















 KOBAYASHIYUKIDA / ProjExD_05

 |     

 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights 

 main **ProjExD_05 / kill_kokaton.py** 

 Go to file t ...

 KOBAYASHIYUKIDA 初期状態

last week



259 lines (212 loc) · 7.98 KB

CodeBlame

RawCopyDownloadEditDropdown

```
1  import math
2  import random
3  import sys
4  import time
5
6
7  import pygame as pg
8
9
10 WIDTH = 1200 # ゲームウィンドウの幅
11 HEIGHT = 600 # ゲームウィンドウの高さ
12
13
14 def check_bound(obj: pg.Rect) -> tuple[bool, bool]:
15     """
16     オブジェクトが画面内か画面外かを判定し、真理値タプルを返す
17     引数 obj: オブジェクト (爆弾, こうかとん, ビーム) Surface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.left < 0 or WIDTH < obj.right: # 横方向のはみ出し判定
22         yoko = False
23     if obj.top < 0 or HEIGHT < obj.bottom: # 縦方向のはみ出し判定
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     return x_diff/norm, y_diff/norm
38
39
40 class Bird(pg.sprite.Sprite):
41     """
42     ゲームキャラクター (こうかとん) に関するクラス
43     """
44     delta = { # 押下キーと移動量の辞書
```

```
45     pg.K_UP: (0, -1),
46     pg.K_DOWN: (0, +1),
47 }
48
49 ✓ def __init__(self, xy: tuple[int, int]):
50     """
51     こうかтон画像Surfaceを生成する
52     引数1 xy: こうかтон画像の位置座標タプル
53     """
54     super().__init__()
55     img0 = pg.transform.rotozoom(pg.image.load(f"ex05/fig/cat.png"), 0, 0.1)
56     self.image = pg.transform.flip(img0, True, False) # デフォルトのこうかтон
57     self.dire = (+1, 0)
58     self.rect = self.image.get_rect()
59     self.rect.left = 0
60     self.speed = 10
61
62
63 ✓ def update(self, key_lst: list[bool], screen: pg.Surface):
64     """
65     押下キーに応じてこうかтонを移動させる
66     引数1 key_lst: 押下キーの真理値リスト
67     引数2 screen: 画面Surface
68     """
69
70     sum_mv = [0, 0]
71     for k, mv in __class__.delta.items():
72         if key_lst[k]:
73             self.rect.move_ip(+self.speed*mv[0], +self.speed*mv[1])
74             sum_mv[0] += mv[0]
75             sum_mv[1] += mv[1]
76     if check_bound(self.rect) != (True, True):
77         for k, mv in __class__.delta.items():
78             if key_lst[k]:
79                 self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
80     if not (sum_mv[0] == 0 and sum_mv[1] == 0):
81         self.dire = tuple(sum_mv)
82
83     screen.blit(self.image, self.rect)
84
85
86     def get_direction(self) -> tuple[int, int]:
87         return self.dire
88
89
90 ✓ class Beam(pg.sprite.Sprite):
91     """
92     ビームに関するクラス
93     """
94 ✓ def __init__(self, bird: Bird):
95     """
96     引数に基づきビームSurfaceを生成する
97     引数 bird: ビームを放つこうかтон
98     """
99     super().__init__()
100     self.image = pg.transform.rotozoom(pg.image.load(f"ex05/fig/beam.png"), 0, 2.0)
101     self.rect = self.image.get_rect()
102     self.rect.left = bird.rect.right
103     self.rect.centery = bird.rect.centery
104     self.vx, self.vy = +5, 0
```

```

105         self.speed = 5
106
107
108     def update(self):
109         """
110         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
111         引数 screen : 画面Surface
112         """
113         self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
114         if check_bound(self.rect) != (True, True):
115             self.kill()
116
117
118     class Explosion(pg.sprite.Sprite):
119         """
120         爆発に関するクラス
121         """
122     def __init__(self, obj: "Bomb|Enemy", life: int):
123         """
124         爆弾が爆発するエフェクトを生成する
125         引数1 obj : 爆発するBombまたは敵機インスタンス
126         引数2 life : 爆発時間
127         """
128         super().__init__()
129         img = pg.image.load("ex05/fig/explosion.gif")
130         self.imgs = [img, pg.transform.flip(img, 1, 1)]
131         self.image = self.imgs[0]
132         self.rect = self.image.get_rect(center=obj.rect.center)
133         self.life = life
134
135     def update(self):
136         """
137         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
138         爆発エフェクトを表現する
139         """
140         self.life -= 1
141         self.image = self.imgs[self.life//10%2]
142         if self.life < 0:
143             self.kill()
144
145
146     class Enemy(pg.sprite.Sprite):
147         """
148         敵機に関するクラス
149         """
150         imgs = [pg.image.load(f"ex05/fig/alien{i}.png") for i in range(1, 4)]
151
152     def __init__(self):
153         super().__init__()
154         self.image = random.choice(__class__.imgs)
155         self.rect = self.image.get_rect()
156         self.rect.right = WIDTH
157         self.vy = +6
158         self.bound = random.randint(0, HEIGHT) # 停止位置
159         self.state = "down" # 降下状態or停止状態
160         self.interval = random.randint(50, 300) # 爆弾投下インターバル
161
162     def update(self):
163         """
164         敵機を速度ベクトルself.vyに基づき移動（降下）させる
165         """

