

Branches

Overview

Active

Stale

ΑII

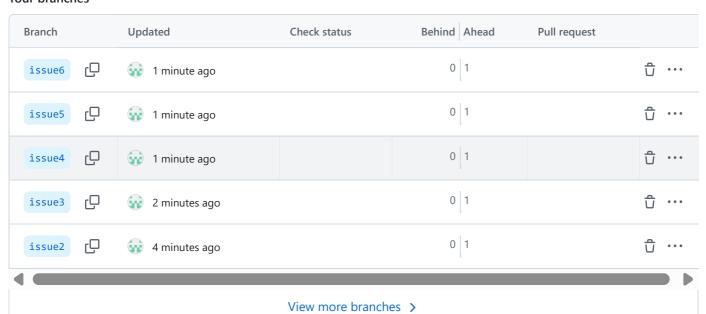
New branch

Q Search	n branches			

Default

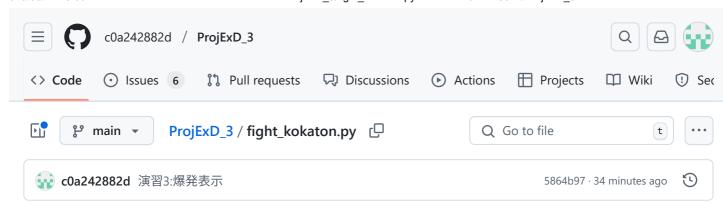
Branch	Updated	Check status	Behind Ahead	Pull request	
main C	31 minutes ago		Default		បំ …

Your branches



Active branches

Branch	Updated	Check status	Behind Ahead	Pull request		
issue6 📮	3 1 minute ago		0 1		បំ …	
issue5	1 minute ago		0 1		ů ···	
issue4	1 minute ago		0 1		ů ···	
issue3	2 minutes ago		0 1		ů ···	
issue2	4 minutes ago		0 1		ů ···	
4					— •	
View more branches >						



263 lines (232 loc) · 9.11 KB

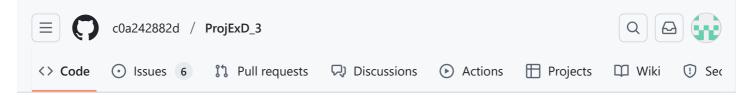
```
Raw 📮 🕹 🗷
                                                                                                <>
Code
        Blame
    1
         import os
    2
         import random
    3
         import sys
         import time
         import pygame as pg
    6
    7
    8
         WIDTH = 1100 # ゲームウィンドウの幅
    9
         HEIGHT = 650 # ゲームウィンドウの高さ
   10
         NUM_OF_BOMBS = 5 # 爆弾の数
   11
         os.chdir(os.path.dirname(os.path.abspath(__file__)))
   12
   13
   14
         def check_bound(obj_rct: pg.Rect) -> tuple[bool, bool]:
             オブジェクトが画面内or画面外を判定し、真理値タプルを返す関数
   16
             引数:こうかとんや爆弾、ビームなどのRect
   17
             戻り値:横方向,縦方向のはみ出し判定結果(画面内:True/画面外:False)
   18
   19
   20
             yoko, tate = True, True
   21
             if obj_rct.left < 0 or WIDTH < obj_rct.right:</pre>
                 yoko = False
   22
             if obj_rct.top < 0 or HEIGHT < obj_rct.bottom:</pre>
   23
   24
                 tate = False
             return yoko, tate
   25
   26
   27
         class Bird:
   28
   29
             ゲームキャラクター(こうかとん)に関するクラス
   30
   31
             delta = { # 押下キーと移動量の辞書
   32
                 pg.K_UP: (0, -5),
   33
                 pg.K_DOWN: (0, +5),
   34
   35
                 pg.K_LEFT: (-5, 0),
   36
                 pg.K_RIGHT: (+5, 0),
   37
             img0 = pg.transform.rotozoom(pg.image.load("fig/3.png"), 0, 0.9)
   38
             img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん (右向き)
   39
   40
             imgs = { # 0度から反時計回りに定義
   41
                 (+5, 0): img, #右
                 (+5, -5): pg.transform.rotozoom(img, 45, 0.9), # 右上
```

```
(0, -5): pg.transform.rotozoom(img, 90, 0.9), #上
43
44
              (-5, -5): pg.transform.rotozoom(img0, -45, 0.9), # 左上
45
              (-5, 0): img0, #左
              (-5, +5): pg.transform.rotozoom(img0, 45, 0.9), # 左下
46
47
              (0, +5): pg.transform.rotozoom(img, -90, 0.9), #下
              (+5, +5): pg.transform.rotozoom(img, -45, 0.9), # 右下
48
          }
49
50
51
          def __init__(self, xy: tuple[int, int]):
52
              こうかとん画像Surfaceを生成する
53
              引数 xy:こうかとん画像の初期位置座標タプル
54
55
56
              self.img = __class__.imgs[(+5, 0)]
57
              self.rct: pg.Rect = self.img.get_rect()
              self.rct.center = xy
58
59
60
          def change_img(self, num: int, screen: pg.Surface):
61
62
              こうかとん画像を切り替え、画面に転送する
              引数1 num:こうかとん画像ファイル名の番号
63
              引数2 screen:画面Surface
64
65
              self.img = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
66
67
              screen.blit(self.img, self.rct)
68
          def update(self, key_lst: list[bool], screen: pg.Surface):
69
70
              押下キーに応じてこうかとんを移動させる
71
72
              引数1 key lst:押下キーの真理値リスト
              引数2 screen:画面Surface
73
74
75
              sum_mv = [0, 0]
              for k, mv in __class__.delta.items():
76
                  if key_lst[k]:
77
                     sum_mv[0] += mv[0]
78
79
                      sum mv[1] += mv[1]
20
              self.rct.move_ip(sum_mv)
              if check_bound(self.rct) != (True, True):
81
82
                  self.rct.move_ip(-sum_mv[0], -sum_mv[1])
              if not (sum mv[0] == 0 and sum mv[1] == 0):
83
                  self.img = __class__.imgs[tuple(sum_mv)]
84
85
              screen.blit(self.img, self.rct)
86
87
88
     class Beam:
89
90
           こうかとんが放つビームに関するクラス
91
92
          def __init__(self, bird:"Bird"):
93
              ビーム画像Surfaceを生成する
94
              引数 bird:ビームを放つこうかとん(Birdインスタンス)
95
96
97
              self.img = pg.image.load("fig/beam.png")
98
              self.rct = self.img.get rect()
99
              self.rct.centery = bird.rct.centery
              self.rct.left = bird.rct.right # ビームの左座標=こうかとんの右座標
```

