











 c0b23082de / ProjExD\_Group04



 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights 

ProjExD\_Group04 / ex5.py

...



c0b23082de Update ex5.py

caca999 · last week



410 lines (354 loc) · 13.7 KB

Code Blame Raw Copy Download Edit Dropdown Expand

```
1  import math
2  import os
3  import random
4  import sys
5  import time
6  import pygame as pg
7
8
9  WIDTH = 1100 # ゲームウィンドウの幅
10 HEIGHT = 650 # ゲームウィンドウの高さ
11 os.chdir(os.path.dirname(os.path.abspath(__file__)))
12
13
14  def check_bound(obj_rct: pg.Rect) -> tuple[bool, bool]:
15      """
16      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
17      引数：こうかとんや爆弾、ビームなどのRect
18      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）
19      """
20      yoko, tate = True, True
21      if obj_rct.left < 0 or WIDTH < obj_rct.right:
22          yoko = False
23      if obj_rct.top < 0 or HEIGHT < obj_rct.bottom:
24          tate = False
25      return yoko, tate
26
27
28  def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
29      """
30      orgから見て、dstがどこにあるかを計算し、方向ベクトルをタプルで返す
31      引数1 org：爆弾SurfaceのRect
32      引数2 dst：こうかとんSurfaceのRect
33      戻り値：orgから見たdstの方向ベクトルを表すタプル
34      """
35      x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
36      norm = math.sqrt(x_diff**2+y_diff**2)
37      try:
38          return x_diff/norm, y_diff/norm
39      except ZeroDivisionError:
40          return 0, 0
41
42  def draw_hp_bar(surface,x,y,k_hp:int,k_max_hp:int):
43      """
44      こうかとんのHPバーを表示
```

```

45     引数1 surface: screen
46     引数2 x: HPバーのx座標
47     引数3 y: HPバーのy座標
48     引数4 k_hp: こうかとの今のHP
49     引数5 k_max_hp こうかとの最大HP
50     """
51
52     hp_bar_width = 150
53     hp_bar_height = 20
54     #HPの割合を計算
55     hp_ratio = k_hp / k_max_hp
56
57     #バー外枠を描画
58     pg.draw.rect(surface, (255, 0, 0), (x, y, hp_bar_width, hp_bar_height))
59
60     #HPバーの中身を描画
61     fill_width = int(hp_bar_width * hp_ratio)
62     pg.draw.rect(surface, (0, 255, 0), (x, y, fill_width, hp_bar_height))
63
64     class Bird(pg.sprite.Sprite):
65         """
66         ゲームキャラクター（こうかとん）に関するクラス
67         """
68         delta = { # 押下キーと移動量の辞書
69             pg.K_UP: (0, -1),
70             pg.K_DOWN: (0, +1),
71             pg.K_LEFT: (-1, 0),
72             pg.K_RIGHT: (+1, 0),
73         }
74
75         def __init__(self, num: int, xy: tuple[int, int]):
76             """
77             こうかとん画像Surfaceを生成する
78             引数1 num: こうかとん画像ファイル名の番号
79             引数2 xy: こうかとん画像の位置座標タプル
80             """
81             super().__init__()
82             img0 = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
83             img = pg.transform.flip(img0, True, False) # デフォルトのこうかとん
84             self.imgs = {
85                 (+1, 0): img, # 右
86                 (+1, -1): pg.transform.rotozoom(img, 45, 1.0), # 右上
87                 (0, -1): pg.transform.rotozoom(img, 90, 1.0), # 上
88                 (-1, -1): pg.transform.rotozoom(img0, -45, 1.0), # 左上
89                 (-1, 0): img0, # 左
90                 (-1, +1): pg.transform.rotozoom(img0, 45, 1.0), # 左下
91                 (0, +1): pg.transform.rotozoom(img, -90, 1.0), # 下
92                 (+1, +1): pg.transform.rotozoom(img, -45, 1.0), # 右下
93             }
94             self.dire = (+1, 0)
95             self.image = self.imgs[self.dire]
96             self.rect = self.image.get_rect()
97             self.rect.center = xy
98             self.speed = 10
99
100         def change_img(self, num: int, screen: pg.Surface):
101             """
102             こうかとん画像を切り替え、画面に転送する
103             引数1 num: こうかとん画像ファイル名の番号
104             引数2 screen: 画面Surface

