

c0a2303678 /
ProjExD_Group05

<> Code

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

ProjExD_Group05 / suta-_koukaton.py



c0a2303678 item branch 実装(デバッグメッセージ削除版)

52e4451 · now



466 lines (387 loc) · 15.6 KB

Code

Blame

Raw



```
1  #!/usr/bin/env python
2  import os
3  import random
4  from typing import List
5
6  # import basic pygame modules
7  import pygame as pg
8
9  # see if we can load more than standard BMP
10 if not pg.image.get_extended():
11     raise SystemExit("Sorry, extended image module required")
12
13
14 # game constants
15 MAX_SHOTS = 1 # most player bullets onscreen
16 MAX_BOMBS = 1
17 ALIEN_ODDS = 22 # chances a new alien appears
18 BOMB_ODDS = 60 # chances a new bomb will drop
19 ALIEN_RELOAD = 12 # frames between new aliens
20 SCREENRECT = pg.Rect(0, 0, 640, 480)
21 SCORE = 0
22
23 main_dir = os.path.split(os.path.abspath(__file__))[0]
24
25
26 def load_image(file):
27     """loads an image, prepares it for play"""
28     file = os.path.join(main_dir, "data", file)
29     try:
30         surface = pg.image.load(file)
31     except pg.error:
32         raise SystemExit(f'Could not load image "{file}" {pg.get_error()}')
33     return surface.convert()
34
35
36 def load_sound(file):
37     """because pygame can be compiled without mixer."""
38     if not pg.mixer:
39         return None
40     file = os.path.join(main_dir, "data", file)
41     try:
42         sound = pg.mixer.Sound(file)
```

```
43         return sound
44     except pg.error:
45         print(f"Warning, unable to load, {file}")
46     return None
47
48
49 ✓ class Player(pg.sprite.Sprite):
50     """
51     Playerのイニシャライザ
52     動作メソッド、
53     銃の発射位置メソッドを生成しているクラス
54     """
55
56     speed = 5
57     bounce = 24
58     gun_offset = 0
59     images: List[pg.Surface] = []
60
61 ✓ def __init__(self, *groups):
62     pg.sprite.Sprite.__init__(self, *groups)
63     self.image = self.images[0]
64     self.rect = self.image.get_rect(midbottom=SCREENRECT.midbottom)
65     self.reloading = 0
66     self.origtop = self.rect.top
67     self.facing = -1
68
69 ✓ def move(self, direction):
70     if direction:
71         self.facing = direction
72         self.rect.move_ip(direction * self.speed, 0)
73         self.rect = self.rect.clamp(SCREENRECT)
74         if direction < 0:
75             self.image = self.images[0]
76         elif direction > 0:
77             self.image = self.images[1]
78         # self.rect.top = self.origtop - (self.rect.left // self.bounce % 2)
79
80     def gunpos(self):
81         pos = self.facing * self.gun_offset + self.rect.centerx
82         return pos, self.rect.top
83
84
85 ✓ class Alien(pg.sprite.Sprite):
86     """
87     エイリアンのイニシャライザ
88     動作メソッド
89     銃の発射位置メソッド
90     エイリアンの位置更新メソッドを生成しているクラス
91     """
92
93     speed = 5
94     gun_offset = 0
95     images: List[pg.Surface] = []
96
97 ✓ def __init__(self, *groups):
98     pg.sprite.Sprite.__init__(self, *groups)
99     self.image = self.images[0]
100     self.reloading = 0
```

```
101         self.rect = self.image.get_rect(midtop=SCREENRECT.midtop)
102         self.facing = -1
103         self.origbottom = self.rect.bottom
104
105     def move(self, direction):
106         if direction:
107             self.facing = direction
108             self.rect.move_ip(direction * self.speed, 0)
109             self.rect = self.rect.clamp(SCREENRECT)
110             if direction < 0:
111                 self.image = self.images[0]
112             elif direction > 0:
113                 self.image = self.images[1]
114
115     def gunpos(self):
116         pos = self.rect.centerx
117         return pos, self.rect.bottom
118
119     def update(self):
120         #self.rect.move_ip(self.facing, 0)
121         if not SCREENRECT.contains(self.rect):
122             self.facing = -self.facing
123             self.rect = self.rect.clamp(SCREENRECT)
124         # self.frame = self.frame + 1
125         # self.image = self.images[self.frame // self.animcycle % 3]
