




 c0a2302825 / ProjExD_Group13





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

 Berserk ▾

ProjExD_Group13 / Shouting_game.py





c0a2302825 Berserk

e44b2d4 · 37 minutes ago



318 lines (270 loc) · 12.2 KB

```
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         """
64         敵キャラとして、ドラゴンの画像をロードする。
65         """
66         self.width = 60
67         self.height = 60
68         self.x = random.randint(0, SCREEN_WIDTH - self.width)
69         self.y = 50
70         self.speed = random.uniform(2, 5)
71         self.hp = 100 # 敵HPの追加 (初期化)
72         self.direction = random.choice([-1, 1])
73         self.change_direction_counter = 0 # 敵の移動判定のカウンターの初期化
74         self.change_direction_threshold = random.randint(60, 180) # 敵の停止時間をランダム値で設定
75         self.image = pg.image.load("ex5/fig/doragon.3.png").convert_alpha() # ドラゴンの画像をロード
76         self.image = pg.transform.rotozoom(self.image, 0, 0.05) # 画像のサイズを調整
77
78     def move(self):
79         """
80         敵キャラを速度ベクトルself.x,self.directionに基づき移動させる
81         """
82         self.x += self.speed * self.direction
83         if self.x <= 0 or self.x >= SCREEN_WIDTH - self.width:
84             self.direction *= -1
85
86         self.change_direction_counter += 1 # 敵の移動判定のカウンターの更新
87         if self.change_direction_counter >= self.change_direction_threshold: # 敵の停止時間超えたら敵:
88             self.direction = random.choice([-1, 1]) # 敵の動くy軸+-の方向をランダムに設定
89             self.speed = random.uniform(2, 5) # 敵の移動量をランダムに設定
90             self.change_direction_counter = 0 # 敵の移動判定のカウンターのリセット
91             self.change_direction_threshold = random.randint(60, 180) # 敵の停止時間をランダム値で設定
92
93     def draw(self, screen: pg.Surface):
94         """
95         引数 screen : 画面surface
96         ドラゴンの画像の描写
97         """
98         screen.blit(self.image, (self.x, self.y)) # 画像を描画
99

