos.

- Essa variável é o limite superior do intervalo de números possíveis de serem gerados pela função random que existe dentro do método **criarCenario**. Desta forma, quanto menor for o intervalo, maior serão as probabilidades de serem gerados números que atendem às condições de aparecimento de elementos do cenário.

APARECIMENTO DO CENÁRIO

- Desta forma, quanto menor for o intervalo, maior serão as probabilidades de serem gerados números que atendem às condições de aparecimento de elementos do cenário.
- Este código deve ser transcrito para o seu jogo!

```
def run(self, game):
    self.tempo = 1

    while game.telaAtual == self.name and not game.usuarioSaiu:
         # aumentar a taxa de aparecimento de elementos do cenario
        if game.aparecimentoElementos > 25 and self.tempo % 300 == 0:
            game.aparecimentoElementos -= 1

        if self.tempo % 30 == 0:
            self.criarCenario(game)

        self.interpretarEventos(game)
        self.checarColisoes(game)
        self.atualizar(game)
        self.desenhar(game)

        self.tempo += 1
```

APARECIMENTO DO CENÁRIO

- Já para aumentar a velocidade com a qual os objetos são inicializados ao longo do tempo, pode-se utilizar uma variável, por exemplo, **game.dvel** que é somada à velocidade inicial dos objetos criados no jogo. Então, no método **run**, incrementa-se o valor desta variável de tempos em tempos.

APARECIMENTO DO CENÁRIO

- Este código deve ser transcrito para o seu jogo!

```
# cria itens do cenario na tela
def criarCenario(self, game):
    # geracao de numero aleatorio
    # game.aparecimentoElementos eh uma constante que vale, inicialmente, 50
    r = random.randrange(0, game.aparecimentoElementos)
    # se o numero gerado for menor que 8
    if r < 8:
        if len(game.obstaculos) == 0:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 10
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 10 + game.dvel))
        elif game.obstaculos[-1].x + game.obstaculos[-1].largura + self.tolerancia < LARGURA_DA_TELA:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 5
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 5 + game.dvel))
```

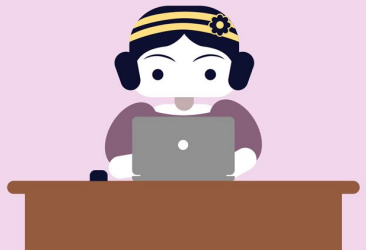
APARECIMENTO DO CENÁRIO

- Este código deve ser transcrito para o seu jogo!

```
def run(self, game):
    self.tempo = 1

    while game.telaAtual == self.name and not game.usuarioSaiu:
        # aumentar a taxa de aparecimento de elementos do cenario
        if game.aparecimentoElementos > 25 and self.tempo % 300 == 0:
            game.aparecimentoElementos -= 1
        # aumentar a velocidade com a qual os elementos do cenario sao inicializados
        if self.tempo % 1200 == 0:
            game.dvel += 1

        if self.tempo % 30 == 0:
            self.criarCenario(game)

        self.interpretarEventos(game)
        self.checarColisoes(game)
        self.atualizar(game)
        self.desenhar(game)

        self.tempo += 1
```

APARECIMENTO DO CENÁRIO

- É importante setar um valor inicial para as variáveis **game.aparecimentoElementos** e **game.dvel** a cada novo jogo, com o intuito de garantir que elas assumirão os valores esperados. Por causa disso, na classe **AdministradorDoJogo** do arquivo **main.py**, faremos as inicializações no método **novoJogo**.



APARECIMENTO DO CENÁRIO

```
# esse metodo inicializa as constantes do jogo a cada novo jogo
    def novoJogo(self):
        self.pontuacao = 0
        self.aparecimentoElementos = 50
        self.vidasExtras = 3
        self.ehInvencivel = False
        self.respostaCorreta = 0
        self.respostaUsuario = 0
        self.tempoDeInvencibilidade = 15
        self.dvel = 0
        self.jogador = Jogador(self)
        self.obstaculos.clear()
        self.inimigos.clear()
        self.vidas.clear()
        self.impulsionadores.clear()
        self.tiros.clear()
        self.tirosInimigo.clear()
```

EXERCÍCIO

- Implemente o aparecimento de vidas (**vida.png**) no método **criarCenario** da classe **TelaDeJogo**. Lembre-se de deixar este aparecimento mais raro que o de obstáculos e restrinja o aparecimento a uma única vida ao longo da tela. **Dica:**

```
# cria itens do cenario na tela
def criarCenario(self, game):
    # geracao de numero aleatorio
    # game.aparecimentoElementos eh uma constante que vale, inicialmente, 50
    r = random.randrange(0, game.aparecimentoElementos)
    # se o numero gerado for menor que 8
    if r < 8:
        if len(game.obstaculos) == 0:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 10
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 10))
        elif game.obstaculos[-1].x + game.obstaculos[-1].largura + self.tolerancia < LARGURA_DA_TELA:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 5
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 5))
```

EXERCÍCIO

- Implemente o aparecimento de impulsionadores (**impulsionador_X.png**) no método **criarCenario** da classe **TelaDeJogo**. Lembre-se de deixar este aparecimento mais raro que o de obstáculos e restrinja o aparecimento a um único impulsionador ao longo da tela. **Dica:**

```
# cria itens do cenario na tela
def criarCenario(self, game):
    # geracao de numero aleatorio
    # game.aparecimentoElementos eh uma constante que vale, inicialmente, 50
    r = random.randrange(0, game.aparecimentoElementos)
    # se o numero gerado for menor que 8
    if r < 8:
        if len(game.obstaculos) == 0:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 10
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 10))
        elif game.obstaculos[-1].x + game.obstaculos[-1].largura + self.tolerancia < LARGURA_DA_TELA:
            # adiciona um obstaculo no jogo, na posicao (LARGURA_DA_TELA, 562) com velocidade 5
            game.obstaculos.append(Obstaculo(LARGURA_DA_TELA, 562, pygame.image.load(os.path.join('Imagens',
'obstaculo_1_1.png')), 5))
```


STEM2D
Go girls!
Muito obrigada!