```

```

105         """
106         self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 2.0)
107         screen.blit(self.image, self.rect)
108
109     def update(self, key_lst: list[bool], screen: pg.Surface):
110         """
111         押下キーに応じてこうかとんを移動させる
112         引数1 key_lst : 押下キーの真理値リスト
113         引数2 screen : 画面Surface
114         """
115         sum_mv = [0, 0]
116         for k, mv in __class__.delta.items():
117             if key_lst[k]:
118                 sum_mv[0] += mv[0]
119                 sum_mv[1] += mv[1]
120         self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
121         if check_bound(self.rect) != (True, True):
122             self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
123         if not (sum_mv[0] == 0 and sum_mv[1] == 0):
124             self.dire = tuple(sum_mv)
125             self.image = self.imgs[self.dire]
126         if key_lst[pg.K_LSHIFT]:
127             self.speed = 20
128         else:
129             self.speed = 10
130         screen.blit(self.image, self.rect)
131
132
133     class Bomb(pg.sprite.Sprite):
134         """
135         爆弾に関するクラス
136         """
137         colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
138
139     def __init__(self, emy: "Enemy", bird: Bird):
140         """
141         爆弾円Surfaceを生成する
142         引数1 emy : 爆弾を投下する敵機
143         引数2 bird : 攻撃対象のこうかとん
144         """
145         super().__init__()
146         rad = random.randint(10, 50) # 爆弾円の半径 : 10以上50以下の乱数
147         self.image = pg.Surface((2*rad, 2*rad))
148         color = random.choice(__class__.colors) # 爆弾円の色 : クラス変数からランダム選択
149         pg.draw.circle(self.image, color, (rad, rad), rad)
150         self.image.set_colorkey((0, 0, 0))
151         self.rect = self.image.get_rect()
152         # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
153         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
154         self.rect.centerx = emy.rect.centerx
155         self.rect.centery = emy.rect.centery+emy.rect.height//2
156         self.speed = 6
157
158     def update(self):
159         """
160         爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
161         引数 screen : 画面Surface
162         """
163         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
164         if check_bound(self.rect) != (True, True):

