

Mudanças Estruturais no Código

1. Criação da função “gerar_canos”, evitando repetições no código e melhorando o entendimento ao ler o arquivo e também reduzindo o número de linhas.

```
def gerar_canos():  
    for i in range(2):  
        pipes = random_pipes(WIDTH * i + 700)  
        pipe_group.add(pipes[0])  
        pipe_group.add(pipes[1])
```

2. O código que detectava quando uma sprite deixava os limites da tela, em grande parte responsável pela geração de canos infinita, também se repetia, a função “off_screen_aux” foi criada para evitar essa repetição.

```
def off_screen_aux():  
    pipe_group.remove(pipe_group.sprites()[0])  
    pipe_group.remove(pipe_group.sprites()[0])  
    pipes = random_pipes(WIDTH * 2)  
  
    pipe_group.add(pipes[0])  
    pipe_group.add(pipes[1])
```

3. O código que reiniciava a posição da nave toda vez que o jogo resetava após uma morte e até mesmo em seu início também se repetia, uma nova função foi adicionada ao main, “posicionar”.

```
def posicionar(self):  
    self.rect[0] = 1  
    self.rect[1] = HEIGHT/2
```

4. Criação da função “movernave”, com o próprio nome já diz, ela seta os botões para que a nave se mova pela tela. Essa função foi adicionada para facilitar a implementação do pytest.

```
218 #funcao para mover a nave
219 def movernave():
220     keys = pygame.key.get_pressed()
221     if keys[pygame.K_UP] or keys[pygame.K_w]:
222         nave.up()
223     if keys[pygame.K_DOWN] or keys[pygame.K_s]:
224         nave.down()
225     if keys[pygame.K_RIGHT] or keys[pygame.K_d]:
226         nave.right()
227     if keys[pygame.K_LEFT] or keys[pygame.K_a]:
228         nave.left()
```

5. Criação da função “atalho”, ela seta as teclas de atalho para mudar a dificuldade do jogo, dando o score e chamando a função reset, essa função também foi adicionada para facilitar a implementação do pytest.


```
207 #adiciona teclas de atalho para pular de dificuldade
208 def atalho():
209     for event in pygame.event.get():
210         if event.type == KEYDOWN:
211             if event.key == K_x:
212                 resetGame()
213                 score = 1000 #media
214             if event.key == K_z:
215                 resetGame()
216                 score = 2500 #dificil
217
```

6. Função “opening”, ela chama a abertura e fica em loop até pressionar a tecla espaço e pular indo para a tela de menu. Essa função também foi adicionada para a implementação do pytest.

```
230 #funcao para iniciar abertura
231 def opening():
232     opening_group.add(opening)
233     opening_group.draw(screen)
234     opening_group.update()
235     screen.blit(texto_Opening, (100,400))
236     for event in pygame.event.get():
237         if event.type == QUIT:
238             exit = False
239         if event.type == KEYDOWN:
240             if event.key == K_SPACE:
241                 currentScreen = "Menu"
```

Suítes de Testes

1. Testes de Caixa-Preta

Função nave.rect[0]		
Classes:		
	rect	Resultado
	x = 1	True
	Qualquer outro valor	False
Dados:		
	rect	Resultado
	x = 1	True
	X = 2	False
	 x = 10	False
	x = 0	False

Teste da função nave.rect[0], que informa qual a posição da tela. Quando o jogo começar a nave deve estar em nave.rect[0] = 1, ao fazer o teste deve retornar True para x = 1 e false para qualquer outro valor.

Função random_pipes(xpos)		
Classes:		
	size	Resultado
	size < 150	False
	size > 150	True
	size > 550	False
Dados:		
	size	Resultado
	size < 0	False
	size < 100	False
	size > 150 and size < 550	True
	size > 550	False
	size > 1000	False

Teste da função random_pipes, que gera os canos do jogo de forma aleatória usando o random, a variável size entre 150 e 500, de tal forma que os canos do jogo tem que ser gerados nesse intervalo, ou seja, para qualquer outro intervalo de valores, retornar False no teste.

