327 lines (275 loc) · 10.1 KB

```python
import math
import random
import sys
import time


import pygame as pg


WIDTH = 1200  # ゲームウィンドウの幅
HEIGHT = 600  # ゲームウィンドウの高さ


def check_bound(obj: pg.Rect) -> tuple[bool, bool]:
    """
    オブジェクトが画面内か画面外かを判定し，真理値タプルを返す
    引数 obj：オブジェクト（爆弾，こうかとん，ビーム）SurfaceのRect
    戻り値：横方向，縦方向のはみ出し判定結果（画面内：True／画面外：False）
    """
    yoko, tate = True, True
    if obj.left < 0 or WIDTH < obj.right:  # 横方向のはみ出し判定
        yoko = False
    if obj.top < 0 or HEIGHT < obj.bottom:  # 縦方向のはみ出し判定
        tate = False
    return yoko, tate


def calc_orientation(org: pg.Rect, dst: pg.Rect) -> tuple[float, float]:
    """
    orgから見て，dstがどこにあるかを計算し，方向ベクトルをタプルで返す
    引数1 org：爆弾SurfaceのRect
    引数2 dst：こうかとんSurfaceのRect
    戻り値：orgから見たdstの方向ベクトルを表すタプル
    """
    x_diff, y_diff = dst.centerx-org.centerx, dst.centery-org.centery
    norm = math.sqrt(x_diff**2+y_diff**2)
    return x_diff/norm, y_diff/norm


class Bird(pg.sprite.Sprite):
    """
    ゲームキャラクター（こうかとん）に関するクラス
    """
    delta = {  # 押下キーと移動量の辞書
```

```python
            pg.K_UP: (0, -1),
            pg.K_DOWN: (0, +1),
        }

        def __init__(self, xy: tuple[int, int]):
            """
            こうかとん画像Surfaceを生成する
            引数1 xy：こうかとん画像の位置座標タプル
            """
            super().__init__()
            img0 = pg.transform.rotozoom(pg.image.load(f"ex05/fig/cat.png"), 0, 0.1)
            self.image = pg.transform.flip(img0, True, False)  # デフォルトのこうかとん
            self.dire = (+1, 0)
            self.rect = self.image.get_rect()
            self.rect.left = 0
            self.speed = 10


        def update(self, key_lst: list[bool], screen: pg.Surface):
            """
            押下キーに応じてこうかとんを移動させる
            引数1 key_lst：押下キーの真理値リスト
            引数2 screen：画面Surface
            """

            sum_mv = [0, 0]
            for k, mv in __class__.delta.items():
                if key_lst[k]:
                    self.rect.move_ip(+self.speed*mv[0], +self.speed*mv[1])
                    sum_mv[0] += mv[0]
                    sum_mv[1] += mv[1]
            if check_bound(self.rect) != (True, True):
                for k, mv in __class__.delta.items():
                    if key_lst[k]:
                        self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
            if not (sum_mv[0] == 0 and sum_mv[1] == 0):
                self.dire = tuple(sum_mv)

            screen.blit(self.image, self.rect)


        def get_direction(self) -> tuple[int, int]:
            return self.dire


    class Beam(pg.sprite.Sprite):
        """
        ビームに関するクラス
        """
        def __init__(self, bird: Bird):
            """
            引数に基づきビームSurfaceを生成する
            引数 bird：ビームを放つこうかとん
            """
            super().__init__()
            self.image = pg.transform.rotozoom(pg.image.load(f"ex05/fig/beam.png"), 0, 2.0)
            self.rect = self.image.get_rect()
            self.rect.left = bird.rect.right
            self.rect.centery = bird.rect.centery
            self.vx, self.vy = +5, 0
```

```
105             self.speed = 5
106
107
108         def update(self):
109             """
110             ビームを速度ベクトルself.vx, self.vyに基づき移動させる
111             引数 screen：画面Surface
112             """
113             self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
114             if check_bound(self.rect) != (True, True):
115                 self.kill()
116
117
118     class Explosion(pg.sprite.Sprite):
119         """
120         爆発に関するクラス
121         """
122         def __init__(self, obj: "Bomb|Enemy", life: int):
123             """
124             爆弾が爆発するエフェクトを生成する
125             引数1 obj：爆発するBombまたは敵機インスタンス
126             引数2 life：爆発時間
127             """
128             super().__init__()
129             img = pg.image.load("ex05/fig/explosion.gif")
130             self.imgs = [img, pg.transform.flip(img, 1, 1)]
131             self.image = self.imgs[0]
132             self.rect = self.image.get_rect(center=obj.rect.center)
133             self.life = life
134
135         def update(self):
136             """
137             爆発時間を1減算した爆発経過時間_lifeに応じて爆発画像を切り替えることで
138             爆発エフェクトを表現する
139             """
140             self.life -= 1
141             self.image = self.imgs[self.life//10%2]
142             if self.life < 0:
143                 self.kill()
144
145
146     class Enemy(pg.sprite.Sprite):
147         """
148         敵機に関するクラス
149         """
150         imgs = [pg.image.load(f"ex05/fig/alien{i}.png") for i in range(1, 4)]
151
152         def __init__(self):
153             super().__init__()
154             self.image = random.choice(__class__.imgs)
155             self.rect = self.image.get_rect()
156             self.rect.right = WIDTH
157             self.vy = +6
158             self.bound = random.randint(0, HEIGHT)  # 停止位置
159             self.state = "down"  # 降下状態or停止状態
160
161
162         def update(self):
163             """
164             敵機を速度ベクトルself.vyに基づき移動（降下）させる
```

```python
            ランタムに決めた停止位置_boundまで降下したら，_stateを停止状態に変更する
            引数 screen：画面Surface
            """
            if self.rect.centery > self.bound:
                self.vy = 0
                self.state = "stop"
            self.rect.centery += self.vy


    class Score:
        """
        打ち落とした爆弾，敵機の数をスコアとして表示するクラス
        爆弾：1点
        敵機：10点
        """
        def __init__(self):
            self.font = pg.font.Font(None, 50)
            self.color = (0, 0, 255)
            self.score = 0
            self.image = self.font.render(f"Score: {self.score}", 0, self.color)
            self.rect = self.image.get_rect()
            self.rect.center = 100, HEIGHT-50

        def score_up(self, add): #スコアを加算
            self.score += add


        def update(self, screen: pg.Surface):
            self.image = self.font.render(f"Score: {self.score}", 0, self.color)
            screen.blit(self.image, self.rect)

    class Last_boss(pg.sprite.Sprite):
        """
        ラスボス
        """
        def __init__(self):
            super().__init__()
            self.image = pg.transform.rotozoom(pg.image.load(f"ex05/fig/7.png"), 0, 3.0)
            self.rect = self.image.get_rect()
            self.rect.right = WIDTH
            self.vy = +1

        def update(self):
            self.rect.centery += self.vy * 8
            # 画面端に到達したら方向を反転させる
            if self.rect.bottom >= HEIGHT or self.rect.top <= 0:
                self.vy *= -1

    class Boss_life:
        """
        ボスの体力
        """
        def __init__(self):
            self.font = pg.font.Font(None, 50)
            self.color = (255, 0, 0)
            self.life = 10
            self.image = self.font.render(f"LIFE: {self.life}", 0, self.color)
            self.rect = self.image.get_rect()
            self.rect.center = 100, HEIGHT

        def boss_lifes(self, dm):
```

```
226             self.life += dm
227
228         def update(self, screen: pg.Surface):
229             self.image = self.font.render(f"LIFE: {self.life}", 0, self.color)
230             screen.blit(self.image, self.rect)
231
232     """
233     class Boss_beam(pg.sprite.Sprite):
```



```
239             self.rect.right = boss.rect.left
240             self.rect.centery = boss.rect.centery
241             self.vx, self.vy = -5, 0
242             self.speed = 5
243
244
245         def update(self):
246             ビームを速度ベクトルself.vx, self.vyに基づき移動させる
247             引数 screen：画面Surface
248             self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
249             if check_bound(self.rect) != (True, True):
250                 self.kill()
251     """
252
253     def main():
254         pg.display.set_caption("倒せ！こうかとん！")
255         screen = pg.display.set_mode((WIDTH, HEIGHT))
256         bg_img = pg.image.load("ex05/fig/pg_bg.jpg")
257         score = Score()
258         boss_life = Boss_life()
259         boses = Last_boss()
260         #boss_beam = pg.sprite.Group()
261         bird = Bird( (900, 400))
262         beams = pg.sprite.Group()
263         exps = pg.sprite.Group()
264         emys = pg.sprite.Group()
265         boss = pg.sprite.Group()
266         num = 0
267
268         tmr = 0
269         clock = pg.time.Clock()
270         while True:
271             key_lst = pg.key.get_pressed()
272             for event in pg.event.get():
273                 if event.type == pg.QUIT:
274                     return 0
275
276                 elif event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
277                     beams.add(Beam(bird))
278
279             for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
280                 exps.add(Explosion(emy, 100))  # 爆発エフェクト
281                 score.score_up(50)  # 10点アップ
282
283             if score.score >= 100 and num == 0:
284                 boss_life.update(screen)
285                 bg_img = pg.transform.rotozoom(pg.image.load(f"ex05/fig/pg_bg_2.jpg"), 0, 4.0)
```

```
286                 boss.add(Last_boss())
287                 num = 1
288             else:
289                 if tmr%200 == 0 and num == 0:
290                     emys.add(Enemy())# 200フレームに1回，敵機を出現させる
291 
292             if boss_life.life != 1:
293                 for b in pg.sprite.groupcollide(boss, beams, False, True).keys():
294                     boss_life.boss_lifes(-1)
295                     exps.add(Explosion(b, 100))
296             else:
297                 for bo in pg.sprite.groupcollide(boss, beams, True, True).keys():
298                     exps.add(Explosion(bo, 100))
299                     boss_life.boss_lifes(-1)
300                     score.score_up(100)
301                     pg.display.update()
302                     time.sleep(1)  # 100点アップ
303                     return
304 
305             screen.blit(bg_img, [0, 0])
306 
307             bird.update(key_lst, screen)
308             beams.update()
309             beams.draw(screen)
310             emys.update()
311             emys.draw(screen)
312             exps.update()
313             exps.draw(screen)
314             boss.update()
315             boss.draw(screen)
316             score.update(screen)
317             boss_life.update(screen)
318             pg.display.update()
319             tmr += 1
320             clock.tick(50)
321 
322 
323     if __name__ == "__main__":
324         pg.init()
325         main()
326         pg.quit()
327         sys.exit()
```