```

```
100
101 ✓ class Bullet:
102     """
103     敵味方が攻撃を行う弾を表すクラス。
104
105     変数:
106         x : 弾の現在のx座標
107         y : 弾の現在のy座標
108         dx : x方向の移動速度
109         dy : y方向の移動速度
110
111     メソッド:
112         move(): 弾を移動させる
113         draw(screen): 弾を画面上に描画する
114     """
115 ✓ def __init__(self, x:float, y:float, target_x:float, target_y:float, speed:float=5.0):
116     """
117     Bulletオブジェクトを初期化する。
118
119     引数:
120         x : 弾の初期x座標
121         y : 弾の初期y座標
122         target_x : プレイヤーのx座標
123         target_y : プレイヤーのy座標
124         speed : 弾の移動速度 (デフォルトは5.0)
125     """
126     self.x = x
127     self.y = y
128     self.speed = speed
129     angle = math.atan2(target_y - y, target_x - x)
130     self.dx = math.cos(angle) * self.speed
131     self.dy = math.sin(angle) * self.speed
132
133     def move(self):
134         """弾を現在の速度に基づいて移動させる。"""
135         self.x += self.dx
136         self.y += self.dy
137
138 ✓ def draw(self, screen: pg.Surface):
139     """
140     弾を画面上に描画する。
141
142     引数:
143         screen (pygame.Surface): 描画対象の画面
144     """
145     pg.draw.circle(screen, WHITE, (int(self.x), int(self.y)), 5)
146
147
148 ✓ class OmniBullet:
149 ✓ def __init__(self, x, y):
150     self.bullets = []
151     speed = 5
152     for angle in range(0, 360, 45): # 45度間隔で全方向に弾を作成
153         radians = math.radians(angle)
154         dx = math.cos(radians) * speed
155         dy = math.sin(radians) * speed
156         self.bullets.append(Bullet(x, y, x + dx * 10, y + dy * 10))
157
```

```
158     def move(self):
159         for bullet in self.bullets:
160             bullet.move()
161
162     def draw(self, screen):
163         for bullet in self.bullets:
164             bullet.draw(screen)
165
166
167     class AdvancedEnemy(Enemy):
168         """
169         追尾攻撃を行う敵キャラクター
170         """
171     def __init__(self):
172         super().__init__()
173         self.last_attack_time = pg.time.get_ticks() # 最後の攻撃時間を初期化
174         self.attack_cooldown = 1000 # 攻撃クールダウンを1秒（1000ミリ秒）に設定
175         self.cut_in_image = pg.image.load("ex5/fig/doragon.3.png").convert_alpha() # カットイン画像を
176         self.cut_in_image = pg.transform.rotozoom(self.cut_in_image, 0, 1.0) # 画像のサイズを調整
177         self.has_cut_in = False # カットインが行われたかどうかを管理するフラグ
178
179     def attack(self, player_x, player_y):
180         """
181         敵の攻撃処理
182         敵のhpが80%未満の時、一定間隔で攻撃される。
183         敵のhpが50%未満の時、画像をカットインさせ、敵の攻撃速度が増加する（1度だけ）。
184         """
185         current_time = pg.time.get_ticks()
186         if self.hp <= 80 and current_time - self.last_attack_time > self.attack_cooldown:
187             self.last_attack_time = current_time
188             return Bullet(self.x + self.width // 2, self.y + self.height // 2, player_x, player_y)
189         if self.hp <= 50:
190             if not self.has_cut_in:
191                 self.has_cut_in = True
192                 # カットイン表示
193                 screen.blit(self.cut_in_image, (SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2))
194                 pg.display.flip()
195                 pg.time.wait(1000) # カットインを表示する時間
196                 self.attack_cooldown = 400 # 攻撃速度を増加させる
197             return None # ここで return None を追加して、それ以外の場合は何も返さないようにする
198         return None
199
200
201     def main():
202         player = Player()
203         enemy = AdvancedEnemy() # enemy関数の呼び出し
204         player_bullets = [] # プレイヤーと敵の弾を保持するリスト
205         enemy_bullets = []
206         clock = pg.time.Clock()
207         omni_bullets = [] # 全方向攻撃の弾を保持するリスト
208
209
210         running = True
211         while running:
212             for event in pg.event.get():
213                 if event.type == pg.QUIT:
214                     running = False
215                 elif event.type == pg.KEYDOWN:
```

```
116         if event.key == pg.K_ESCAPE:
117             running = False
118         elif event.key == pg.K_SPACE: # スペースキーで弾の発射
119             player_bullets.append(Bullet(player.x + player.width // 2, player.y,
120                                         player.x + player.width // 2, 0))
121         elif event.key == pg.K_z: # Zキーで全方向攻撃
122             if player.sp >= 5: # SPゲージが5以上の場合
123                 omni_bullets.append(OmniBullet(player.x + player.width // 2, player.y + player.height // 2,
124                                                player.sp - 5))
125
126     keys = pg.key.get_pressed()
127     player.move(keys[pg.K_RIGHT] - keys[pg.K_LEFT], keys[pg.K_DOWN] - keys[pg.K_UP])
128
129     enemy.move()
130
131     # 敵の攻撃処理
132     enemy_bullet = enemy.attack(player.x + player.width // 2, player.y + player.height // 2)
133     if enemy_bullet:
134         enemy_bullets.append(enemy_bullet)
135
136     if random.random() < 0.02: # 弾の発生
137         # 画面の四辺からランダムに弾を発射
138         side = random.choice(['top', 'bottom', 'left', 'right'])
139         if side == 'top':
140             x = random.randint(0, SCREEN_WIDTH)
141             y = 0
142         elif side == 'bottom':
143             x = random.randint(0, SCREEN_WIDTH)
144             y = SCREEN_HEIGHT
145         elif side == 'left':
146             x = 0
147             y = random.randint(0, SCREEN_HEIGHT)
148         else: # right
149             x = SCREEN_WIDTH
150             y = random.randint(0, SCREEN_HEIGHT)
151
152         target_x = GAME_AREA_X + GAME_AREA_SIZE // 2
153         target_y = GAME_AREA_Y + GAME_AREA_SIZE // 2
154         enemy_bullets.append(Bullet(x, y, target_x, target_y))
155
156     # プレイヤーの弾の移動と当たり判定
157     for bullet in player_bullets[:]: # 弾の動きと衝突
158         bullet.move()
159         if bullet.y < 0:
160             player_bullets.remove(bullet)
161         elif (enemy.x < bullet.x < enemy.x + enemy.width and
162              enemy.y < bullet.y < enemy.y + enemy.height):
163             enemy.hp -= 10 # 敵HPの更新
164             player.sp += 5 # プレイヤーSPの更新
165             player_bullets.remove(bullet)
166
167     # 全方向攻撃の弾の移動と当たり判定
168     for omni in omni_bullets[:]:
169         for bullet in omni.bullets[:]:
170             bullet.move()
171             if (bullet.x < 0 or bullet.x > SCREEN_WIDTH or
172                 bullet.y < 0 or bullet.y > SCREEN_HEIGHT):
```

```
274         omni.bullets.remove(bullet)
275         elif (enemy.x < bullet.x < enemy.x + enemy.width and
276               enemy.y < bullet.y < enemy.y + enemy.height):
277             enemy.hp -= 10 # 敵HPの更新
278             omni.bullets.remove(bullet)
279         if not omni.bullets:
280             omni_bullets.remove(omni)
281
282     # 敵の弾の移動と当たり判定
283     for bullet in enemy_bullets[:]:
284         bullet.move()
285         if (bullet.x < 0 or bullet.x > SCREEN_WIDTH or
286             bullet.y < 0 or bullet.y > SCREEN_HEIGHT):
287             enemy_bullets.remove(bullet)
288         elif (player.x < bullet.x < player.x + player.width and
289               player.y < bullet.y < player.y + player.height):
290             player.hp -= 1 # プレイヤーHPの更新
291             enemy_bullets.remove(bullet)
292
293     if player.hp <= 0 or enemy.hp <= 0: # ゲームの終了判定
294         running = False # ゲームを終了させる
295
296     screen.fill((0, 0, 0))
297     # プレイヤーの行動範囲を視覚的に表示する
298     pg.draw.rect(screen, WHITE, (GAME_AREA_X, GAME_AREA_Y, GAME_AREA_SIZE, GAME_AREA_SIZE), 2)
299     player.draw(screen)
300     # 敵キャラを表示
301     enemy.draw(screen)
302     for bullet in player_bullets + enemy_bullets: # 弾の描画
303         bullet.draw(screen)
304     for omni in omni_bullets: # 全方向攻撃の玉の描写
305         omni.draw(screen)
306
307     pg.draw.rect(screen, RED, (10, SCREEN_HEIGHT - 30, player.hp * 2, 20)) # プレイヤーHPのゲージ
308     pg.draw.rect(screen, GREEN, (10, 10, enemy.hp * 2, 20)) # 敵HPのゲージを表示
309     pg.draw.rect(screen, BLUE, (SCREEN_WIDTH - 210, SCREEN_HEIGHT - 30, player.sp * 2, 20)) # プ
310
311     pg.display.flip()
312     clock.tick(60)
313
314     pg.quit()
315
316
317 if __name__ == "__main__":
318     main()
```