Função ChangeMenu()		
Classes:		
	mx	Resultado
	164 <= mx and mx <= 325	"Start"
	185 <= mx and mx <= 310	"Tutorial"
	17 <= mx and mx <= 100	"Return Menu"
Dados:		
	mx	Resultado
	164 <= mx and mx <= 325	"Start"
	185 <= mx and mx <= 310	"Tutorial"
	17 <= mx and mx <= 100	"Return Menu"

Teste da função ChangeMenu(), em que pega o valor mx e my, posições pressionadas pelo mouse, dependendo dos valores de mx e my a função deve retornar o que fazer no menu, ir para o jogo ("Stark"), ir pro tutorial ou retornar para o menu.

Função resetGame()		
Classes:		
	colidiu	Resultado
	colidiu = False	True
	colidiu = True	False
Dados:		
	rect	Resultado
	colidiu = False	True
	colidiu = True	False
		

Teste da função resetGame(), em que a função muda o valor da variável global "colidiu" para False, para fazer o jogo continuar rodando dentro do while, pois essa variável é quem faz o jogo continuar rodando. Ao testar, o colidiu = false deve retornar True.

Função TelaMorte()		
Classes:		
	mx	Resultado
	125 <= mx and mx <= 230	"Start"
	275 <= mx and mx <= 365	"Exit"
Dados:		
	mx	Resultado
	125 <= mx and mx <= 230	"Start"
	275 <= mx and mx <= 365	"Exit"

Teste da função TelaMorte(), em que pega o valor do mx e my, posições na tela pressionados pelo botão do mouse e dependendo desses valores de mx e my a função deve retornar se o usuário deseja reiniciar o jogo ou sair. O teste deve retornar "Start" ou "Exit" para os valores de mx e my.

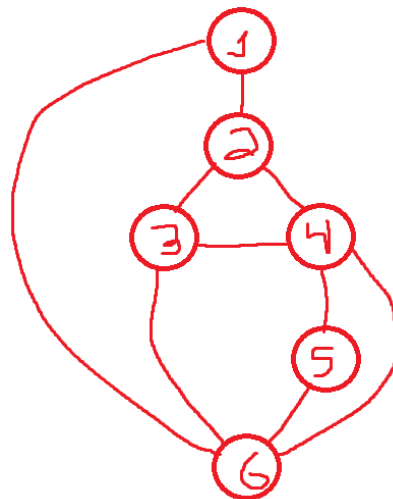
2. Caixa Branca~

Teste na função Oppening.

```

231 #funcao para iniciar abertura
232 def opening():
233     global exit, currentScreen
234     opening_group.add(opening)
235     opening_group.draw(screen)
236     opening_group.update()
237     screen.blit(texto_Opening, (100,400))
238     for event in pygame.event.get():
239         if event.type == QUIT:
240             exit = False
241         if event.type == KEYDOWN:
242             if event.key == K_SPACE:
243                 currentScreen = "Menu"

