ProjExD_Group03 / musou_kokaton.py

c0b23078ca スコアの合計値による爆弾の変化 5334d42 · 5 minutes ago

521 lines (459 loc) · 18.8 KB

Code Blame Raw

```python
import math
import os
import random
import sys
import time
import pygame as pg


WIDTH = 1100  # ゲームウィンドウの幅
HEIGHT = 650  # ゲームウィンドウの高さ

os.chdir(os.path.dirname(os.path.abspath(__file__)))


def check_bound(obj_rct: pg.Rect) -> tuple[bool, bool]:
    """
    オブジェクトが画面内or画面外を判定し，真理値タプルを返す関数
    引数：こうかとんや爆弾，ビームなどの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_rct.left < 0 or WIDTH < obj_rct.right:
        yoko = False
    if obj_rct.top < 0 or HEIGHT < obj_rct.bottom:
        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
    """
```

```python
    delta = {  # 押下キーと移動量の辞書
        pg.K_UP: (0, -1),
        pg.K_DOWN: (0, +1),
        pg.K_LEFT: (-1, 0),
        pg.K_RIGHT: (+1, 0),
    }

    def __init__(self, num: int, 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"fig/3.png"), 0, 1.5)
        img = pg.transform.flip(img0, True, False)  # デフォルトのこうかとん
        simg0 = pg.transform.rotozoom(pg.image.load(f"fig/0.png"), 0, 1.5)
        simg = pg.transform.flip(simg0, True, False)
        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.simgs = {
            (+1, 0): simg,  # 右
            (+1, -1): simg,  # 右上
            (0, -1): simg,  # 上
            (-1, -1): simg0,  # 左上
            (-1, 0): simg0,  # 左
            (-1, +1): simg0,  # 左下
            (0, +1): simg0,  # 下
            (+1, +1): simg,  # 右下
        }
        self.dire = (+1, 0)
        self.image = self.imgs[self.dire]
        self.rect = self.image.get_rect()
        self.rect.center = xy
        self.simage = self.simgs[self.dire]
        self.srect = self.simage.get_rect()
        self.srect.center = self.rect.center
        self.speed = 10
        self.mode = 0

    def change_img(self, num: int, screen: pg.Surface):
        """
        こうかとん画像を切り替え，画面に転送する
        引数1 num：こうかとん画像ファイル名の番号
        引数2 screen：画面Surface
        """
        self.image = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 1.5)
        screen.blit(self.image, self.rect)

    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
```

```
105             引数1 key_lst：押下キーの真理値リスト
106             引数2 screen：画面Surface
107             """
108             sum_mv = [0, 0]
109             for k, mv in __class__.delta.items():
110                 if key_lst[k]:
111                     sum_mv[0] += mv[0]
112                     sum_mv[1] += mv[1]
113             self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
114             if check_bound(self.rect) != (True, True):
115                 self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_mv[1])
116             if not (sum_mv[0] == 0 and sum_mv[1] == 0):
117                 self.dire = tuple(sum_mv)
118                 self.image = self.imgs[self.dire]
119             screen.blit(self.image, self.rect)
120             if self.mode == 1:
121                 self.srect.center = self.rect.center
122                 if not (sum_mv[0] == 0 and sum_mv[1] == 0):
123                     self.simage = self.simgs[self.dire]
124                 screen.blit(self.simage, self.srect)
125 
126 
127 class Bomb(pg.sprite.Sprite):
128     """
129     爆弾に関するクラス
130     """
131     colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
132 
133     def __init__(self, emy: "Enemy", bird: Bird,tscore):
134         """
135         爆弾円Surfaceを生成する
136         引数1 emy：爆弾を投下する敵機
137         引数2 bird：攻撃対象のこうかとん
138         """
139         super().__init__()
140         if tscore <= 50 :  # スコアの合計が50点以下
141             rad = random.randint(10, 10)  # 爆弾円の半径：10の乱数
142             self.speed = 2  # 爆弾の投下速度2
143         elif 51 <= tscore <= 100 :  # スコアの合計が51点以上100点以下
144             rad = random.randint(10, 20)  # 爆弾円の半径：10以上20以下の乱数
145             self.speed = 3  # 爆弾の投下速度3
146         elif 101 <= tscore <= 150:  # スコアの合計が101点以上150点以下
147             rad = random.randint(10, 30)  # 爆弾円の半径：10以上30以下の乱数
148             self.speed = 4  # 爆弾の投下速度4
149         elif 151 <= tscore <= 200:  # スコアの合計が151点以上200点以下
150             rad = random.randint(10, 40)  # 爆弾円の半径：10以上40以下の乱数
