

c0a22070 / ProjExD\_05

Public

<> Code

Issues

Pull requests

Actions

Projects

← Files

main

⌵

⋮

ProjExD\_05 / README.md

c0a22070

4 minutes ago

⋮

🕒

37 lines (35 loc) · 1.24 KB

← Files

main

⌵

ProjExD\_05 / README.md

↑ Top

Preview

Code

Blame

⋮

⋮

# ゲームのタイトル

---

エイリアンの襲来

## 実行環境の必要条件

---

- python >= 3.10
- pygame >= 2.1

## ゲームの概要

---

UFOが上から爆弾を落としてくる。そのUFOを撃ち落とす

## ゲームの実装

---

### 共通基本機能

- 画像を読み込む関数
- 音を読み込む関数
- プレイヤーに関するクラス
- 敵に関するクラス
- 爆発に関するクラス
- 弾丸に関するクラス
- 爆弾に関するクラス
- 爆弾に関するクラス
- スコアに関するクラス
- メインの関数

### 担当追加機能

https://github.com/c0a22070/ProjExD\_05/blob/main/README.md

1/2


- 右shiftを押すと短時間弾の数が増える (松村春輝 c0b22136)
- 戦車の数を2台に増やした (坂田匡弥 c0a22070)
- スコアが10増えるごとに画面の敵をすべて倒す (山本昂 c0a22146)
- ゲームが終了時Gameoverと表示 (東杏澄 c0a21144)
- 各残機を1から3に増加 (福元悠太 c0b20136)

## ToDo

- スタート画面の作成
- スコアのボーナスの選択肢の増加
- それぞれ違う機能の戦車を作成
- 弾の種類を増加
- 残機の数を表示

## メモ

- playr1の移動は右がA左はD弾の発射はW
- playr2の移動は右がJ左はL弾の発射はU
- 右Shiftで一時的に弾を増やす

 [c0a22070](#) / [ProjExD\\_05](#) Public[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#) [main](#) [ProjExD\\_05 / alien.py](#)  Go to file

t

...