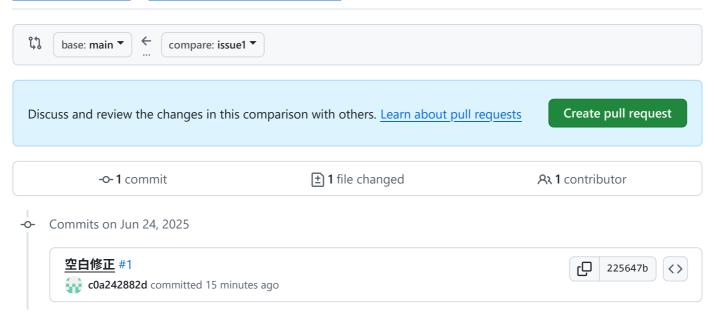
```
101
               self.vx, self.vy = +5, 0
102
103 🗸
           def update(self, screen: pg.Surface):
104
105
               ビームを速度ベクトルself.vx, self.vyに基づき移動させる
               引数 screen:画面Surface
106
107
108
               if check_bound(self.rct) == (True, True):
                   self.rct.move_ip(self.vx, self.vy)
109
110
                   screen.blit(self.img, self.rct)
111
112
113 ∨ class Bomb:
114
115
           爆弾に関するクラス
116
117 🗸
           def __init__(self, color: tuple[int, int, int], rad: int):
118
               引数に基づき爆弾円Surfaceを生成する
119
120
               引数1 color:爆弾円の色タプル
               引数2 rad: 爆弾円の半径
121
122
               self.img = pg.Surface((2*rad, 2*rad))
123
124
               pg.draw.circle(self.img, color, (rad, rad), rad)
125
               self.img.set colorkey((0, 0, 0))
126
               self.rct = self.img.get_rect()
127
               self.rct.center = random.randint(0, WIDTH), random.randint(0, HEIGHT)
               self.vx, self.vy = +5, +5
128
129
130 🗸
           def update(self, screen: pg.Surface):
131
               爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
132
               引数 screen:画面Surface
133
134
135
               yoko, tate = check_bound(self.rct)
               if not yoko:
136
                   self.vx *= -1
137
               if not tate:
138
                   self.vy *= -1
139
140
               self.rct.move_ip(self.vx, self.vy)
               screen.blit(self.img, self.rct)
141
142
143 ∨ class Score:
          def __init__(self):
144 🗸
145
               self.fonto = pg.font.SysFont("hgp創英角ポップ体", 30)
               self.color = (0, 0, 255)
146
147
               self.value = 0
148
               self.img = self.fonto.render(f"スコア: {self.value}", 0, self.color)
               self.rect = self.img.get_rect()
149
150
               self.rect.center = 100, HEIGHT-50
           def update(self, screen: pg.Surface):
151
               self.img = self.fonto.render(f"スコア: {self.value}", 0, self.color)
153
               screen.blit(self.img, self.rect)
154
155 ∨ class Explosion:
156
157
            爆発エフェクトに関するクラス
158
```

```
159 🗸
           def __init__(self, center: tuple[int, int]):
160
               爆発画像Surfaceを生成する
161
               引数 center:爆発の中心座標
162
163
               ex_img = pg.image.load("fig/explosion.gif") #爆発gifを読み込む
164
               self.imgs = [
165
                   ex_img,
166
167
                   pg.transform.flip(ex_img, True, False),
168
                   pg.transform.flip(ex_img, False, True),
169
               ]
               self.life = 30 # 爆発の表示時間
170
               self.rct = self.imgs[0].get_rect()
171
172
               self.rct.center = center
173
174 🗸
           def update(self, screen: pg.Surface):
175
               爆発画像を交互に反転表示して爆発演出
176
               引数 screen:画面Surface
177
178
               self.life -= 1 #爆発経過時間を1減算
179
               img = self.imgs[self.life % len(self.imgs)]
180
               screen.blit(img, self.rct)
181
182
183
184
185 ∨ def main():
           pg.display.set_caption("たたかえ!こうかとん")
186
            screen = pg.display.set mode((WIDTH, HEIGHT))
187
188
           bg_img = pg.image.load("fig/pg_bg.jpg")
           bird = Bird((300, 200))
189
190
           bomb = Bomb((255, 0, 0), 10)
191
           score = Score()