151             self.speed = 5  # 爆弾の投下速度5
152         elif 201 <= tscore <= 250:  # スコアの合計が201点以上250点以下
153             rad = random.randint(10, 50)  # 爆弾円の半径：10以上50以下の乱数
154             self.speed = 6  # 爆弾の投下速度6
155         self.image = pg.Surface((2*rad, 2*rad))
156         color = random.choice(__class__.colors)  # 爆弾円の色：クラス変数からランダム選択
157         pg.draw.circle(self.image, color, (rad, rad), rad)
158         self.image.set_colorkey((0, 0, 0))
159         self.rect = self.image.get_rect()
160         # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
161         self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
162         self.rect.centerx = emy.rect.centerx
163         self.rect.centery = emy.rect.centery+emy.rect.height//2
164 
```

```python
    def update(self):
        """
        爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
        引数 screen：画面Surface
        """
        self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
        if check_bound(self.rect) != (True, True):
            self.kill()


class Beam(pg.sprite.Sprite):
    """
    ビームに関するクラス
    """
    def __init__(self, bird: Bird, angle0=0, mode=0):
        """
        ビーム画像Surfaceを生成する
        引数 bird：ビームを放つこうかとん
        """
        super().__init__()
        self.vx, self.vy = bird.dire
        angle = math.degrees(math.atan2(-self.vy, self.vx))
        angle+=angle0
        if mode == 0:
            self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 1.0)
        elif mode == 1:
            self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 1.0)
        self.vx = math.cos(math.radians(angle))
        self.vy = -math.sin(math.radians(angle))
        self.rect = self.image.get_rect()
        self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
        self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
        self.speed = 15
        self.count = 0

    def update(self):
        """
        ビームを速度ベクトルself.vx, self.vyに基づき移動させる
        引数 screen：画面Surface
        """
        self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
        if check_bound(self.rect) != (True, True):
            self.count += 1
            if self.count == 15:
                self.kill()


class Explosion(pg.sprite.Sprite):
    """
    爆発に関するクラス
    """
    def __init__(self, obj: "Bomb|Enemy", life: int):
        """
        爆弾が爆発するエフェクトを生成する
        引数1 obj：爆発するBombまたは敵機インスタンス
        引数2 life：爆発時間
        """
        super().__init__()
        img = pg.image.load(f"fig/explosion.gif")
        self.imgs = [img, pg.transform.flip(img, 1, 1)]
```

```
225             self.image = self.imgs[0]
226             self.rect = self.image.get_rect(center=obj.rect.center)
227             self.life = life
228 
229         def update(self):
230             """
231             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
232             爆発エフェクトを表現する
233             """
234             self.life -= 1
235             self.image = self.imgs[self.life//10%2]
236             if self.life < 0:
237                 self.kill()
238 
239 
240     class Enemy(pg.sprite.Sprite):
241         """
242         敵機に関するクラス
243         """
244         imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
245 
246         def __init__(self):
247             super().__init__()
248             self.image = random.choice(__class__.imgs)
249             self.rect = self.image.get_rect()
250             self.rect.center = random.randint(0, WIDTH), 0
251             self.vx, self.vy = 0, +6
252             self.bound = random.randint(50, HEIGHT//2)  # 停止位置
253             self.state = "down"  # 降下状態or停止状態
254             self.interval = random.randint(50, 300)  # 爆弾投下インターバル
255 
256         def update(self):
257             """
258             敵機を速度ベクトルself.vyに基づき移動（降下）させる
259             ランダ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
260             引数 screen：画面Surface
261             """
262             if self.rect.centery > self.bound:
263                 self.vy = 0
264                 self.state = "stop"
265             self.rect.move_ip(self.vx, self.vy)
266 
267 