```



1->{1,6}

2->{1,2,3,6}

3->{1,2,4,6}

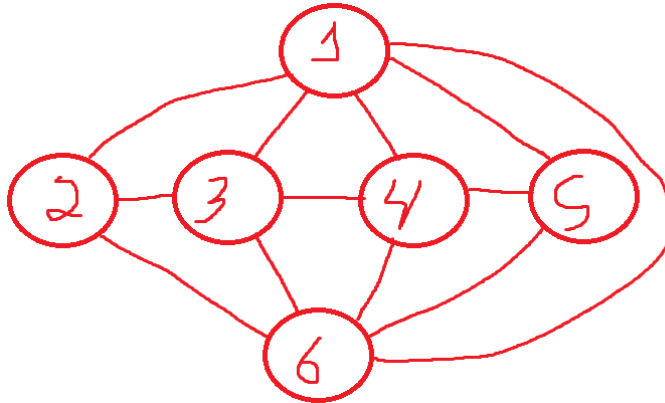
4->{1,2,4,5,6}

5->{1,2,3,4,5,6}

Teste na função movernave

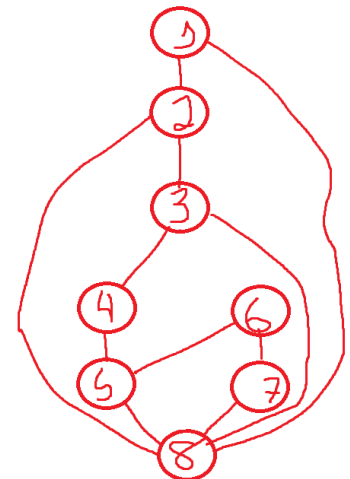
```
#funcao para mover a nave
def movernave():
1 | keys = pygame.key.get_pressed()
2 | if keys[pygame.K_UP] or keys[pygame.K_w]:
   |     nave.up()
3 | if keys[pygame.K_DOWN] or keys[pygame.K_s]:
   |     nave.down()
4 | if keys[pygame.K_RIGHT] or keys[pygame.K_d]:
   |     nave.right()
5 | if keys[pygame.K_LEFT] or keys[pygame.K_a]:
   |     nave.left()
```

1->{1,6}
 2->{1,2,6}
 3->{1,3,6}
 4->{1,4,6}
 5->{1,5,6}
 6->{1,2,3,6}
 7->{1,3,4,6}
 9->{1,4,5,6}
 10->{1,2,3,4,6}
 11->{1,3,4,5,6}
 12->{1,2,3,4,5,6}



Teste na função atalho

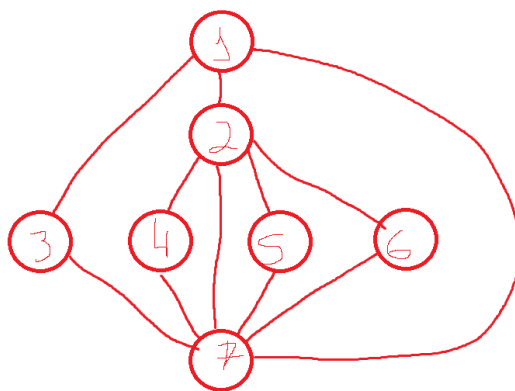
```
#adiciona teclas de atalho para pular de dificuldade
def atalho():
1 | global score
2 | for event in pygame.event.get():
   |     3 | if event.type == KEYDOWN:
       |         4 | if event.key == K_x:
           |             5 | resetGame()
           |             5 | score = 1000 #media
           |         6 | if event.key == K_z:
               |             7 | resetGame()
               |             7 | score = 2500 #dificil
```



1->{1,8}
 2->{1,2,8}
 3->{1,2,3,8}
 4->{1,2,3,4,5,8}
 5->{1,2,3,4,5,6,7,8}

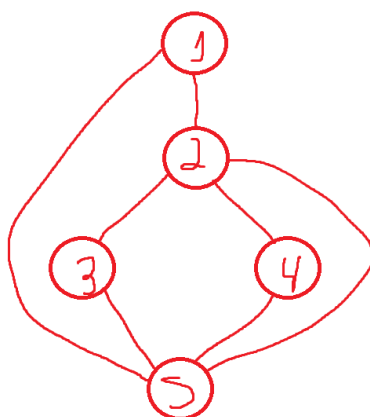
Teste na função ChangeMenu

```
#funcao menu, botoes: start, minitutorial, exit e para retornar menu
def ChangeMenu():
    1 mx, my = pygame.mouse.get_pos()
    2 if pygame.mouse.get_pressed() == (1, 0, 0):
        3 if 164 <= mx and mx <= 325 and 435 <= my and my <= 500:
            return "Start"
        4 elif 185 <= mx and mx <= 310 and 525 <= my and my <= 585:
            return "Tutorial"
        5 elif 185 <= mx and mx <= 310 and 605 <= my and my <= 665:
            return "Exit"
        6 elif 17 <= mx and mx <= 100 and 610 <= my and my <= 680:
            return "ReturnMenu"
```



- 1->{1,7}
- 2->{1,2,7}
- 3->{1,2,3,7}
- 4->{1,2,4,7}
- 5->{1,2,5,7}
- 6->{1,2,6,7}

```
#menu pos morte, com restart e exit
def telaMorte():
    1 mx, my = pygame.mouse.get_pos()
    2 if pygame.mouse.get_pressed() == (1, 0, 0):
        3 if 125 <= mx and mx <= 230 and 410 <= my and my <= 500:
            return "Start"
        4 elif 275 <= mx and mx <= 365 and 420 <= my and my <= 500:
            return "Exit"
```



- 1->{1,5}
- 2->{1,2,5}
- 3->{1,2,3,5}
- 4->{1,2,4,5}