```

```
165         self.kill()
166
167
168     class Beam(pg.sprite.Sprite):
169         """
170         ビームに関するクラス
171         """
172     def __init__(self, bird: Bird, angle0=0):
173         """
174         ビーム画像Surfaceを生成する
175         引数 bird: ビームを放つこうかとん
176         """
177         super().__init__()
178         self.vx, self.vy = bird.dire
179         angle = math.degrees(math.atan2(-self.vy, self.vx)) + angle0
180         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 2.0)
181         self.vx = math.cos(math.radians(angle))
182         self.vy = -math.sin(math.radians(angle))
183         self.rect = self.image.get_rect()
184         self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
185         self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
186         self.speed = 10
187
188
189     def update(self):
190         """
191         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
192         引数 screen: 画面Surface
193         """
194         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
195         if check_bound(self.rect) != (True, True):
196             self.kill()
197
198
199     class NeoBeam:
200         """
201         ビームを複数打てるかもしれないクラス
202         """
203     def __init__(self, bird: Bird, num: int):
204         self.bird = bird
205         self.num = num
206     def gen_beams(self):
207         """
208         引数 num: ビームの本数
209         """
210         return [Beam(self.bird, angle) for angle in range(-50, 51, int(100/(self.num-1)))]
211
212     class Explosion(pg.sprite.Sprite):
213         """
214         爆発に関するクラス
215         """
216     def __init__(self, obj: "Bomb|Enemy", life: int):
217         """
218         爆弾が爆発するエフェクトを生成する
219         引数1 obj: 爆発するBombまたは敵機インスタンス
220         引数2 life: 爆発時間
221         """
222         super().__init__()
223         img = pg.image.load(f"fig/explosion.gif")
224         self.imgs = [img, pg.transform.flip(img, 1, 1)]
```

```
225         self.image = self.imgs[0]
226         self.rect = self.image.get_rect(center=obj.rect.center)
227         self.life = life
228
229     def update(self):
230         """
231         爆発時間を1減算した爆発経過時間_lifeに応じて爆発画像を切り替えることで
232         爆発エフェクトを表現する
233         """
234         self.life -= 1
235         self.image = self.imgs[self.life//10%2]
236         if self.life < 0:
237             self.kill()
238
239     class Gravity(pg.sprite.Sprite):
240     def __init__(self, life: int):
241         super().__init__()
242
243         self.image = pg.Surface((WIDTH, HEIGHT))
244         self.life = life
245         pg.draw.rect(self.image, (0, 0, 0), pg.Rect(0, 0, WIDTH, HEIGHT))
246         self.rect = self.image.get_rect()
247         self.image.set_alpha(180)
248
249     def update(self):
250         self.life -= 1
251         if self.life < 0:
252             self.kill()
253
254     # class newbeam(pg.sprite.Sprite):
255     #     def __init__(self, bird: Bird, num: int):
256     #         self.bird = bird
257     #         self.num = num
258     #     def gen_beams(self):
259     #         """
260     #         引数 num: ビームの本数
261     #         """
262     #         return [Beam(self.bird, angle) for angle in range(-50, 51, int(100/(self.num-1)))]
263
264     class Enemy(pg.sprite.Sprite):
265         """
266         敵機に関するクラス
267         """
268         imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
269
270
271     def __init__(self, bird: Bird):
272         smplace = [(random.randint(0, WIDTH), 0),
273                    (random.randint(0, WIDTH), HEIGHT),
274                    (0, random.randint(0, HEIGHT)),
275                    (WIDTH, random.randint(0, HEIGHT))]
276         super().__init__()
277         self.image = random.choice(__class__.imgs)
278         self.rect = self.image.get_rect()
279         self.rect.center = smplace[random.randint(0, 3)]
280         self.vx, self.vy = calc_orientation(self.rect, bird.rect)
281         self.speed = 2
282         self.rect.centerx = self.rect.centerx
283         self.rect.centery = self.rect.centery + self.rect.height // 2
284
```

```
285  ✓ def update(self, bird: Bird):
286      """
287      敵機を速度ベクトルself.vyに基づき移動（降下）させる
288      ランダムに決めた停止位置_boundまで降下したら、_stateを停止状態に変更する
289      引数 screen: 画面Surface
290      """
291      self.vx, self.vy = calc_orientation(self.rect, bird.rect)
292      self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
293
294
295  ✓ class Score:
296      """
297      打ち落とした爆弾、敵機の数スコアとして表示するクラス
298      爆弾: 1点
299      敵機: 10点
300      """
301  ✓ def __init__(self):
302      self.font = pg.font.Font(None, 50)
303      self.color = (0, 0, 255)
304      self.value = 0
305      self.image = self.font.render(f"Score: {self.value}", 0, self.color)
306      self.rect = self.image.get_rect()
307      self.rect.center = 100, HEIGHT-50
308
309      def update(self, screen: pg.Surface):
310          self.image = self.font.render(f"Score: {self.value}", 0, self.color)
311          screen.blit(self.image, self.rect)
312
313
314  ✓ def main():
315      pg.display.set_caption("真! こうかとん無双")
316      screen = pg.display.set_mode((WIDTH, HEIGHT))
317      bg_img = pg.image.load(f"fig/pg_bg.jpg")
318      score = Score()
319
320      bird = Bird(3, (600, 500))
321      bombs = pg.sprite.Group()
322      beams = pg.sprite.Group()
323      exps = pg.sprite.Group()
324      emys = pg.sprite.Group()
325      grav = pg.sprite.Group()
326      k_max_hp = 5
327      k_hp = k_max_hp
328      tmr = 0
329      clock = pg.time.Clock()
330      flag = 1.0
331      framer = 20
332
333      while True:
334          key_lst = pg.key.get_pressed()
335          for event in pg.event.get():
336              if event.type == pg.QUIT:
337                  return 0
338              if key_lst[pg.K_RETURN] and score.value >= 200:
339                  grav.add(Gravity(400))
340                  score.value -= 200
341
342          screen.blit(bg_img, [0, 0])
343
344
```

```
345     a = score.value
346     if a/100 == flag: # スコアが100の倍数ごとにframerを値を減る
347         flag+=1      # 値が減ることに来る敵の数が増えていく
348         framer -= 1
349     if framer <= 0:
350         framer = 1
351     print(framer)
352     if tmr%framer == 0: # 200フレームに1回, 敵機を出現させる
353         emys.add(Enemy(bird))
354
355     if tmr%30 == 0: # 300フレームに1回, 敵機を出現させる
356         beams.add(Beam(bird))
357
358
359     # for emy in emys:
360     #     if emy.state == "stop" and tmr%emy.interval == 0:
361     #         # 敵機が停止状態に入ったら, intervalに応じて爆弾投下
362     #         bombs.add(Bomb(emy, bird))
363
364     for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
365         exps.add(Explosion(emy, 100)) # 爆発エフェクト
366         score.value += 10 # 10点アップ
367         bird.change_img(6, screen) # こうかたん喜びエフェクト
368
369     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
370         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
371
372
373     for bomb in pg.sprite.groupcollide(bombs, grav, True, False).keys():
374         exps.add(Explosion(bomb, 50))
375
376     for emy in pg.sprite.groupcollide(emys, grav, True, False).keys():
377         exps.add(Explosion(emy, 100))
378
379     draw_hp_bar(screen, bird.rect.centerx - 75, bird.rect.centery - 65, k_hp, k_max_hp)
380     if len(pg.sprite.spritecollide(bird, emys, True)) != 0:
381         k_hp -= 1
382     if k_hp < 0:
383         bird.change_img(8, screen) # こうかたん悲しみエフェクト
384         score.update(screen)
385         pg.display.update()
386         time.sleep(2)
387         return
388
389     grav.update()
390     grav.draw(screen)
391     bird.update(key_lst, screen)
392     beams.update()
393     beams.draw(screen)
394     emys.update(bird)
395     emys.draw(screen)
396     bombs.update()
397     bombs.draw(screen)
398     exps.update()
399     exps.draw(screen)
400     score.update(screen)
401     pg.display.update()
402     tmr += 1
403     clock.tick(50)
404
```

```
405
406     if __name__ == "__main__":
407         pg.init()
408         main()
409         pg.quit()
410         sys.exit()
```