268     class NeoBeam(pg.sprite.Sprite):
269 
270         def __init__(self,bird:Bird,num):
271             super().__init__()
272             self.bird=bird
273             self.num=num
274 
275         def gen_beams(self):
276             return [Beam(self.bird,angle0,1)for angle0 in range(-50,+51,100//(self.num-1))]
277 
278 
279     class Score:
280         """
281         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
282         爆弾：1点
283         敵機：10点
284         """
```

```python
    def __init__(self):
        self.font = pg.font.Font(None, 50)
        self.color = (250, 20, 20)
        self.value = 0
        self.image = self.font.render(f"Score: {self.value}", 0, self.color)
        self.rect = self.image.get_rect()
        self.rect.center = 100, HEIGHT-50

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

class Bullet(pg.sprite.Sprite):
    """
    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,bullet :pg.Surface,step=1,value=1,mvalue=1,mct=0,ns=0):
        self.font = pg.font.Font(None, 60)
        self.color = (10, 10, 10)
        self.mct = mct
        self.ct = 0
        self.step = step
        self.ns = ns
        self.mode = 0

        self.value = value
        self.mvalue = mvalue
        self.bimg = bullet
        self.back = pg.Surface((240,60))
        pg.draw.rect(self.back,(192,192,192), (0,0,140,60))
        pg.draw.rect(self.back,(100,255,100), (140,0,240,60))
        self.back.set_alpha(128)
        self.image = self.font.render(f"{self.value}", 0, self.color)
        self.brct = self.bimg.get_rect()
        self.bact = self.back.get_rect()
        self.rect = self.image.get_rect()
        self.brct.center = WIDTH-174, HEIGHT-(60*step) + 30
        self.bact.center = WIDTH-120, HEIGHT-(60*step) + 30
        self.rect.center = WIDTH-75, HEIGHT-(60*step) + 30

    def update(self,screen: pg.Surface,score):
        if self.value == 0:
            self.color = (255, 10, 10)
            if self.ct > 0:
                self.ct -= 1
        else:
            self.color = (10, 10, 10)
        if self.ct == 0 and self.ns <= score:
            self.value = self.mvalue
        if self.mode == 1:
            self.value -= 1
            if self.value == 0:
                self.mode = 0
        self.image = self.font.render(f"{self.value}", 0, self.color)
        screen.blit(self.back,self.bact)
        screen.blit(self.bimg,self.brct)
        screen.blit(self.image,self.rect)

```

```python

class Shield(pg.sprite.Sprite):
    """
    敵の攻撃を防ぐシールドを展開する
    """
    def __init__(self, bird: Bird ,life:int):
        """
        シールドの基本情報
        """
        super().__init__()
        bird.mode = 1
        self.vx ,self.vy = bird.dire
        img = pg.Surface((bird.rect.height*1.8,bird.rect.height*1.8))
        pg.draw.circle(img,(50,255,50),(bird.rect.height*0.9,bird.rect.height*0.9),bird.rect.height*0.9)
        self.image = img
        self.image.set_alpha(100)
        self.image.set_colorkey((0, 0, 0))
        self.rect = self.image.get_rect()
        self.rect.centery = bird.rect.centery
        self.rect.centerx = bird.rect.centerx
        self.life = life

    def update(self ,bird: Bird ,key_lst: list[bool]):
        """
        shieldをこうかとんの位置に基づき移動させる
        引数 screen：画面Surface
        """
        self.rect.center = bird.rect.center
        self.life -= 1
        if self.life == 0:
            bird.mode = 0
            self.kill()


class Gravity(pg.sprite.Sprite):
    """
    重力場を発生させるクラス
    """
    def __init__(self):
        super().__init__()
        self.image = pg.Surface((WIDTH, HEIGHT))
        self.life = 400
        pg.draw.rect(self.image,(100,255,100), (0,0,WIDTH,HEIGHT))
        self.image.set_alpha(128)
        self.rect = self.image.get_rect()
        self.rect.center = WIDTH//2, HEIGHT//2

    def update(self):
        self.life -= 1
        if self.life < 0:
            self.kill()



def main():
    pg.display.set_caption("真！こうかとん無双")
    screen = pg.display.set_mode((WIDTH, HEIGHT))
    bg_img = pg.image.load(f"fig/pg_bg.jpg")
    score = Score()
```

```python
    tscore = 0

    bu_beam = Bullet(pg.transform.rotozoom(pg.image.load(f"fig/beam.png"),0,0.5),4,5,5,100,0)
    bu_unig = Bullet(pg.transform.rotozoom(pg.image.load(f"fig/beam.png"),0,0.5),3,1,1,200,0)
    bu_psyc = Bullet(pg.transform.rotozoom(pg.image.load(f"fig/0.png"),0,1),2,0,400,800,50)
    bu_grav = Bullet(pg.transform.rotozoom(pg.image.load(f"fig/6.png"),0,1),1,0,400,1200,200)

    bird = Bird(3, (550, 600))
    bombs = pg.sprite.Group()
    beams = pg.sprite.Group()
    exps = pg.sprite.Group()
    emys = pg.sprite.Group()
    shields = pg.sprite.Group()
    gravitys = pg.sprite.Group()
    bullets = [bu_beam,bu_unig,bu_psyc,bu_grav]

    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_c and len(shields) == 0 and score.value >= 5
                if bu_psyc.ct <= 0:
                    shields.add(Shield(bird,400))
                    bu_psyc.mode = 1
                    bu_psyc.ct = bu_psyc.mct
                    score.value -= 50
            if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
                if key_lst[pg.K_LSHIFT]:
                    if bu_unig.ct <= 0:
                        m_beam=NeoBeam(bird,3)
                        beams.add(m_beam.gen_beams())
                        bu_unig.value -= 1
                        bu_unig.ct = bu_unig.mct
                else:
                    if -5 < bu_beam.ct <= 0:
                        beams.add(Beam(bird))
                        bu_beam.value -= 1
                        bu_beam.ct -= 1
                    if bu_beam.ct <= (-5):
                        bu_beam.ct = bu_beam.mct
            if event.type == pg.KEYDOWN and event.key == pg.K_RETURN and len(gravitys) == 0 and score.valu
                if bu_grav.ct <= 0:
                    gravitys.add(Gravity())
                    bird.change_img(6, screen)
                    bu_grav.mode = 1
                    bu_grav.ct = bu_grav.mct
                    score.value -= 200
        screen.blit(bg_img, [0, 0])

        if tmr%25 == 0:  # 200フレームに1回，敵機を出現させる
            emys.add(Enemy())

        for emy in emys:
            if emy.state == "stop" and tmr%emy.interval == 0:
                # 敵機が停止状態に入ったら，intervalに応じて爆弾投下
                bombs.add(Bomb(emy, bird,score.value))

```

```
            for emy in pg.sprite.groupcollide(emys, beams, True, False).keys():
                exps.add(Explosion(emy, 100))  # 爆発エフェクト
                score.value += 10  # 10点アップ
                tscore += 10  # 10点加算

            if bird.mode == 1 and score.value >= 1000:
                for emy in pg.sprite.groupcollide(emys, shields, True, False).keys():
                    exps.add(Explosion(emy, 100))

            for bomb in pg.sprite.groupcollide(bombs, beams, True, False).keys():
                exps.add(Explosion(bomb, 50))  # 爆発エフェクト
                score.value += 1  # 1点アップ
                tscore += 1  # 1点加算

            for bomb in pg.sprite.groupcollide(bombs, shields, True, False).keys():
                exps.add(Explosion(bomb, 50))  # 爆発エフェクト

            if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
                bird.change_img(8, screen) # こうかとん悲しみエフェクト
                score.update(screen)
                pg.display.update()
                time.sleep(2)
                return

            for emy in pg.sprite.groupcollide(emys, gravitys, True, False).keys(): # 敵との衝突判定
                exps.add(Explosion(emy, 50))  # 爆発エフェクト

            for bomb in pg.sprite.groupcollide(bombs, gravitys, True, False).keys(): # 爆弾との衝突判定
                exps.add(Explosion(bomb, 50))  # 爆発エフェクト


            bird.update(key_lst, screen)
            beams.update()
            beams.draw(screen)
            emys.update()
            emys.draw(screen)
            bombs.update()
            bombs.draw(screen)
            exps.update()
            exps.draw(screen)
            score.update(screen)
            shields.update(bird,key_lst)
            shields.draw(screen)
            gravitys.update()
            gravitys.draw(screen)
            for i in bullets:
                i.update(screen,score.value)
            pg.display.update()
            tmr += 1
            clock.tick(50)


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