           bombs=[] # 爆弾用の空のリスト
192
193
           beams=[] # ビーム用のリスト
           explosions = [] # Explosionインスタンス用の空のリスト
194
            for _ in range(NUM_OF_BOMBS):
195
196
               bombs.append(Bomb((255,0,0),10))
            # 内包表記
197
198
           beam = None # ゲーム初期化時にはビームは存在しない
           clock = pg.time.Clock()
199
           tmr = 0
200
201
           while True:
               for event in pg.event.get():
202
203
                   if event.type == pg.QUIT:
204
205
                   if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
                       # スペースキー押下でBeamクラスのインスタンス生成
206
                       beams.append(Beam(bird))
207
208
               screen.blit(bg_img, [0, 0])
209
               for n2,bomb2 in enumerate(bombs):
210
211
                   for n,beam2 in enumerate(beams):
                       if bomb2.rct.colliderect(beam2.rct):
212
                           explosions.append(Explosion(bomb.rct.center)) #爆発追加
213
214
                           bombs[n2]=None
215
                           beams[n]=None
216
                           bird.change_img(6, screen)
```

```
pg.display.update()
217
218
                            score.value += 1 # スコアを1点加算
                    beams = [beam for beam in beams if beam is not None]
219
220
                bombs=[bomb for bomb in bombs if bomb is not None]
221
222
                new_explosions = []
223
                for ex in explosions:
                    if ex.life > 0: # 爆発時間が0以上なら表示
224
225
                        ex.update(screen)
226
                        new_explosions.append(ex)
                explosions = new_explosions # 新しい画像に置き換える
227
228
229
                for bomb in bombs:
                    if bird.rct.colliderect(bomb.rct):
230
                        # ゲームオーバー時に,こうかとん画像を切り替え,1秒間表示させる
231
                        bird.change_img(8, screen)
232
233
                        fonto = pg.font.Font(None, 80)
234
                        txt = fonto.render("Game Over", True, (255, 0, 0))
                        screen.blit(txt, [WIDTH//2-150, HEIGHT//2])
235
236
                        score.update(screen) # スコアの描画
237
238
239
                        pg.display.update()
240
                       time.sleep(1)
                        return
242
243
244
245
                key_lst = pg.key.get_pressed()
246
                bird.update(key_lst, screen)
247
                for i,beam in enumerate(beams):
                    if check_bound(beam.rct)==(False,False): # ビームが存在する時
248
                        del beams[i]
249
                    else:
250
251
                        beam.update(screen)
252
                for bomb in bombs:
                    bomb.update(screen)
253
254
                pg.display.update()
                tmr += 1
255
256
                clock.tick(50)
257
258
        if __name__ == "__main__":
259
260
            pg.init()
261
            main()
            pg.quit()
262
263
            sys.exit()
```



Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.



\triangle

This comparison is taking too long to generate.

Unfortunately it looks like we can't render this comparison for you right now. It might be too big, or there might be something weird with your repository.

You can try running this command locally to see the comparison on your machine:



