











 c0b230680f /
ProjExD_Group01



 Code  Issues  Pull requests  Actions  Projects  Wiki  Security  Insights 

ProjExD_Group01 / koukoku.py 

c0b23056c0 こうかとかべがあたったときのしより

55a4a66 · 7 minutes ago



862 lines (754 loc) · 33.5 KB

CodeBlame

RawCopyDownloadEditDropdownFullscreen

```
1  import math
2  import os
3  import sys
4  import time
5  from pygame.locals import *
6  import pygame as pg
7  import time
8
9
10 os.chdir(os.path.dirname(os.path.abspath(__file__)))
11 WIDTH, HEIGHT = 600, 900
12 STAGE_NUM = 0
13
14 #クリアコード
15 # class Clear():
16 #     def __init__(self, screen):
17 #         font = pg.font.Font(None, 80)
18 #         self.img = pg.image.load("fig/宇宙.png")
19 #         self.txt = font.render("CLEAR", True, (255, 255, 0))
20 #         screen.fill((0, 255, 0))
21 #         self.img1 = pg.image.load("fig/2.png")
22 #         self.img2 = pg.image.load("fig/5.png")
23 #         self.img3 = pg.image.load("fig/2.png")
24 #         self.img4 = pg.image.load("fig/5.png")
25 #         self.img5 = pg.image.load("fig/ペンギン.png")
26 #         self.img6 = pg.image.load("fig/アボカド.png")
27
28 #     def update(self, screen):
29 #         screen.blit(self.img, [0, 0])
30 #         screen.blit(self.txt, [200, 400])
31 #         screen.blit(self.img1, [100, 150])
32 #         screen.blit(self.img2, [450, 150])
33 #         screen.blit(self.img3, [100, 750])
34 #         screen.blit(self.img4, [450, 750])
35 #         screen.blit(self.img5, [450, 400])
36 #         screen.blit(self.img6, [75, 400])
37
38
39  ✓ def check_bound(obj_rct:pg.Rect) -> tuple[bool, bool]:
40      """
41      Rectの画面内外判定用の関数
42      引数： こうかとんRect, または, 爆弾Rect, またはビームRect
43      戻り値： 横方向判定結果, 縦方向判定結果 (True : 画面内 / False : 画面外)
44      """
```

```
45     yoko, tate = True, True
46     if obj_rct.right < 0 or WIDTH < obj_rct.left: # 横方向のはみ出し判定
47         yoko = False
48     if obj_rct.bottom < 0 or HEIGHT < obj_rct.top: # 縦方向のはみ出し判定
49         tate = False
50     return yoko, tate
51
52
53     def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
54         """
55         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
56         引数1 org: 爆弾SurfaceのRect
57         引数2 dst: こうかとんSurfaceのRect
58         戻り値: orgから見たdstの方向ベクトルを表すタプル
59         """
60         x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
61         norm = math.sqrt(x_diff**2+y_diff**2)
62         return x_diff/norm
63
64
65     class Yoko_Stage(pg.sprite.Sprite):
66         """
67         ステージの描写に関するクラス
68         """
69     def __init__(self, xy: tuple[int, int]) -> None:
70         """
71         ステージSurfaceを生成する
72         """
73         super().__init__()
74         self.image = pg.Surface((100, 25)) # 横長の枠組みのSurface
75         pg.draw.rect(self.image, (180,82,45), [0, 0, 100, 25])
76         pg.draw.rect(self.image, (255, 255, 255), [0, 0, 100, 25], 2) # 横向き of 四角形の枠組みの描写
77         self.rect = self.image.get_rect()
78         self.rect.topleft = xy
79
80     def update(self):
81         pass
82
83
84     class Tate_Stage(pg.sprite.Sprite):
85         """
86         ステージの描写に関するクラス
87         """
88     def __init__(self, xy: tuple[int, int]) -> None:
89         """
90         ステージSurfaceを生成する
91         """
92         super().__init__()
93         self.image = pg.Surface((25, 100)) # 縦長の枠組みのSurface
94         pg.draw.rect(self.image, (180,82,45), [0, 0, 25, 100])
95         pg.draw.rect(self.image, (255, 255, 255), [0, 0, 25, 100], 2) # 縦向き of 四角形の枠組みの描写
96         self.rect = self.image.get_rect()
97         self.rect.topleft = xy
98
99     def update(self):
100         pass
101
102
103     class Kao():
104         """
105         おこげは用のカニフ
```

```

105     こうかとんがクリアしたり、マグマなどに衝突したら呼び出されるクラス
106     """
107
108     def __init__(self):
109         """
110         顔のSurfaceを生成する
111         """
112         self.image = self.image_kao = pg.transform.rotozoom(pg.image.load(f"fig/kao.png"), 0, 0.15) # 顔
113         self.rect = self.image.get_rect()
114
115     def update(self, ko: "Kokaton", screen: pg.Surface):
116         """
117         こうかとんの位置に基づいて顔を表示する
118         引数 1 ko : こうかとんのインスタンスの取得
119         引数 2 screen : 画面Surface
120         """
121         self.rect.center = ko.rect.center
122         screen.blit(self.image, self.rect)
123
124
125     class Kokaton(pg.sprite.Sprite):
126         """
127         主人公こうかとんに関する関数
128         """
129     def __init__(self, x, y):
130         """
131         こうかとん画像Surfaceを生成する
132         """
133         super().__init__()
134         self.image = pg.transform.flip(pg.image.load(f"fig/0.png"), True, False) # こうかとんの画像を読み込
135         self.rect = self.image.get_rect()
136         self.rect.centerx = x
137         self.rect.bottom = y - 1 # こうかとんがステージに埋まらないようにしてる
138         self.vx, self.vy = 0, 0
139         self.speed = 5
140
141     def update(self, screen: pg.Surface):
142         """
143         こうかとんをクリックしたらこうかとんを動かす
144         引数 screen : 画面Surface
145         """
146         self.rect.move_ip(self.speed * self.vx, self.speed * self.vy)
147         screen.blit(self.image, self.rect)
148
149
150     class Yoko_Pin(pg.sprite.Sprite): # 横用のピンのクラス
151         """
152         横長用のピンのクラス
153         リストにあらかじめ図形左上の座標タプルを入れておくことで呼び出しを簡単にしている
154         """
155         yoko_topleft_xy = [(50, 165), (50, 300), (50, 575), (300, 250), (300, 450), (300, 775)] # ピンの座標
156
157     def __init__(self, ytl):
158         """
159         横長用のピンのSurface
160         引数 1 : 座標タプルの引数
161         """
162         super().__init__()
163         self.image = pg.Surface((250, 20)) # ピンの横用のSurface
164         pg.draw.rect(self.image, (255, 215, 0), [0, 0, 250, 20])
165         pg.draw.rect(self.image, (0, 0, 0), [0, 0, 250, 20], 2) # ピンの枠線

```

```

166     self.rect = self.image.get_rect()
167     # ステージごとに配置を変えられるようにする
168     self.rect.topleft = __class__.yoko_topleft_xy[ytl]
169     self.vx, self.vy = 0, 0
170
171     def update(self):
172         """
173         マウスがピンの半分より壁側にあるとき左クリックするとピンを動かす
174         画面外に出たらGroupオブジェクトから消去される
175         """
176         # マウスの処理
177         if pg.mouse.get_pressed()[0]:
178             mouse_x, mouse_y = pg.mouse.get_pos() # マウスの x 座標と y 座標の取得
179             if self.rect.centerx > (WIDTH / 2): # ピンが画面の右側にあったら
180                 if (self.rect.centerx <= mouse_x <= self.rect.right and self.rect.top <= mouse_y <= self.rect.bottom):
181                     self.vx = +2
182             elif self.rect.centerx < (WIDTH / 2): # ピンが画面の左側にあったら
183                 if (self.rect.left <= mouse_x <= self.rect.centerx and self.rect.top <= mouse_y <= self.rect.bottom):
184                     self.vx = -2
185             self.rect.move_ip(self.vx, self.vy)
186
187             if check_bound(self.rect) != (True, True):
188                 self.kill() # 画面外にでたらGroupオブジェクトから消去される
189
190
191     class Tate_Pin(pg.sprite.Sprite): # 縦用のピンのクラス
192         """
193         縦長用のピンのクラス
194         リストにあらかじめ図形左上の座標タプルを入れておくことで呼び出しを簡単にしている
195         """
196         tate_topleft_xy = [(290, 50), (290, 600)] # 座標タプルのリスト
197
198     def __init__(self, ttl):
199         """
200         縦長用のピンのSurface
201         引数 1 : 座標タプルの引数
202         """
203         super().__init__()
204         self.image = pg.Surface((20, 250)) # ピンの縦用のSurface
205         pg.draw.rect(self.image, (255, 215, 0), [0, 0, 20, 250])
206         pg.draw.rect(self.image, (0, 0, 0), [0, 0, 20, 250], 2)
207         self.rect = self.image.get_rect()
208         # ステージごとに変えることのできる配置
209         self.rect.topleft = __class__.tate_topleft_xy[ttl]
210         self.vx, self.vy = 0, 0
211
212     def update(self):
213         """
214         マウスがピンの半分より壁側にあるとき左クリックするとピンを動かす
215         画面外に出たらGroupオブジェクトから消去される
216         """
217         if pg.mouse.get_pressed()[0]: # マウスの処理
218             mouse_x, mouse_y = pg.mouse.get_pos()
219             if self.rect.centery > (HEIGHT / 2):
220                 if (self.rect.left <= mouse_x <= self.rect.right and self.rect.centery <= mouse_y <= self.rect.bottom):
221                     self.vy = +2
222             elif self.rect.centery < (HEIGHT / 2):
223                 if (self.rect.left <= mouse_x <= self.rect.right and self.rect.top <= mouse_y <= self.rect.bottom):
224                     self.vy = -2
225

```

```
226         self.rect.move_ip(self.vx, self.vy)
227         if check_bound(self.rect) != (True, True):
228             self.kill() # 画面外にでたらGroupオブジェクトから消去される
229
230
231     class Obj(pg.sprite.Sprite):
232         """
233         マグマなどの描写に関するクラス
234         """
235         colors = [(232, 57, 41), (255, 239, 108), (188, 226, 232)]
236     def __init__(self, num, xy: tuple[int, int]):
237         """
238         マグマなどの円Surface
239         引数2 : 色タプルリストを指定するための引数
240         引数3 : 物体の位置タプル
241         """
242         super().__init__()
243         rad = 10
244         self.image = pg.Surface((2*rad, 2*rad))
245         self.color = __class__.colors[num]
246         pg.draw.circle(self.image, self.color, (rad, rad), rad)
247         self.image.set_colorkey((0, 0, 0))
248         self.rect = self.image.get_rect()
249         self.rect.center = xy
250         self.vx, self.vy = 0, 0
251         self.speed = 3
252
253     def update(self):
254         self.rect.move_ip(self.vx * self.speed, self.vy * self.speed)
255
256
257     class Stone(pg.sprite.Sprite):
258         """
259         マグマと水が当たった時の黒曜石を表示させるクラス
260         """
261     def __init__(self, obj: "Obj"):
262         """
263         石の表示のSurface
264         引数1 : マグマなどのインスタンス
265         """
266         super().__init__()
267         self.image = pg.Surface((20, 20))
268         pg.draw.rect(self.image, (0, 0, 0), [0, 0, 20, 20])
269         self.rect = self.image.get_rect()
270         self.rect.center = obj.rect.center
271
272     def update(self):
273         pass
274
275
276     class Start:
277     def __init__(self, stage_num):
278         self.fonto = pg.font.Font(None, 120)
279         self.image = pg.Surface((250, 80))
280         self.txt = self.fonto.render("Start", True, (0,0,0)) #start文字
281         pg.draw.rect(self.image, (255, 255, 255), [0, 0, 250, 80], 2) # startの周りの四角
282         self.image.set_colorkey((0, 0, 0))
283         self.rect = self.image.get_rect()
284         self.txt_rect = self.txt.get_rect()
285         self.s_font = pg.font.Font(None, 100)
```

```
286         self.font_image = self.s_font.render(f"{stage_num + 1}", 0, (0, 0, 0))
287         self.font_rect = self.font_image.get_rect()
288         self.font_rect.center = WIDTH / 2, HEIGHT / 4
289         self.image3 = pg.image.load("fig/mura1.png")
290         self.image1 = pg.image.load("fig/2.png") #こうかとん
291         self.image2 = pg.image.load("fig/9.png") #こうかとん
292         self.rect1 = self.image1.get_rect()
293         self.rect2 = self.image2.get_rect()
294         self.rect3 = self.image3.get_rect()
295         self.txt_rect.center = WIDTH / 2, HEIGHT / 2
296         self.rect.center = self.txt_rect.center
297
298     def update(self, screen: pg.Surface):
299         screen.blit(self.image3,[0,0])
300         screen.blit(self.txt,self.txt_rect)
301         screen.blit(self.image1,[100, 380])
302         screen.blit(self.image2,[450, 380])
303         screen.blit(self.image, self.rect)
304         screen.blit(self.font_image, self.font_rect)
305
306
307     class Gameover:
308     def __init__(self):
309         #ゲームオーバーの文字列
310         self.image = pg.image.load("fig/game.png") # ゲームオーバー画像
311         self.image_rect = self.image.get_rect()
312         self.image_rect.center = 300, 200
313         #
314         # self.go = pg.font.Font(None, 100)
315         # self.go_image = self.go.render(f"GAME OVER", 0, (255, 0, 0), (255, 255, 255))
316         # self.go_rect = self.go_image.get_rect()
317         # self.go_rect.center = 300, 200
318         #コンチヌー
319         self.con = pg.font.Font(None, 50)
320         self.con_image = self.con.render(f"CONTINUE ???", 1, (255, 255, 255), (255, 0, 0))
321         self.con_rect = self.con_image.get_rect()
322         self.con_rect.center = 300, 500
323         #リタイア
324         self.ret = pg.font.Font(None, 50)
325         self.ret_image = self.ret.render(f"RETIRE ???", 1, (255, 255, 255), (0, 0, 255))
326         self.ret_rect = self.ret_image.get_rect()
327         self.ret_rect.center = 300, 650
328
329     def update(self, screen: pg.Surface):
330         # screen.blit(self.go_image, self.go_rect)
331         screen.blit(self.con_image, self.con_rect)
332         screen.blit(self.ret_image, self.ret_rect)
333         screen.blit(self.image, self.image_rect)
334
335
336     class Clear:
337     def __init__(self):
338         #クリアの文字列
339         font = pg.font.Font(None, 80)
340         self.img = pg.image.load("fig/宇宙.png")
341         self.txt = font.render("CLEAR", True, (255, 255, 0))
342         self.txt_rect = self.txt.get_rect()
343         self.img1 = pg.image.load("fig/2.png")
344         self.img2 = pg.image.load("fig/5.png")
345         self.img3 = pg.image.load("fig/2.png")
```

```
346         self.img4 = pg.image.load("fig/5.png")
347         self.img5 = pg.image.load("fig/ペンギン.png")
348         self.img6 = pg.image.load("fig/アボカド.png")
349         self.txt_rect.center = 300, 200
350         # self.cl = pg.font.Font(None, 100)
351         # self.cl_image = self.cl.render(f"CLEAR", 0, (255, 0, 0), (255, 255, 255))
352         # self.cl_rect = self.cl_image.get_rect()
353         # self.cl_rect.center = 300, 200
354         #コンチヌー
355         self.next = pg.font.Font(None, 50)
356         self.next_image = self.next.render(f"NEXT STAGE", 1, (255, 255, 255), (255, 0, 0))
357         self.next_rect = self.next_image.get_rect()
358         self.next_rect.center = 300, 500
359         #リタイア
360         self.con = pg.font.Font(None, 50)
361         self.con_image = self.con.render(f"CONTINUE ??", 1, (255, 255, 255), (0, 0, 255))
362         self.con_rect = self.con_image.get_rect()
363         self.con_rect.center = 300, 650
364
365     def update(self, screen: pg.Surface):
366         #
367         screen.blit(self.img, [0, 0])
368         # screen.blit(self.txt, [200, 400])
369         screen.blit(self.img1, [100, 150])
370         screen.blit(self.img2, [450, 150])
371         screen.blit(self.img3, [100, 750])
372         screen.blit(self.img4, [450, 750])
373         screen.blit(self.img5, [450, 400])
374         screen.blit(self.img6, [75, 400])
375
376         screen.blit(self.txt, self.txt_rect)
377         screen.blit(self.next_image, self.next_rect)
378         screen.blit(self.con_image, self.con_rect)
379
380
381     def main(stage_num):
382         game_stats = None
383         pg.display.set_caption("広告ゲーム")
384         screen = pg.display.set_mode((WIDTH, HEIGHT))
385
386         bg_img = pg.image.load(f"fig/bg_img{stage_num}.jpg")
387
388         start = Start(stage_num)
389         go = Gameover()
390         cl = Clear()
391
392         kao = Kao()
393         ysts = pg.sprite.Group()
394         tsts = pg.sprite.Group()
395         mgms = pg.sprite.Group()
396         wtrs = pg.sprite.Group()
397         trs = pg.sprite.Group()
398         sixtones = pg.sprite.Group()
399         ypins = pg.sprite.Group()
400         tpins = pg.sprite.Group()
401
402         clock = pg.time.Clock()
403
404         # 基本ステージの描写(周りの壁)
405         for x in range(7):
```

```

406     ysts.add(Yoko_Stage((100 * x, 0)))
407     ysts.add(Yoko_Stage((100 * x - 50, 25)))
408     ysts.add(Yoko_Stage((100 * x, 875)))
409     ysts.add(Yoko_Stage((100 * x - 50, 850)))
410     for y in range(10):
411         tsts.add(Tate_Stage((0, 100 * y)))
412         tsts.add(Tate_Stage((25, 100 * y - 50)))
413         tsts.add(Tate_Stage((575, 100 * y)))
414         tsts.add(Tate_Stage((550, 100 * y - 50)))
415
416     #ここはステージ毎に数値を変更することができる
417     kokaton = Kokaton(100, 775)
418     # 追加ステージの座標をfor文で指定する
419     ysts.add(Yoko_Stage((200, 800)))
420     for y in range(16):
421         for x in range(3):
422             if y % 2 == 0:
423                 ysts.add(Yoko_Stage((100 * x, 50 + 25 * y)))
424             if y % 2 == 1:
425                 ysts.add(Yoko_Stage((200, 50 + 25 * y)))
426                 ysts.add(Yoko_Stage((100 * x - 50, 50 + 25 * y)))
427             ysts.add(Yoko_Stage((100 * x, 775)))
428             ysts.add(Yoko_Stage((100 * x - 50, 800)))
429             ysts.add(Yoko_Stage((100 * x, 825)))
430
431     #ステージによって変えられる
432     for x in range(11):
433         mgms.add(Obj(0, (325 + 20 * x, 400)))
434         wtrs.add(Obj(2, (325 + 20 * x, 820)))
435         trs.add(Obj(1, (325 + 20 * x, 200)))
436
437     # 使いたい棒の座標タプルが入ったリストを指定する
438     ypins.add(Yoko_Pin(3))
439     ypins.add(Yoko_Pin(4))
440     ypins.add(Yoko_Pin(5))
441     #tpins.add(Tate_Pin(0))
442
443     while True:
444         start.update(screen)
445         pg.display.update()
446         for event in pg.event.get():
447             if event.type == pg.QUIT:
448                 return "retire"
449             # 一ステージとばすチート
450             if event.type == pg.KEYDOWN and event.key == pg.K_g:
451                 return "clear"
452
453             if pg.mouse.get_pressed()[0]: # マウスの処理
454                 mouse_x, mouse_y = pg.mouse.get_pos()
455                 if (start.rect.topleft[0] <= mouse_x <= start.rect.bottomright[0]) and (start.rect.topleft[1]
456                     <= mouse_y <= start.rect.bottomright[1]):
457                     break
458
459     while True:
460         for event in pg.event.get():
461             if event.type == pg.QUIT:
462                 return "retire"
463             if event.type == pg.KEYDOWN and event.key == pg.K_r:
464                 return "continue"
465
466         screen.blit(bg_img, [0, 0])

```



```
466
467     kokaton.vy = +1
468     # 何にも衝突がなければobj.vyは+2になる
469     for mgm in mgms:
470         mgm.vy = +2
471     for wtr in wtrs:
472         wtr.vy = +2
473     for tre in trs:
474         tre.vy = +2
475
476     if pg.mouse.get_pressed()[0]: # マウスの処理
477         mouse_x, mouse_y = pg.mouse.get_pos()
478         # マウスがこうかとんに重なっていたら
479         if (kokaton.rect.topleft[0] <= mouse_x <= kokaton.rect.bottomright[0]) and (kokaton.rect.top
480             kokaton.vx = calc_orientation(kokaton.rect, tre.rect)
481
482     # こうかとんとステージの当たり判定
483     if len(pg.sprite.spritecollide(kokaton, ysts, False)) != 0:
484         kokaton.vy = 0
485     for tst in pg.sprite.spritecollide(kokaton, tsts, False):
486         if tst.rect.left <= kokaton.rect.right and tst.rect.right >= kokaton.rect.left:
487             kokaton.vx = 0
488             kokaton.rect.right -= 10
489         elif tst.rect.right >= kokaton.rect.left and tst.rect.left <= kokaton.rect.right:
490             kokaton.vy = 0
491             kokaton.rect.centerx += 10
492
493     # こうかとんとピンの当たり判定
494     if len(pg.sprite.spritecollide(kokaton, tpins, False)) != 0:
495         kokaton.vx = 0
496     if len(pg.sprite.spritecollide(kokaton, ypins, False)) != 0:
497         kokaton.vy = 0
498
499     # こうかとんとマグマと水で出来た石の当たり判定
500     if len(pg.sprite.spritecollide(kokaton, sixtones, False)) != 0:
501         kokaton.vy = 0
502
503     # ピンとobjの当たり判定
504     for mgm in pg.sprite.groupcollide(mgms, ypins, False, False).keys():
505         mgm.vy = 0
506     for wtr in pg.sprite.groupcollide(wtrs, ypins, False, False).keys():
507         wtr.vy = 0
508     for tre in pg.sprite.groupcollide(trs, ypins, False, False).keys():
509         tre.vy = 0
510
511     # 床とobjの当たり判定
512     for mgm in pg.sprite.groupcollide(mgms, ysts, False, False).keys():
513         mgm.vy = 0
514     for wtr in pg.sprite.groupcollide(wtrs, ysts, False, False).keys():
515         wtr.vy = 0
516     for tre in pg.sprite.groupcollide(trs, ysts, False, False).keys():
517         tre.vy = 0
518
519     # 水とマグマの当たり判定 ... ステージごとに下においてある方を基準とする
520     for wtr in pg.sprite.groupcollide(wtrs, mgms, True, True).keys():
521         sixtones.add(Stone(wtr))
522
523     # 宝とマグマと水で出来た石の当たり判定
524     for tre in pg.sprite.groupcollide(trs, sixtones, False, False).keys():
525         tre.vy = 0
526
```

```
527 # マグマと宝物の当たり判定
528 if len(pg.sprite.grouppcollide(mgms, trs, False, True).values()) != 0:
529     game_stats = "gameover"
530     kokaton.update(screen)
531     kao.update(kokaton, screen)
532     sixtones.draw(screen)
533     ysts.draw(screen)
534     tsts.draw(screen)
535     pg.display.update()
536     time.sleep(2)
537
538 # こうかんとマグマの当たり判定, ゲームオーバー
539 if len(pg.sprite.spritecollide(kokaton, mgms, False)) != 0:
540     game_stats = "gameover"
541     kokaton.update(screen)
542     kao.update(kokaton, screen)
543     sixtones.draw(screen)
544     ysts.draw(screen)
545     tsts.draw(screen)
546     pg.display.update()
547     time.sleep(2)
548
549 # こうかんと宝の当たり判定, クリアー!!
550 if len(pg.sprite.spritecollide(kokaton, trs, True)) != 0:
551     if len(trs) == 0:
552         game_stats = "clear"
553         sixtones.draw(screen)
554         kokaton.update(screen)
555         kao.update(kokaton, screen)
556         ysts.draw(screen)
557         tsts.draw(screen)
558         pg.display.update()
559         time.sleep(2)
560
561 if game_stats != None:
562     break
563
564 kokaton.update(screen)
565 ypins.update()
566 ypins.draw(screen)
567 tpins.update()
568 tpins.draw(screen)
569 mgms.update()
570 mgms.draw(screen)
571 wtrs.update()
572 wtrs.draw(screen)
573 trs.update()
574 trs.draw(screen)
575 sixtones.update()
576 sixtones.draw(screen)
577 ysts.draw(screen)
578 tsts.draw(screen)
579 pg.display.update()
580 clock.tick(60)
581
582 while True:
583     if game_stats == "gameover":
584         go.update(screen)
585         pg.display.update()
586         for event in pg.event.get():
```

```
587         if event.type == pg.QUIT:
588             return "retire"
589     if pg.mouse.get_pressed()[0]: # マウスの処理
590         mouse_x, mouse_y = pg.mouse.get_pos()
591         if (go.con_rect.topleft[0] <= mouse_x <= go.con_rect.bottomright[0]) and (go.con_rect.to
592             return "continue"
593         elif (go.ret_rect.topleft[0] <= mouse_x <= go.ret_rect.bottomright[0]) and (go.ret_rect.
594             return "retire"
595
596     elif game_stats == "clear":
597         cl.update(screen)
598         pg.display.update()
599         for event in pg.event.get():
600             if event.type == pg.QUIT:
601                 return "retire"
602             if pg.mouse.get_pressed()[0]: # マウスの処理
603                 mouse_x, mouse_y = pg.mouse.get_pos()
604                 if (cl.next_rect.topleft[0] <= mouse_x <= cl.next_rect.bottomright[0]) and (cl.next_rect
605                     return "clear"
606                 elif (cl.con_rect.topleft[0] <= mouse_x <= cl.con_rect.bottomright[0]) and (cl.con_rect.
607                     return "continue"
608
609
610     def main2(stage_num):
611         game_stats = None
612         pg.display.set_caption("広告ゲーム2")
613         screen = pg.display.set_mode((WIDTH, HEIGHT))
614         bg_img = pg.image.load(f"fig/bg_img{stage_num}.jpg")
615
616         start = Start(stage_num)
617         go = Gameover()
618         cl = Clear()
619
620         kao = Kao()
621         ysts = pg.sprite.Group()
622         tsts = pg.sprite.Group()
623         mgms = pg.sprite.Group()
624         wtrs = pg.sprite.Group()
625         trs = pg.sprite.Group()
626         sixtones = pg.sprite.Group()
627         ypins = pg.sprite.Group()
628         tpins = pg.sprite.Group()
629
630         clock = pg.time.Clock()
631
632         # 基本ステージの描写(周りの壁)
633         for x in range(7):
634             ysts.add(Yoko_Stage((100 * x, 0)))
635             ysts.add(Yoko_Stage((100 * x - 50, 25)))
636             ysts.add(Yoko_Stage((100 * x, 875)))
637             ysts.add(Yoko_Stage((100 * x - 50, 850)))
638         for y in range(10):
639             tsts.add(Tate_Stage((0, 100 * y)))
640             tsts.add(Tate_Stage((25, 100 * y - 50)))
641             tsts.add(Tate_Stage((575, 100 * y)))
642             tsts.add(Tate_Stage((550, 100 * y - 50)))
643
644         #ここはステージ毎に数値を変更することができる
645         kokaton = Kokaton(175, 550)
646         # 追加ステージの座標をfor文で指定する
```

```
647     for i in range(6):
648         tsts.add(Tate_Stage((290, 100 * i)))
649     for y in range(22):
650         for x in range(3):
651             if y % 2 == 0:
652                 ysts.add(Yoko_Stage((300 + 100 * x, 50 + 25 * y)))
653             if y % 2 == 1:
654                 ysts.add(Yoko_Stage((300, 50 + 25 * y)))
655                 ysts.add(Yoko_Stage((500 - 100 * x + 50, 50 + 25 * y))) # マイナスから回すことでステージの
656
657     #ステージによって変えられる
658     for x in range(11):
659         mgms.add(Obj(0, (75 + 20 * x, 800)))
660         wtrs.add(Obj(2, (75 + 20 * x, 250)))
661         trs.add(Obj(1, (325 + 20 * x, 800)))
662
663     # 使いたい棒の座標タプルが入ったリストを指定する
664     ypins.add(Yoko_Pin(1))
665     ypins.add(Yoko_Pin(2))
666     tpins.add(Tate_Pin(1))
667
668     while True:
669         start.update(screen)
670         pg.display.update()
671         for event in pg.event.get():
672             if event.type == pg.QUIT:
673                 return "retire"
674             # 一ステージとばすチート
675             if event.type == pg.KEYDOWN and event.key == pg.K_g:
676                 return "clear"
677             if event.type == pg.KEYDOWN and event.key == pg.K_r:
678                 return "continue"
679
680             if pg.mouse.get_pressed()[0]: # マウスの処理
681                 mouse_x, mouse_y = pg.mouse.get_pos()
682                 if (start.rect.topleft[0] <= mouse_x <= start.rect.bottomright[0]) and (start.rect.topleft[1]
683                     break
684
685     while True:
686         for event in pg.event.get():
687             if event.type == pg.QUIT:
688                 return "retire"
689
690         screen.blit(bg_img, [0, 0])
691
692         kokaton.vy = +1
693         # 何にも衝突がなければobj.vyは+2になる
694         for mgm in mgms:
695             mgm.vy = +2
696         for wtr in wtrs:
697             wtr.vy = +2
698         for tre in trs:
699             tre.vy = +2
700
701         if pg.mouse.get_pressed()[0]: # マウスの処理
702             mouse_x, mouse_y = pg.mouse.get_pos()
703             # マウスがこうかとんに重なっていたら
704             if (kokaton.rect.topleft[0] <= mouse_x <= kokaton.rect.bottomright[0]) and (kokaton.rect.top
705                 kokaton.vx = calc_orientation(kokaton.rect, tre.rect)
706
```

```
707 # こうかんとステージの当たり判定
708 if len(pg.sprite.spritecollide(kokaton, ysts, False)) != 0:
709     kokaton.vy = 0
710 for tst in pg.sprite.spritecollide(kokaton, tsts, False):
711     if tst.rect.left <= kokaton.rect.right and tst.rect.right >= kokaton.rect.left:
712         kokaton.vx = 0
713         kokaton.rect.right -= 10
714     elif tst.rect.right >= kokaton.rect.left and tst.rect.left <= kokaton.rect.right:
715         kokaton.vy = 0
716         kokaton.rect.centerx += 10
717
718 # こうかんとピンの当たり判定
719 if len(pg.sprite.spritecollide(kokaton, tpins, False)) != 0:
720     kokaton.vx = 0
721 if len(pg.sprite.spritecollide(kokaton, ypins, False)) != 0:
722     kokaton.vy = 0
723
724 # こうかんとマグマと水で出来た石の当たり判定
725 if len(pg.sprite.spritecollide(kokaton, sixtones, False)) != 0:
726     kokaton.vy = 0
727
728 # ピンとobjの当たり判定
729 for mgm in pg.sprite.groupcollide(mgms, ypins, False, False).keys():
730     mgm.vy = 0
731 for wtr in pg.sprite.groupcollide(wtrs, ypins, False, False).keys():
732     wtr.vy = 0
733 for tre in pg.sprite.groupcollide(trs, ypins, False, False).keys():
734     tre.vy = 0
735
736 # 床とobjの当たり判定
737 for mgm in pg.sprite.groupcollide(mgms, ysts, False, False).keys():
738     mgm.vy = 0
739 for wtr in pg.sprite.groupcollide(wtrs, ysts, False, False).keys():
740     wtr.vy = 0
741 for tre in pg.sprite.groupcollide(trs, ysts, False, False).keys():
742     tre.vy = 0
743
744 # 水とマグマの当たり判定 ... ステージごとに下においてある方を基準とする
745 for mgm in pg.sprite.groupcollide(mgms, wtrs, True, True).keys():
746     sixtones.add(Stone(mgm))
747
748 # 宝とマグマと水で出来た石の当たり判定
749 for tre in pg.sprite.groupcollide(trs, sixtones, False, False).keys():
750     tre.vy = 0
751
752 # マグマと宝物の当たり判定
753 if len(pg.sprite.groupcollide(mgms, trs, False, True).values()) != 0:
754     game_stats = "gameover"
755     kokaton.update(screen)
756     kao.update(kokaton, screen)
757     sixtones.draw(screen)
758     ysts.draw(screen)
759     tsts.draw(screen)
760     pg.display.update()
761     time.sleep(2)
762
763 # こうかんとマグマの当たり判定, ゲームオーバー
764 if len(pg.sprite.spritecollide(kokaton, mgms, False)) != 0:
765     game_stats = "gameover"
766     kokaton.update(screen)
```

```
767         kao.update(kokaton, screen)
768         sixtones.draw(screen)
769         ysts.draw(screen)
770         tsts.draw(screen)
771         pg.display.update()
772         time.sleep(2)
773
774     # こうかんと宝の当たり判定, クリアー!!
775     if len(pg.sprite.spritecollide(kokaton, trs, True)) != 0:
776         if len(trs) == 0:
777             game_stats = "clear"
778             sixtones.draw(screen)
779             kokaton.update(screen)
780             kao.update(kokaton, screen)
781             ysts.draw(screen)
782             tsts.draw(screen)
783             pg.display.update()
784             time.sleep(2)
785
786         if game_stats != None:
787             break
788
789     kokaton.update(screen)
790     ypins.update()
791     ypins.draw(screen)
792     tpins.update()
793     tpins.draw(screen)
794     mgms.update()
795     mgms.draw(screen)
796     wtrs.update()
797     wtrs.draw(screen)
798     trs.update()
799     trs.draw(screen)
800     sixtones.update()
801     sixtones.draw(screen)
802     ysts.draw(screen)
803     tsts.draw(screen)
804     pg.display.update()
805     clock.tick(60)
806
807     while True:
808         if game_stats == "gameover":
809             go.update(screen)
810             pg.display.update()
811             for event in pg.event.get():
812                 if event.type == pg.QUIT:
813                     return "retire"
814             if pg.mouse.get_pressed()[0]: # マウスの処理
815                 mouse_x, mouse_y = pg.mouse.get_pos()
816                 if (go.con_rect.topleft[0] <= mouse_x <= go.con_rect.bottomright[0]) and (go.con_rect.to
817                     return "continue"
818                 elif (go.ret_rect.topleft[0] <= mouse_x <= go.ret_rect.bottomright[0]) and (go.ret_rect.
819                     return "retire"
820             elif game_stats == "clear":
821                 cl.update(screen)
822                 pg.display.update()
823                 for event in pg.event.get():
824                     if event.type == pg.QUIT:
825                         return "retire"
826                 if pg.mouse.get_pressed()[0]: # マウスの処理
```

```
827         mouse_x, mouse_y = pg.mouse.get_pos()
828         if (cl.next_rect.topleft[0] <= mouse_x <= cl.next_rect.bottomright[0]) and (cl.next_rect
829             return "clear"
830         elif (cl.con_rect.topleft[0] <= mouse_x <= cl.con_rect.bottomright[0]) and (cl.con_rect.
831             return "continue"
832
833
834 if __name__=="__main__":
835     pg.init()
836     # ステージの変化とゲームオーバーなどの処理
837     while True:
838         GAME_STATS = None
839         GAME_STATS = main(STAGE_NUM) # main関数内でreturnしてくる文字列を変数に格納する
840         if GAME_STATS == "clear":
841             STAGE_NUM += 1
842             break
843         elif GAME_STATS == "continue":
844             continue
845         elif GAME_STATS == "retire":
846             pg.quit()
847             sys.exit()
848
849     while True:
850         GAME_STATS = None
851         GAME_STATS = main2(STAGE_NUM)
852         if GAME_STATS == "clear":
853             STAGE_NUM += 1
854             break
855         elif GAME_STATS == "continue":
856             continue
857         elif GAME_STATS == "retire":
858             pg.quit()
859             sys.exit()
860     pg.quit()
861     sys.exit()
862
```