

c0a2302825 / ProjExD_Group13

🔍

📧

🏠

<> Code

🔗 Pull requests

🔄 Actions

📁 Projects

📖 Wiki

🛡 Security

📈 Insights

⚙ Settings



☆ 0 stars 🍴 4 forks 👁 0 watching 🌿 3 Branches 🏷 0 Tags 🔄 Activity

🌐 Public repository · Forked from [c0a23140d8/ProjExD_Group13](#)

doragon had recent pushes 26 minutes ago

Compare & pull request

3 Branches 🏷 0 Tags 🍴 📁

🔍 Go to file t

Go to file + Add file <> Code ...

This branch is **1 commit ahead of, 8 commits behind** `c0a23140d8/ProjExD_Group13:main`.

Contribute 🔄 Sync fork

c0a2302825 全方位攻撃	866d5cc · 1 hour ago	🕒
README.md	READMEの修正	last week
Shouting_game.py	全方位攻撃	1 hour ago

📖 README ✎ ⋮

しゅうていんぐげえむ

開発環境

- 言語 : Python 3.10.9
- フレームワーク : PyGame 2.5.2

1. ゲーム概要

しゅうていんぐげえむは2Dシューティングゲームである。ゲーム内容はプレイヤーは中央の制限エリア内で敵の攻撃をかわしながら移動しながら、上部を移動する敵を倒すことである。

2. ゲーム画面レイアウト

- 中央 : 正方形の自機移動エリア
- 左下 : 自機のHPゲージ
- 左上 : 敵のHPゲージ
- 右下 : 自機のSPゲージ

3. ゲームオブジェクト

a. プレイヤー

- 戦闘機として表示
- 初期位置：画面中央
- 中央の正方形エリア内のみ移動可能
- 真上にのみ攻撃可能

b. 敵

- 画面上部に位置
- Y軸固定でX軸上を左右に不規則に移動
- 移動速度はランダム（2～5の間）
- 1～3秒ごとにランダムに方向変更

4. システム

a. プレイヤーの攻撃

- 真上にのみ発射可能
- 敵に当たるとダメージを与える
- 敵に当てるとSPが増加

b. 敵の攻撃

- 画面全体から四方八方に攻撃
- 中央の正方形エリアをめがけて攻撃
- 自機に当たるとHPが1減少

c. SPシステム

- 敵に攻撃を当てるとSPが増加
- 一定量たまるとスキルが使用可能

5. ゲームオーバー条件

- 自機のHPが0になった時

6. クリア条件

- 敵のHPが0になった時

7. 実装要件

- クラス設計（最低限）：
 - Player：自機の挙動とHP, SP管理
 - Enemy：敵の挙動とHP管理
 - Bullet：弾丸オブジェクトの管理（自機と敵の両方）
- main関数： 各クラス（Player、Enemy、Bullet）のインスタンス作成、ゲームループの実行

追加検討事項

1. スコアシステムの追加（クリアタイム）[小松原]
2. 音響効果の実装 [松岡]
3. レベルシステムの導入（ゲーム進行に伴う難易度上昇 → 複数の敵の追加）[倉本]
4. スキル（プレイヤー、敵 両方）の実装 [田辺, 大本]



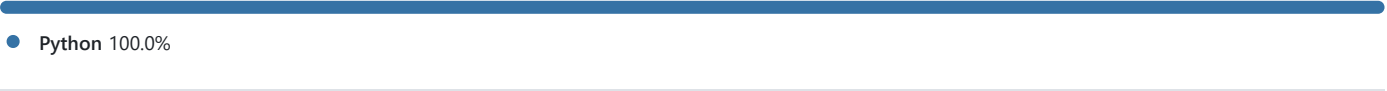
Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages



Suggested workflows

Based on your tech stack

Publish Python Package

Publish a Python Package to PyPI on release.

Configure

Python application



Create and test a Python application.




Configure

Django


Build and Test a Django Project

Configure


 c0a2302825 / ProjExD_Group13




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

 allatack ▾

ProjExD_Group13 / Shouting_game.py



 t ...



c0a2302825 全方位攻撃

866d5cc · 1 hour ago



293 lines (251 loc) · 10.6 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
73         def move(self):
74             """
75             敵キャラを速度ベクトルself.x,self.directionに基づき移動させる
76             """
77             self.x += self.speed * self.direction
78             if self.x <= 0 or self.x >= SCREEN_WIDTH - self.width:
79                 self.direction *= -1
80
81             self.change_direction_counter += 1 # 敵の移動判定のカウンターの更新
82             if self.change_direction_counter >= self.change_direction_threshold: # 敵の停止時間を超えたら敵:
83                 self.direction = random.choice([-1, 1]) # 敵の動くy軸+-の方向をランダムに設定
84                 self.speed = random.uniform(2, 5) # 敵の移動量をランダムに設定
85                 self.change_direction_counter = 0 # 敵の移動判定のカウンターのリセット
86                 self.change_direction_threshold = random.randint(60, 180) # 敵の停止時間をランダム値で設定
87
88         def draw(self, screen: pg.Surface):
89             """
90             引数 screen : 画面surface
91             """
92             pg.draw.rect(screen, RED, (self.x, self.y, self.width, self.height))
93
94
95     class Bullet:
96         """
97         敵味方が攻撃を行う弾を表すクラス。
98
99         変数:
```



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




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

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





```
274     # 敵キャラを表示
275     enemy.draw(screen)
276     for bullet in player_bullets + enemy_bullets: # 弾の描画
277         bullet.draw(screen)
278     for omni in omni_bullets:
279         omni.draw(screen)
280
281
282     pg.draw.rect(screen, RED, (10, SCREEN_HEIGHT - 30, player.hp * 2, 20)) # プレイヤーHPのゲージ
283     pg.draw.rect(screen, GREEN, (10, 10, enemy.hp * 2, 20)) # 敵HPのゲージを表示
284     pg.draw.rect(screen, BLUE, (SCREEN_WIDTH - 210, SCREEN_HEIGHT - 30, player.sp * 2, 20)) # フ
285
286     pg.display.flip()
287     clock.tick(60)
288
289     pg.quit()
290
291
292 if __name__ == "__main__":
293     main()
```


 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 Copy Download Edit View

```
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.sp - 5)) # SPゲージ5を消費して全方向攻撃
206
207             keys = pg.key.get_pressed()
208             player.move(keys[pg.K_RIGHT] - keys[pg.K_LEFT], keys[pg.K_DOWN] - keys[pg.K_UP])
209             enemy.move()
210
211             if random.random() < 0.02: # 弾の発生
212                 # 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)) # フ
288
289     pg.display.flip()
290     clock.tick(60)
291
292     pg.quit()
293
294
295 if __name__ == "__main__":
296     main()
```


 c0a2302825 / ProjExD_Group13





 Code  Pull requests  Actions  Projects  Wiki  Security  Insights  Settings

 Berserk ▾

ProjExD_Group13 / Shouting_game.py



 Go to file  t ...



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ラス
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()
```