126
127
128     class Explosion(pg.sprite.Sprite):
129         """
130         オブジェクトが衝突した際に爆発する演出を作成するクラス
131         """
132
133         defaultlife = 12
134         animcycle = 3
135         images: List[pg.Surface] = []
136
137     def __init__(self, actor, *groups):
138         pg.sprite.Sprite.__init__(self, *groups)
139         self.image = self.images[0]
140         self.rect = self.image.get_rect(center=actor.rect.center)
141         self.life = self.defaultlife
142
143     def update(self):
144         """
145         called every time around the game loop.
146
147         Show the explosion surface for 'defaultlife'.
148         Every game tick(update), we decrease the 'life'.
149
150         Also we animate the explosion.
151         """
152         self.life = self.life - 1
153         self.image = self.images[self.life // self.animcycle % 2]
154         if self.life <= 0:
155             self.kill()
156
157
158     class Shot(pg.sprite.Sprite):
```

```
159     """
160     Playerが使う銃を生成するクラス
161     """
162
163     speed = -10
164     images: List[pg.Surface] = []
165
166     def __init__(self, pos, *groups):
167         pg.sprite.Sprite.__init__(self, *groups)
168         self.image = self.images[0]
169         self.rect = self.image.get_rect(midbottom=pos)
170
171     def update(self):
172         """
173         called every time around the game loop.
174
175         Every tick we move the shot upwards.
176         """
177         self.rect.move_ip(0, self.speed)
178         if self.rect.top <= 0:
179             self.kill()
180
181
182     class Bomb(pg.sprite.Sprite):
183         """
184         Alienが落とす爆弾を生成するクラス
185         """
186
187         speed = 10
188         images: List[pg.Surface] = []
189
190         def __init__(self, alien_pos, *groups):
191             pg.sprite.Sprite.__init__(self, *groups)
192             self.image = self.images[0]
193             self.rect = self.image.get_rect(midtop=alien_pos)
194
195         def update(self):
196             """
197             - make an explosion.
198             - remove the Bomb.
199             """
200             self.rect.move_ip(0, self.speed)
201             if self.rect.bottom >= SCREENRECT.bottom:
202                 self.kill()
203
204
205     class Score(pg.sprite.Sprite):
206         """
207         状況に応じて増減し、MAX_GUNSとMAX_BOMBSに関与するスコアクラス
208         """
209
210     def __init__(self, *groups):
211         pg.sprite.Sprite.__init__(self, *groups)
212         self.font = pg.font.Font(None, 20)
213         self.font.set_italic(1)
214         self.color = "white"
215         self.lastscore = -1
216         self.update()
```

```
217         self.rect = self.image.get_rect().move(10, 450)
218
219     def update(self):
220         """We only update the score in update() when it has changed."""
221         if SCORE != self.lastscore:
222             self.lastscore = SCORE
223             msg = f"Score: {SCORE}"
224             self.image = self.font.render(msg, 0, self.color)
225
226
227     class Item(pg.sprite.Sprite):
228         """
229         ゲーム内でアイテムを表現するクラス。
230         speed : int : アイテムの移動速度。
231         images : List[pg.Surface] : アイテムを表現する画像のリスト。
232         rect : pg.Rect : アイテムの位置とサイズを表す矩形。
233         spawned : bool : アイテムが生成されたかどうかを示すフラグ。
234         メソッド:
235         update():アイテムの位置を更新し、画面端との衝突を処理する。
236         spawn():アイテムを画面の中央に生成する。
237         is_spawned() -> bool:アイテムが現在生成されているかどうかを確認する。
238         collide_bombs(bombs: pg.sprite.Group) -> bool:爆弾との衝突を確認し、処理する。
239         collide_shots(shots: pg.sprite.Group) -> bool:ショットとの衝突を確認し、処理する。
240         reset():アイテムを初期状態にリセットする。
241         """
242
243         speed: int = 2 #itemの移動速度
244         images: List[pg.Surface] = []#itemの画像リスト
245
246     def __init__(self, *groups: pg.sprite.AbstractGroup) -> None:
247         """
248         Itemオブジェクトを初期化する。
249         引数: *groups : pg.sprite.AbstractGroup : スプライトが所属するグループ。
250         """