[c0a22070](#) 2 hours ago

466 lines (371 loc) · 13.9 KB

Code

Blame



```
1  #!/usr/bin/env python
2  """ pygame.examples.aliens
3
4  Shows a mini game where you have to defend against aliens.
5
6  What does it show you about pygame?
7
8  * pg.sprite, the difference between Sprite and Group.
9  * dirty rectangle optimization for processing for speed.
10 * music with pg.mixer.music, including fadeout
11 * sound effects with pg.Sound
12 * event processing, keyboard handling, QUIT handling.
13 * a main loop frame limited with a game clock from pg.time.Clock
14 * fullscreen switching.
15
16
17 Controls
18 -----
19
20 * Left and right arrows to move.
21 * Space bar to shoot
22 * f key to toggle between fullscreen.
23
24 """
25
26 import random
27 import os
28
29 # import basic pygame modules
30 import pygame as pg
31
32 # see if we can load more than standard BMP
33 if not pg.image.get_extended():
34     raise SystemExit("Sorry, extended image module required")
35
36
37 # game constants
38 MAX_SHOTS = 4 # most player bullets onscreen
39 ALIEN_ODDS = 22 # chances a new alien appears
40 BOMB_ODDS = 60 # chances a new bomb will drop
41 ALIEN_RELOAD = 12 # frames between new aliens
42 SCREENRECT = pg.Rect(0, 0, 640, 480)
43 SCORE = 0
44
```

```
45     main_dir = os.path.split(os.path.abspath(__file__))[0]
46
47
48     def load_image(file):
49         """loads an image, prepares it for play"""
50         file = os.path.join(main_dir, "data", file)
51         try:
52             surface = pg.image.load(file)
53         except pg.error:
54             raise SystemExit('Could not load image "%s" %s' % (file, pg.get_error()))
55         return surface.convert()
56
57
58     def load_sound(file):
59         """because pygame can be compiled without mixer."""
60         if not pg.mixer:
61             return None
62         file = os.path.join(main_dir, "data", file)
63         try:
64             sound = pg.mixer.Sound(file)
65             return sound
66         except pg.error:
67             print("Warning, unable to load, %s" % file)
68             return None
69
70
71     # Each type of game object gets an init and an update function.
72     # The update function is called once per frame, and it is when each object should
73     # change its current position and state.
74     #
75     # The Player object actually gets a "move" function instead of update,
76     # since it is passed extra information about the keyboard.
77
78
79     class Player1(pg.sprite.Sprite):
80         """Representing the player as a moon buggy type car."""
81
82         speed = 10
83         bounce = 24
84         gun_offset = -11
85         images = []
86
87     def __init__(self):
88         pg.sprite.Sprite.__init__(self, self.containers)
89         self.image = self.images[0]
90         self.rect = self.image.get_rect(midbottom=SCREENRECT.midbottom)
91         self.reload = 0
92         self.origtop = self.rect.top
93         self.facing = -1
94
95     def move(self, direction):
96         if direction:
97             self.facing = direction
98             self.rect.move_ip(direction * self.speed, 0)
99             self.rect = self.rect.clamp(SCREENRECT)
100         if direction < 0:
101             self.image = self.images[0]
102         elif direction > 0:
103             self.image = self.images[1]
104         self.rect.top = self.origtop - (self.rect.left // self.bounce % 2)
```

```

105
106     def gunpos(self):
107         pos = self.facing * self.gun_offset + self.rect.centerx
108         return pos, self.rect.top
109
110 ✓ class Player2(pg.sprite.Sprite): # 2 台目の戦車
111     """Representing the player as a moon buggy type car."""
112
113     speed = 10
114     bounce = 24
115     gun_offset = -11
116     images = []
117
118 ✓ def __init__(self):
119     pg.sprite.Sprite.__init__(self, self.containers)
120     self.image = self.images[0]
121     self.rect = self.image.get_rect(midbottom=SCREENRECT.midbottom)
122     self.reloading = 0
123     self.origtop = self.rect.top
124     self.facing = 1
125
126 ✓ def move(self, direction):
127     if direction:
128         self.facing = direction
129     self.rect.move_ip(direction * self.speed, 0)
130     self.rect = self.rect.clamp(SCREENRECT)
131     if direction < 0:
132         self.image = self.images[0]
133     elif direction > 0:
134         self.image = self.images[1]
135     self.rect.top = self.origtop - (self.rect.left // self.bounce % 2)
136
137     def gunpos(self):
138         pos = self.facing * self.gun_offset + self.rect.centerx
139         return pos, self.rect.top
140
141
142 ✓ class Alien(pg.sprite.Sprite):
143     """An alien space ship. That slowly moves down the screen."""
144
145     speed = 13
146     animcycle = 12
147     images = []
148
149 ✓ def __init__(self):
150     pg.sprite.Sprite.__init__(self, self.containers)
151     self.image = self.images[0]
152     self.rect = self.image.get_rect()
153     self.facing = random.choice((-1, 1)) * Alien.speed
154     self.frame = 0
155     if self.facing < 0:
156         self.rect.right = SCREENRECT.right
157
158 ✓ def update(self):
159     self.rect.move_ip(self.facing, 0)
160     if not SCREENRECT.contains(self.rect):
161         self.facing = -self.facing
162         self.rect.top = self.rect.bottom + 1
163         self.rect = self.rect.clamp(SCREENRECT)
164         self.frame = self.frame + 1