```

```
165     フンタムに決めた停止位置_boundまで降したら、_stateを停止状態に変更する
166     引数 screen : 画面Surface
167     """
168     if self.rect.centery > self.bound:
169         self.vy = 0
170         self.state = "stop"
171     self.rect.centery += self.vy
172
173
174 ✓ class Score:
175     """
176     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
177     爆弾 : 1点
178     敵機 : 10点
179     """
180 ✓ def __init__(self):
181     self.font = pg.font.Font(None, 50)
182     self.color = (0, 0, 255)
183     self.score = 0
184     self.image = self.font.render(f"Score: {self.score}", 0, self.color)
185     self.rect = self.image.get_rect()
186     self.rect.center = 100, HEIGHT-50
187
188     def score_up(self, add): #スコアを加算
189         self.score += add
190
191
192     def update(self, screen: pg.Surface):
193         self.image = self.font.render(f"Score: {self.score}", 0, self.color)
194         screen.blit(self.image, self.rect)
195
196
197
198 ✓ def main():
199     pg.display.set_caption("倒せ！こうかとん！")
200     screen = pg.display.set_mode((WIDTH, HEIGHT))
201     bg_img = pg.image.load("ex05/fig/pg_bg.jpg")
202     score = Score()
203
204     bird = Bird((900, 400))
205     bombs = pg.sprite.Group()
206     beams = pg.sprite.Group()
207     exps = pg.sprite.Group()
208     emys = pg.sprite.Group()
209
210     tmr = 0
211     clock = pg.time.Clock()
212     while True:
213         key_lst = pg.key.get_pressed()
214         for event in pg.event.get():
215             if event.type == pg.QUIT:
216                 return 0
217
218             elif event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
219                 beams.add(Beam(bird))
220
221         screen.blit(bg_img, [0, 0])
222
223         if tmr%200 == 0: # 200フレームに1回、敵機を出現させる
224             emys.add(Enemy())
225
226
```

```
226     for emy in pg.sprite.groupcollide(emy, beams, True, True).keys():
227         exps.add(Explosion(emy, 100)) # 爆発エフェクト
228         score.score_up(10) # 10点アップ
229
230     for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
231         exps.add(Explosion(bomb, 50)) # 爆発エフェクト
232         score.score_up(1) # 1点アップ
233
234     if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
235         score.update(screen)
236         pg.display.update()
237         time.sleep(2)
238         return
239
240     bird.update(key_lst, screen)
241     beams.update()
242     beams.draw(screen)
243     emys.update()
244     emys.draw(screen)
245     bombs.update()
246     bombs.draw(screen)
247     exps.update()
248     exps.draw(screen)
249     score.update(screen)
250     pg.display.update()
251     tmr += 1
252     clock.tick(50)
253
254
255 if __name__ == "__main__":
256     pg.init()
257     main()
258     pg.quit()
259     sys.exit()
```