251         pg.sprite.Sprite.__init__(self, *groups)
252         self.image = pg.transform.scale(self.images[0], (64, 48)) # 画像サイズを変更
253         self.image.set_colorkey((255, 255, 255)) # 背景を透明に設定
254         self.rect = self.image.get_rect(center=SCREENRECT.center) # 矩形を取得
255         self.rect.topleft = (-100, -100) # 初期位置を画面外に設定
256         self.spawned = False # アイテムが生成されたかどうかのフラグ
257
258     def update(self) -> None:
259         """
260         アイテムの位置を更新し、画面端との衝突を処理する。
261         """
262         if self.spawned:
263             self.rect.move_ip(self.speed, 0) # アイテムを移動
264             if self.rect.top > SCREENRECT.height:
265                 self.kill() # 画面外に出たらアイテムを消す
266                 self.spawned = False # フラグをリセット
267             if self.rect.right >= SCREENRECT.right or self.rect.left <= 0:
268                 self.speed = -self.speed # 画面端に当たったら移動方向を反転
269
270     def spawn(self) -> None:
271         """
272         アイテムを画面の中央に生成する。
273         """
274         if not self.spawned:
```

```
275         self.rect.center = SCREENRECT.center # アイテムを中央に移動
276         self.spawned = True # フラグを設定
277
278     ✓ def is_spawned(self) -> bool:
279         """
280         アイテムが現在生成されているかどうかを確認する。
281         戻り値: bool : アイテムが生成されていればTrue、そうでなければFalse。
282         """
283         return self.spawned
284
285     ✓ def collide_bombs(self, bombs: pg.sprite.Group) -> bool:
286         """
287         爆弾との衝突を確認し、処理する。
288         引数: bombs : pg.sprite.Group : 衝突を確認する爆弾のグループ。
289         戻り値: bool : アイテムが爆弾と衝突した場合はTrue、そうでない場合はFalse。
290         """
291         if self.spawned:
292             collided = pg.sprite.spritecollide(self, bombs, True) # 衝突を確認
293             if collided:
294                 self.kill() # 衝突したらアイテムを消す
295                 self.spawned = False # フラグをリセット
296                 self.rect.topleft = (-100, -100) # 初期位置にリセット
297                 return True
298         return False
299
300     ✓ def collide_shots(self, shots: pg.sprite.Group) -> bool:
301         """
302         ショットとの衝突を確認し、処理する。
303         引数: shots : pg.sprite.Group : 衝突を確認するショットのグループ。
304         戻り値: bool : アイテムがショットと衝突した場合はTrue、そうでない場合はFalse。
305         """
306         if self.spawned:
307             collided = pg.sprite.spritecollide(self, shots, True)
308             if collided:
309                 self.kill()
310                 self.spawned = False # 衝突したらフラグをリセット
311                 self.rect.topleft = (-100, -100) # 画面外の初期位置にリセット
312                 return True
313         return False
314
315     ✓ def reset(self) -> None:
316         """
317         アイテムを初期状態にリセットする。
318         """
319         self.spawned = False # フラグをリセット
320         self.rect.topleft = (-100, -100) # 画面外に初期位置をリセット
321
322
323     ✓ def main(winstyle=0):
324         # Initialize pygame
325         if pg.get_sdl_version()[0] == 2:
326             pg.mixer.pre_init(44100, 32, 2, 1024)
327         pg.init()
328         if pg.mixer and not pg.mixer.get_init():
329             print("Warning, no sound")
330             pg.mixer = None
331
332         fullscreen = False
```

```
333     winstyle = 0 # |FULLSCREEN
334     bestdepth = pg.display.mode_ok(SCREENRECT.size, winstyle, 32)
335     screen = pg.display.set_mode(SCREENRECT.size, winstyle, bestdepth)
336
337     # Load images, assign to sprite classes
338     img = load_image("3.png")
339     Player.images = [img, pg.transform.flip(img, 1, 0)]
340     img = load_image("explosion1.gif")
341     Explosion.images = [img, pg.transform.flip(img, 1, 1)]
342     Alien.images = [load_image(im) for im in ("alien1.gif", "alien2.gif", "alien3.gif")]
343     Bomb.images = [load_image("bomb.gif")]
344     Shot.images = [load_image("shot.gif")]
345     Item.images = [load_image("item.png")] # アイテム画像を読み込む
346
347     icon = pg.transform.scale(Alien.images[0], (32, 32))
348     pg.display.set_icon(icon)
349     pg.display.set_caption("Pygame Aliens")
350     pg.mouse.set_visible(0)
351
352     bgdtile = load_image("utyuu.jpg")
