





 c0a2302825 / ProjExD_Group13




[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

  doragon ▾

ProjExD_Group13 / Shouting_game.py



 t ...



c0a2302825 doragon

a3c8fca · 1 hour ago



296 lines (254 loc) · 10.9 KB

[Code](#) [Blame](#)

[Raw](#)     

```
1  import pygame as pg
2  import random
3  import math
4
5  pg.init()
6
7  # ディスプレイの設定
8  SCREEN_WIDTH = pg.display.Info().current_w
9  SCREEN_HEIGHT = pg.display.Info().current_h
10 screen = pg.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT), pg.FULLSCREEN)
11 pg.display.set_caption("しゅうていんぐげえむ")
12
13 # 色の定義
14 WHITE = (255, 255, 255)
15 RED = (255, 0, 0)
16 GREEN = (0, 255, 0)
17 BLUE = (0, 0, 255)
18
19 # ゲームエリア
20 GAME_AREA_SIZE = min(SCREEN_WIDTH, SCREEN_HEIGHT) * 0.6
21 GAME_AREA_X = (SCREEN_WIDTH - GAME_AREA_SIZE) / 2
22 GAME_AREA_Y = (SCREEN_HEIGHT - GAME_AREA_SIZE) / 2
23
24
25 class Player:
26     """
27     Playerの操作するキャラのクラス
28     """
29     def __init__(self):
30         self.width = 50
31         self.height = 50
32         self.x = SCREEN_WIDTH // 2 - self.width // 2
33         self.y = SCREEN_HEIGHT // 2 - self.height // 2
34         self.speed = 5
35         self.hp = 100 # プレイヤーHPの追加（初期化）
36         self.sp = 0 # プレイヤーSP(スキルポイント)の追加（初期化）
37
38     def move(self, dx:int, dy:int):
39         """
40         自機を速度ベクトルself.x,self.yに基づき、
41         new_x,new_yとして移動させる
```

```
42     プレイヤーの行動範囲を制御する
43     """
44     new_x = self.x + dx * self.speed
45     new_y = self.y + dy * self.speed
46     if (GAME_AREA_X < new_x < GAME_AREA_X + GAME_AREA_SIZE - self.width and
47         GAME_AREA_Y < new_y < GAME_AREA_Y + GAME_AREA_SIZE - self.height):
48         self.x = new_x
49         self.y = new_y
50
51     def draw(self, screen: pg.Surface):
52         """
53         引数 screen : 画面surface
54         """
55         pg.draw.rect(screen, BLUE, (self.x, self.y, self.width, self.height))
56
57
58     class Enemy:
59         """
60         敵キャラを表示するクラス
61         """
62     def __init__(self):
63         self.width = 60
64         self.height = 60
65         self.x = random.randint(0, SCREEN_WIDTH - self.width)
66         self.y = 50
67         self.speed = random.uniform(2, 5)
68         self.hp = 100 # 敵HPの追加 (初期化)
69         self.direction = random.choice([-1, 1])
70         self.change_direction_counter = 0 # 敵の移動判定のカウンターの初期化
71         self.change_direction_threshold = random.randint(60, 180) # 敵の停止時間をランダム値で設定
72         self.image = pg.image.load("ex5/fig/doragon.3.png").convert_alpha() # ドラゴンの画像をロード
73         self.image = pg.transform.rotozoom(self.image, 0, 0.05) # 画像のサイズを調整
74
75     def move(self):
76         """
77         敵キャラを速度ベクトルself.x,self.directionに基づき移動させる
78         """
79         self.x += self.speed * self.direction
80         if self.x <= 0 or self.x >= SCREEN_WIDTH - self.width:
81             self.direction *= -1
82
83         self.change_direction_counter += 1 # 敵の移動判定のカウンターの更新
84         if self.change_direction_counter >= self.change_direction_threshold: # 敵の停止時間超えたら敵
85             self.direction = random.choice([-1, 1]) # 敵の動くy軸+-の方向をランダムに設定
86             self.speed = random.uniform(2, 5) # 敵の移動量をランダムに設定
87             self.change_direction_counter = 0 # 敵の移動判定のカウンターのリセット
88             self.change_direction_threshold = random.randint(60, 180) # 敵の停止時間をランダム値で設定
89
90     def draw(self, screen: pg.Surface):
91         """
92         引数 screen : 画面surface
93         ドラゴンの画像の描写
94         """
95         screen.blit(self.image, (self.x, self.y)) # 画像を描画
96
97
98     class Bullet:
99         """
```

```
100     敵味方が攻撃を行う弾を表すクラス。
101
102     変数:
103         x : 弾の現在のx座標
104         y : 弾の現在のy座標
105         dx : x方向の移動速度
106         dy : y方向の移動速度
107
108     メソッド:
109         move(): 弾を移動させる
110         draw(screen): 弾を画面上に描画する
111     """
112     def __init__(self, x:float, y:float, target_x:float, target_y:float):
113         """
114         Bulletオブジェクトを初期化する。
115
116         引数:
117             x : 弾の初期x座標
118             y : 弾の初期y座標
119             target_x : プレイヤーのx座標
120             target_y : プレイヤーのy座標
121         """
122         self.x = x
123         self.y = y
124         angle = math.atan2(target_y - y, target_x - x)
125         speed = 5
126         self.dx = math.cos(angle) * speed
127         self.dy = math.sin(angle) * speed
128
129     def move(self):
130         """弾を現在の速度に基づいて移動させる。"""
131         self.x += self.dx
132         self.y += self.dy
133
134     def draw(self, screen: pg.Surface):
135         """
136         弾を画面上に描画する。
137
138         引数:
139             screen (pygame.Surface): 描画対象の画面
140         """
141         pg.draw.circle(screen, WHITE, (int(self.x), int(self.y)), 5)
142
143
144     class OmniBullet:
145         """
146         プレイヤーが全方向に発射する弾のクラス。
147         属性:
148             bullets (list): 発射された弾を保持するリスト。
149         メソッド:
150             __init__(x, y): OmniBulletオブジェクトを初期化する。
151             move(): 全ての弾を移動させる。
152             draw(screen): 全ての弾を画面上に描画する。
153         """
154     def __init__(self, x, y):
155         """
156         OmniBulletオブジェクトを初期化する。
157         引数:
```

```
158         x (float): 弾の初期x座標。
159         y (float): 弾の初期y座標。
160     """
161     self.bullets = []
162     speed = 5
163     for angle in range(0, 360, 45): # 45度間隔で全方向に弾を作成
164         radians = math.radians(angle)
165         dx = math.cos(radians) * speed
166         dy = math.sin(radians) * speed
167         self.bullets.append(Bullet(x, y, x + dx * 10, y + dy * 10))
168     def move(self):
169         """
170         全ての弾を現在の速度に基づいて移動させる。
171         """
172         for bullet in self.bullets:
173             bullet.move()
174     def draw(self, screen):
175         """
176         全ての弾を画面上に描画する。
177         引数:
178             screen (pygame.Surface): 描画対象の画面。
179         """
180         for bullet in self.bullets:
181             bullet.draw(screen)
182
183     def main():
184         player = Player()
185         enemy = Enemy() # enemy関数の呼び出し
186         player_bullets = [] # プレイヤーと敵の弾を保持するリスト
187         enemy_bullets = []
188         clock = pg.time.Clock()
189         omni_bullets = [] # 全方向攻撃の弾を保持するリスト
190
191         running = True
192         while running:
193             for event in pg.event.get():
194                 if event.type == pg.QUIT:
195                     running = False
196                 elif event.type == pg.KEYDOWN:
197                     if event.key == pg.K_ESCAPE:
198                         running = False
199                     elif event.key == pg.K_SPACE: # スペースキーで弾の発射
200                         player_bullets.append(Bullet(player.x + player.width // 2, player.y,
201                                                         player.x + player.width // 2, 0))
202                     elif event.key == pg.K_z: # Zキーで全方向攻撃
203                         if player.sp >= 5: # SPゲージが5以上の場合
204                             omni_bullets.append(OmniBullet(player.x + player.width // 2, player.y + player.height // 2,
205                                                                player.x + player.width // 2, player.y + player.height // 2,
206                                                                player.sp - 5 # SPゲージ5を消費して全方向攻撃
207
208                         )
209             keys = pg.key.get_pressed()
210             player.move(keys[pg.K_RIGHT] - keys[pg.K_LEFT], keys[pg.K_DOWN] - keys[pg.K_UP])
211
212             enemy.move()
213
214             if random.random() < 0.02: # 弾の発生
215                 # 画面の四辺からランダムに弾を発射
```

```
126 side = random.choice(['top', 'bottom', 'left', 'right'])
127 if side == 'top':
128     x = random.randint(0, SCREEN_WIDTH)
129     y = 0
130 elif side == 'bottom':
131     x = random.randint(0, SCREEN_WIDTH)
132     y = SCREEN_HEIGHT
133 elif side == 'left':
134     x = 0
135     y = random.randint(0, SCREEN_HEIGHT)
136 else: # right
137     x = SCREEN_WIDTH
138     y = random.randint(0, SCREEN_HEIGHT)
139
140 target_x = GAME_AREA_X + GAME_AREA_SIZE // 2
141 target_y = GAME_AREA_Y + GAME_AREA_SIZE // 2
142 enemy_bullets.append(Bullet(x, y, target_x, target_y))
143
144 # 全方向攻撃の弾の移動と当たり判定
145 for omni in omni_bullets[:]:
146     for bullet in omni.bullets[:]:
147         bullet.move()
148         if (bullet.x < 0 or bullet.x > SCREEN_WIDTH or
149             bullet.y < 0 or bullet.y > SCREEN_HEIGHT):
150             omni.bullets.remove(bullet)
151         elif (enemy.x < bullet.x < enemy.x + enemy.width and
152             enemy.y < bullet.y < enemy.y + enemy.height):
153             enemy.hp -= 10 # 敵HPの更新
154             omni.bullets.remove(bullet)
155     if not omni.bullets:
156         omni_bullets.remove(omni)
157
158 # プレイヤーの弾の移動と当たり判定
159 for bullet in player_bullets[:]: # 弾の動きと衝突
160     bullet.move()
161     if bullet.y < 0:
162         player_bullets.remove(bullet)
163     elif (enemy.x < bullet.x < enemy.x + enemy.width and
164         enemy.y < bullet.y < enemy.y + enemy.height):
165         enemy.hp -= 10 # 敵HPの更新
166         player.sp += 5 # プレイヤーSPの更新
167         player_bullets.remove(bullet)
168
169 # 敵の弾の移動と当たり判定
170 for bullet in enemy_bullets[:]:
171     bullet.move()
172     if (bullet.x < 0 or bullet.x > SCREEN_WIDTH or
173         bullet.y < 0 or bullet.y > SCREEN_HEIGHT):
174         enemy_bullets.remove(bullet)
175     elif (player.x < bullet.x < player.x + player.width and
176         player.y < bullet.y < player.y + player.height):
177         player.hp -= 1 # プレイヤーHPの更新
178         enemy_bullets.remove(bullet)
179
180 if player.hp <= 0 or enemy.hp <= 0: # ゲームの終了判定
181     running = False # ゲームを終了させる
182
183 screen.fill((0, 0, 0))
```

```
274     # プレイヤーの行動範囲を視覚的に表示する
275     pg.draw.rect(screen, WHITE, (GAME_AREA_X, GAME_AREA_Y, GAME_AREA_SIZE, GAME_AREA_SIZE), 2)
276     player.draw(screen)
277     # 敵キャラを表示
278     enemy.draw(screen)
279     for bullet in player_bullets + enemy_bullets: # 弾の描画
280         bullet.draw(screen)
281     for omni in omni_bullets:
282         omni.draw(screen)
283
284
285     pg.draw.rect(screen, RED, (10, SCREEN_HEIGHT - 30, player.hp * 2, 20)) # プレイヤーHPのゲージ
286     pg.draw.rect(screen, GREEN, (10, 10, enemy.hp * 2, 20)) # 敵HPのゲージを表示
287     pg.draw.rect(screen, BLUE, (SCREEN_WIDTH - 210, SCREEN_HEIGHT - 30, player.sp * 2, 20)) # プ
288
289     pg.display.flip()
290     clock.tick(60)
291
292     pg.quit()
293
294
295 if __name__ == "__main__":
296     main()
```