245 lines (212 loc) · 8.57 KB

Code Blame Raw

```python
import math
import random
import sys
import time
from typing import Any

import pygame as pg
from pygame.sprite import AbstractGroup


WIDTH = 1600  # ゲームウィンドウの幅
HEIGHT = 900  # 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 Hero(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
        pg.K_UP: (0, -1),
        pg.K_DOWN: (0, +1),
```

```python
            pg.K_LEFT: (-1, 0),
            pg.K_RIGHT: (+1, 0),
        }

        def __init__(self,xy: tuple[int, int]):
            """
            こうかとん画像Surfaceを生成する
            引数1 num：こうかとん画像ファイル名の番号
            引数2 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/hero.png"), 0, 0.1)
            img = pg.transform.flip(img0, True, False)  # デフォルトのこうかとん
            self.imgs = {
                (+1, 0): img,  # 右
                (+1, -1): pg.transform.rotozoom(img, 45, 1.0),  # 右上
                (0, -1): pg.transform.rotozoom(img, 90, 1.0),  # 上
                (-1, -1): pg.transform.rotozoom(img0, -45, 1.0),  # 左上
                (-1, 0): img0,  # 左
                (-1, +1): pg.transform.rotozoom(img0, 45, 1.0),  # 左下
                (0, +1): pg.transform.rotozoom(img, -90, 1.0),  # 下
                (+1, +1): pg.transform.rotozoom(img, -45, 1.0),  # 右下
            }
            self.dire = (+1, 0)
            self.image = self.imgs[self.dire]
            self.rect = self.image.get_rect()
            self.rect.center = xy
            self.speed = 10

        def change_img(self, num: str, screen: pg.Surface):
            """
            主人公画像を切り替え，画面に転送する
            引数1 num：変更後画像ファイル名
            引数2 screen：画面Surface
            """
            self.image = pg.transform.rotozoom(pg.image.load(f"ex05/fig/{num}.png"), 0, 0.1)
            screen.blit(self.image, self.rect)

        def update(self, key_lst: list[bool], screen: pg.Surface):
            """
            押下キーに応じて主人公を移動させる
            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)
                self.image = self.imgs[self.dire]
            screen.blit(self.image, self.rect)

        def get_direction(self) -> tuple[int, int]:
```

```
105             return self.dire
106
107
108     class Tower(pg.sprite.Sprite):
109         """
110         守るタワーについて設定するクラス
111
112         """
113         def __init__(self) :
114             super().__init__()
115             self.image=pg.transform.rotozoom(pg.image.load(f"ex05/fig/tower.png"), 0, 0.8)
116             self.life = 3
117             self.rect = self.image.get_rect()
118             self.rect.center = WIDTH/2,HEIGHT/2
119
120             self.font = pg.font.Font(None, 50)  #life表示のための用意
121             self.color = (255, 0 , 0)
122             self.displife = self.font.render(f"Life: {self.life}", 0, self.color)
123
124
125         def update(self,screen):
126             """
127             現在のlifeやタワーを表示する
128             """
129             screen.blit(self.image,self.rect)
130             self.displife = self.font.render(f"Life: {self.life}", 0, self.color)
131             screen.blit(self.displife,(30,HEIGHT-125))
132
133     class Enemy(pg.sprite.Sprite):
134         """
135         敵機に関するクラス
136         """
137         imgs = [pg.transform.rotozoom(pg.image.load(f"ex05/fig/ene{i}.png"),0,0.15) for i in range(1, 4)]
138
139         def __init__(self,place: int,tower: Tower):
140             super().__init__()
141             self.image = random.choice(__class__.imgs)
142             self.rect = self.image.get_rect()
143             if place == 0:  #以下のif文で呼び出し時の値によって出現場所を決める
144                 self.rect.center = random.randint(0, WIDTH), 0
145             elif place == 1:
146                 self.rect.center = WIDTH,random.randint(0,HEIGHT)
147             elif place == 2:
148                 self.rect.center = random.randint(0,WIDTH),HEIGHT
149             elif place == 3:
150                 self.rect.center = 0,random.randint(0,HEIGHT)
151             self.vx, self.vy = calc_orientation(self.rect, tower.rect)  #towerに向かうように方向ベクトルの設定
152             self.speed = 6
153
154         def update(self):
155             """
156             敵機を速度ベクトルself.vx,vyに基づき移動させる
157             引数 screen：画面Surface
158             """
159             self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
160
161
162
163     class Score:
164         """
```

```
    打ち落とした敵機の数をスコアとして表示するクラス
    敵機：1点
    """
    def __init__(self):
        self.font = pg.font.Font(None, 50)
        self.color = (0, 0, 0)
        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):
        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)

def main():
    pg.display.set_caption("守れ!こうかとん")
    screen = pg.display.set_mode((WIDTH, HEIGHT))
    bg_img = pg.image.load("ex05/fig/bg_natural_mori.jpg")

    score = Score()
    hero = Hero( (900, 400))
    emys = pg.sprite.Group()
    tower = Tower()
    fonto  = pg.font.Font(None, 50)
    tmr = 0
    clock = pg.time.Clock()

    while True:
        key_lst = pg.key.get_pressed()
        for event in pg.event.get():
            if event.type == pg.QUIT:
                return 0
            if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT:
                if score.score >= 50:
                    tower.life += 1
                    score.score -= 50

        screen.blit(bg_img, [0, 0])

        if tmr%20 == 0:  # 200フレームに1回，敵機を出現させる
            emys.add(Enemy(random.randint(0,3),tower))  #randintで出現位置を四方向から選び、towerで方向の指定
        for emy in pg.sprite.spritecollide(hero, emys, True):
                emy.kill()
                score.score_up(1)

        for emy in pg.sprite.spritecollide(tower, emys, True):
            if tower.life >= 2:
                tower.life -=1  #lifeを減らす
                emy.kill()
            else:
                screen.blit(pg.transform.rotozoom(pg.image.load("ex05/fig/text_gameover.png"),0,0.4),[600,25
                hero.change_img("lose", screen) # 悲しみエフェクト
                tower.life -=1
                score.update(screen)
                tower.update(screen)
                screen.blit(txt_time, [WIDTH/2, 50])
```

```
                    pg.display.update()
                    time.sleep(2)
                    return

            txt_time = fonto.render(str(tmr/50), True, (0, 0 ,0 ))
            screen.blit(txt_time, [WIDTH/2, 50])
            hero.update(key_lst, screen)
            emys.update()
            emys.draw(screen)
            score.update(screen)
            tower.update(screen)
            pg.display.update()
            tmr += 1
            clock.tick(50)


    if __name__ == "__main__":
        pg.init()
        main()
        pg.quit()
        sys.exit()
```