```

```
165         self.image = self.images[self.frame // self.animcycle % 3]
166
167
168 ✓ class Explosion(pg.sprite.Sprite):
169     """An explosion. Hopefully the Alien and not the player!"""
170
171     defaultlife = 12
172     animcycle = 3
173     images = []
174
175 ✓ def __init__(self, actor):
176     pg.sprite.Sprite.__init__(self, self.containers)
177     self.image = self.images[0]
178     self.rect = self.image.get_rect(center=actor.rect.center)
179     self.life = self.defaultlife
180
181 ✓ def update(self):
182     """called every time around the game loop.
183
184     Show the explosion surface for 'defaultlife'.
185     Every game tick(update), we decrease the 'life'.
186
187     Also we animate the explosion.
188     """
189     self.life = self.life - 1
190     self.image = self.images[self.life // self.animcycle % 2]
191     if self.life <= 0:
192         self.kill()
193
194
195 ✓ class Shot(pg.sprite.Sprite):
196     """a bullet the Player sprite fires."""
197
198     speed = -11
199     images = []
200
201     def __init__(self, pos):
202         pg.sprite.Sprite.__init__(self, self.containers)
203         self.image = self.images[0]
204         self.rect = self.image.get_rect(midbottom=pos)
205
206 ✓ def update(self):
207     """called every time around the game loop.
208
209     Every tick we move the shot upwards.
210     """
211     self.rect.move_ip(0, self.speed)
212     if self.rect.top <= 0:
213         self.kill()
214
215
216 ✓ class Bomb(pg.sprite.Sprite):
217     """A bomb the aliens drop."""
218
219     speed = 9
220     images = []
221
222     def __init__(self, alien):
223         pg.sprite.Sprite.__init__(self, self.containers)
224         self.image = self.images[0]
```

```

225         self.rect = self.image.get_rect(midbottom=alien.rect.move(0, 5).midbottom)
226
227     def update(self):
228         """called every time around the game loop.
229
230         Every frame we move the sprite 'rect' down.
231         When it reaches the bottom we:
232
233         - make an explosion.
234         - remove the Bomb.
235         """
236         self.rect.move_ip(0, self.speed)
237         if self.rect.bottom >= 470:
238             Explosion(self)
239             self.kill()
240
241
242     class Score(pg.sprite.Sprite):
243         """to keep track of the score."""
244
245     def __init__(self):
246         pg.sprite.Sprite.__init__(self)
247         self.font = pg.font.Font(None, 20)
248         self.font.set_italic(1)
249         self.color = "white"
250         self.lastscore = -1
251         self.update()
252         self.rect = self.image.get_rect().move(10, 450)
253
254     def update(self):
255         """We only update the score in update() when it has changed."""
256         if SCORE != self.lastscore:
257             self.lastscore = SCORE
258             msg = "Score: %d" % SCORE
259             self.image = self.font.render(msg, 0, self.color)
260
261
262     def main(winstyle=0):
263         # Initialize pygame
264         if pg.get_sdl_version()[0] == 2:
265             pg.mixer.pre_init(44100, 32, 2, 1024)
266         pg.init()
267         if pg.mixer and not pg.mixer.get_init():
268             print("Warning, no sound")
269             pg.mixer = None
270
271         fullscreen = False
272         # Set the display mode
273         winstyle = 0 # |FULLSCREEN
274         bestdepth = pg.display.mode_ok(SCREENRECT.size, winstyle, 32)
275         screen = pg.display.set_mode(SCREENRECT.size, winstyle, bestdepth)
276
277         # Load images, assign to sprite classes
278         # (do this before the classes are used, after screen setup)
279         img = load_image("player1.gif")
280         Player1.images = [img, pg.transform.flip(img, 1, 0)]
281         img = load_image("player2.gif") # 2台目の戦車
282         Player2.images = [img, pg.transform.flip(img, -1, 0)] # 2台目の戦車
283         img = load_image("explosion1.gif")
284         Explosion.images = [img, pg.transform.flip(img, 1, 1)]
285         Alien.images = [load_image(im) for im in ("alien1.gif", "alien2.gif", "alien3.gif")]

```

```

285 Alien.images = [load_image(main_dir + "/alien1.gif", alien1.gif, alien2.gif, alien3.gif)]
286 Bomb.images = [load_image("bomb.gif")]
287 Shot.images = [load_image("shot.gif")]
288
289 # decorate the game window
290 icon = pg.transform.scale(Alien.images[0], (32, 32))
291 pg.display.set_icon(icon)
292 pg.display.set_caption("Pygame Aliens")
293 pg.mouse.set_visible(0)
294
295 # create the background, tile the bgd image
296 bgdtile = load_image("background.gif")
297 background = pg.Surface(SCREENRECT.size)
298 for x in range(0, SCREENRECT.width, bgdtile.get_width()):
299     background.blit(bgdtile, (x, 0))
300 screen.blit(background, (0, 0))
301 pg.display.flip()
302
303 # load the sound effects
304 boom_sound = load_sound("boom.wav")
305 shoot_sound = load_sound("car_door.wav")
306 if pg.mixer:
307     music = os.path.join(main_dir, "data", "house_lo.wav")
308     pg.mixer.music.load(music)
309     pg.mixer.music.play(-1)
310
311 # Initialize Game Groups
312 aliens = pg.sprite.Group()
313 shots = pg.sprite.Group()
314 bombs = pg.sprite.Group()
315 all = pg.sprite.RenderUpdates()
316 lastalien = pg.sprite.GroupSingle()
317
318 # assign default groups to each sprite class
319 Player1.containers = all
320 Player2.containers = all # 2台目の戦車
321 Alien.containers = aliens, all, lastalien
322 Shot.containers = shots, all
323 Bomb.containers = bombs, all
324 Explosion.containers = all
325 Score.containers = all
326
327 # Create Some Starting Values
328 global score
329 alienreload = ALIEN_RELOAD
330 clock = pg.time.Clock()
331
332 # initialize our starting sprites
333 global SCORE
334 player1 = Player1()
335 player2 = Player2() # 2台目の戦車
336 Alien() # note, this 'lives' because it goes into a sprite group
337 if pg.font:
338     all.add(Score())
339
340 # Run our main loop whilst the player is alive.
341 while player1.alive():
342
343     # get input
344     for event in pg.event.get():
345         if event.type == pg.QUIT:

```



```

345         return
346     return
347     if event.type == pg.KEYDOWN and event.key == pg.K_ESCAPE:
348         return
349     elif event.type == pg.KEYDOWN:
350         if event.key == pg.K_f:
351             if not fullscreen:
352                 print("Changing to FULLSCREEN")
353                 screen_backup = screen.copy()
354                 screen = pg.display.set_mode(
355                     SCREENRECT.size, winstyle | pg.FULLSCREEN, bestdepth
356                 )
357                 screen.blit(screen_backup, (0, 0))
358             else:
359                 print("Changing to windowed mode")
360                 screen_backup = screen.copy()
361                 screen = pg.display.set_mode(
362                     SCREENRECT.size, winstyle, bestdepth
363                 )
364                 screen.blit(screen_backup, (0, 0))
365             pg.display.flip()
366             fullscreen = not fullscreen
367
368
369
370
371     keystate = pg.key.get_pressed()
372
373     # clear/erase the last drawn sprites
374     all.clear(screen, background)
375
376     # update all the sprites
377     all.update()
378
379     # handle player input
380     direction1 = keystate[pg.K_d] - keystate[pg.K_a]
381     player1.move(direction1)
382     direction2 = keystate[pg.K_l] - keystate[pg.K_j]
383     player2.move(direction2) # 2台目の戦車
384     firing1 = keystate[pg.K_w]
385     firing2 = keystate[pg.K_i]
386     if not player1.reloading and firing1 and len(shots) < MAX_SHOTS:
387         Shot(player1.gunpos())
388         if pg.mixer:
389             shoot_sound.play()
390     player1.reloading = firing1
391
392     if not player2.reloading and firing2 and len(shots) < MAX_SHOTS: # 2台目の戦車
393         Shot(player2.gunpos())
394         if pg.mixer:
395             shoot_sound.play()
396     player2.reloading = firing2 # 2台目の戦車
397
398
399     # Create new alien
400     if alienreload:
401         alienreload = alienreload - 1
402     elif not int(random.random() * ALIEN_ODDS):
403         Alien()
404         alienreload = ALIEN_RELOAD
405

```

```
406         # Drop bombs
407         if lastalien and not int(random.random() * BOMB_ODDS):
408             Bomb(lastalien.sprite)
409
410         # Detect collisions between aliens and players.
411         for alien in pg.sprite.spritecollide(player1, aliens, 1):
412             if pg.mixer:
413                 boom_sound.play()
414             Explosion(alien)
415             Explosion(player1)
416             SCORE = SCORE + 1
417             player1.kill()
418         for alien in pg.sprite.spritecollide(player2, aliens, 1): # 2台目の戦車
419             if pg.mixer:
420                 boom_sound.play()
421             Explosion(alien)
422             Explosion(player2)
423             SCORE = SCORE + 1
424             player2.kill()
425
426         # See if shots hit the aliens.
427         for alien in pg.sprite.groupcollide(aliens, shots, 1, 1).keys():
428             if pg.mixer:
429                 boom_sound.play()
430             Explosion(alien)
431             SCORE = SCORE + 1
432
433         # See if alien boms hit the player.
434         for bomb in pg.sprite.spritecollide(player1, bombs, 1):
435             if pg.mixer:
436                 boom_sound.play()
437             Explosion(player1)
438             Explosion(bomb)
439             player1.kill()
440
441
442
443         for bomb in pg.sprite.spritecollide(player2, bombs, 1): # 2台目の戦車
444             if pg.mixer:
445                 boom_sound.play()
446             Explosion(player2)
447             Explosion(bomb)
448             player2.kill()
449
450         # draw the scene
451         dirty = all.draw(screen)
452         pg.display.update(dirty)
453
454         # cap the framerate at 40fps. Also called 40HZ or 40 times per second.
455         clock.tick(40)
456
457
458         if pg.mixer:
459             pg.mixer.music.fadeout(1000)
460         pg.time.wait(1000)
461
462
463         # call the "main" function if running this script
464         if __name__ == "__main__":
465             main()
```

466

pg.quit()