353     background = pg.Surface(SCREENRECT.size)
354     background.blit(bgdtile, (0, 0))
355     screen.blit(background, (0, 0))
356     pg.display.flip()
357
358     boom_sound = load_sound("boom.wav")
359     shoot_sound = load_sound("car_door.wav")
360     if pg.mixer:
361         music = os.path.join(main_dir, "data", "house_lo.wav")
362         pg.mixer.music.load(music)
363         pg.mixer.music.play(-1)
364
365     player = pg.sprite.Group()
366     aliens = pg.sprite.Group()
367     shots = pg.sprite.Group()
368     bombs = pg.sprite.Group()
369     items = pg.sprite.Group()
370     all = pg.sprite.RenderUpdates()
371
372     global SCORE
373     player = Player(all)
374     alien = Alien(aliens, all)
375     item = Item(items, all) # アイテムを初期化し追加
376
377     if pg.font:
378         all.add(Score(all))
379
380     item_spawn_time = random.randint(300, 600) # 初回のアイテム出現時間をランダムに設定 (5秒から10秒)
381     item_timer = 0
382     item_spawned = False
383
384     clock = pg.time.Clock()
385
386     while player.alive() and alien.alive():
387         for event in pg.event.get():
388             if event.type == pg.QUIT:
389                 return
390             if event.type == pg.KEYDOWN and event.key == pg.K_ESCAPE:
```

```
391         return
392     if event.type == pg.KEYDOWN:
393         if event.key == pg.K_f:
394             if not fullscreen:
395                 print("Changing to FULLSCREEN")
396                 screen_backup = screen.copy()
397                 screen = pg.display.set_mode(SCREENRECT.size, winstyle | pg.FULLSCREEN, bestde
398                 screen.blit(screen_backup, (0, 0))
399             else:
400                 print("Changing to windowed mode")
401                 screen_backup = screen.copy()
402                 screen = pg.display.set_mode(SCREENRECT.size, winstyle, bestdepth)
403                 screen.blit(screen_backup, (0, 0))
404             pg.display.flip()
405             fullscreen = not fullscreen
406
407     keystate = pg.key.get_pressed()
408
409     all.clear(screen, background)
410     all.update()
411
412     direction = keystate[pg.K_RIGHT] - keystate[pg.K_LEFT]
413     player.move(direction)
414     firing = keystate[pg.K_SPACE]
415     if not player.reloading and firing and len(shots) < MAX_SHOTS:
416         Shot(player.gunpos(), shots, all)
417         if pg.mixer and shoot_sound is not None:
418             shoot_sound.play()
419     player.reloading = firing
420
421     direction = keystate[pg.K_d] - keystate[pg.K_a]
422     alien.move(direction)
423     firing = keystate[pg.K_t]
424     if not alien.reloading and firing and len(bombs) < MAX_BOMBS:
425         Bomb(alien.gunpos(), bombs, all)
426         if pg.mixer and shoot_sound is not None:
427             shoot_sound.play()
428     alien.reloading = firing
429
430     for shot in pg.sprite.spritecollide(alien, shots, 1):
431         Explosion(shot, all)
432         Explosion(alien, all)
433         if pg.mixer and boom_sound is not None:
434             boom_sound.play()
435         alien.kill()
436
437     for bomb in pg.sprite.spritecollide(player, bombs, 1):
438         Explosion(bomb, all)
439         Explosion(player, all)
440         if pg.mixer and boom_sound is not None:
441             boom_sound.play()
442         player.kill()
443
444     item_timer += 1# アイテム生成タイマーを更新
445     if not item_spawned and item_timer >= item_spawn_time:
446         item.spawn() # アイテムを生成
447         item_spawned = True
448
```



```
449     # アイテムが爆弾と衝突したかを確認
450     if item.collide_bombs(bombs) or item.collide_shots(shots):
451         print("Item collided") # デバッグメッセージ
452         item_timer = 0
453         item_spawn_time = random.randint(300, 600) # 新しいアイテム出現時間を設定
454         item_spawned = False
455
456     pg.display.update(all.draw(screen))
457
458     clock.tick(40)
459
460     if pg.mixer:
461         pg.mixer.music.fadeout(1000)
462     pg.time.wait(1000)
463
464 if __name__ == "__main__":
465     main()
466     pg.quit()
```