486 lines (423 loc) · 17.4 KB


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

import pygame as pg


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 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
        pg.K_UP: (0, -1),
```

```python
            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"ex04/fig/{num}.png"), 0, 2.0)
        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
        self.state = "normal"

    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"ex04/fig/{num}.png"), 0, 2.0)
        screen.blit(self.image, self.rect)

    def change_state(self, state: str, hyper_life: int):
        self.state = state
        self.hyper_life = hyper_life



    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)
                self.image = self.imgs[self.dire]

            if self.state == "hyper":
                self.image = pg.transform.laplacian(self.image)
                self.hyper_life -= 1
                if self.hyper_life < 0:
                    self.change_state("normal",-1)
            screen.blit(self.image, self.rect)

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




class Bomb(pg.sprite.Sprite):
    """
    爆弾に関するクラス
    """
    colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]

    def __init__(self, emy: "Enemy", bird: Bird):
        """
        爆弾円Surfaceを生成する
        引数1 emy：爆弾を投下する敵機
        引数2 bird：攻撃対象のこうかとん
        """
        super().__init__()
        rad = random.randint(10, 50)  # 爆弾円の半径：10以上50以下の乱数
        color = random.choice(__class__.colors)  # 爆弾円の色：クラス変数からランダム選択
        self.image = pg.Surface((2*rad, 2*rad))
        pg.draw.circle(self.image, color, (rad, rad), rad)
        self.image.set_colorkey((0, 0, 0))
        self.rect = self.image.get_rect()
        # 爆弾を投下するemyから見た攻撃対象のbirdの方向を計算
        self.vx, self.vy = calc_orientation(emy.rect, bird.rect)
        self.rect.centerx = emy.rect.centerx
        self.rect.centery = emy.rect.centery+emy.rect.height/2
        self.speed = 6

    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, spin=0):   #spinの初期値は０
        """
```

```
165             ビーム画像Surfaceを生成する
166             引数 bird：ビームを放つこうかとん
167             """
168             super().__init__()
169             self.vx, self.vy = bird.get_direction()
170             angle = math.degrees(math.atan2(-self.vy, self.vx))
171             angle += spin #angleにspinを加える
172             self.image = pg.transform.rotozoom(pg.image.load(f"ex04/fig/beam.png"), angle, 2.0)
173             self.vx = math.cos(math.radians(angle))
174             self.vy = -math.sin(math.radians(angle))
175             self.rect = self.image.get_rect()
176             self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
177             self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
178             self.speed = 10
179
180         def update(self):
181             """
182             ビームを速度ベクトルself.vx, self.vyに基づき移動させる
183             引数 screen：画面Surface
184             """
185             self.rect.move_ip(+self.speed*self.vx, +self.speed*self.vy)
186             if check_bound(self.rect) != (True, True):
187                 self.kill()
188
189
190     class NeoBeam: #追加機能４弾幕
191         def __init__(self, bird:Bird, num:int):
192             self.bird = bird
193             self.num = num
194
195         def gen_beams(self): #こうかとんに対し-50°~50°の範囲にbeamを発生させる
196             beam_ls = []
197             for spin in range(-50, 51, 25):
198                 beam = Beam(self.bird, spin)
199                 beam_ls.append(beam)
200             return beam_ls
201
202     class Explosion(pg.sprite.Sprite):
203         """
204         爆発に関するクラス
205         """
206         def __init__(self, obj: "Bomb|Enemy", life: int):
207             """
208             爆弾が爆発するエフェクトを生成する
209             引数1 obj：爆発するBombまたは敵機インスタンス
210             引数2 life：爆発時間
211             """
212             super().__init__()
213             img = pg.image.load("ex04/fig/explosion.gif")
214             self.imgs = [img, pg.transform.flip(img, 1, 1)]
215             self.image = self.imgs[0]
216             self.rect = self.image.get_rect(center=obj.rect.center)
217             self.life = life
218
219         def update(self):
220             """
221             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
222             爆発エフェクトを表現する
223             """
224             self.life -= 1
```

```python
            self.image = self.imgs[self.life//10%2]
            if self.life < 0:
                self.kill()


    class Shield(pg.sprite.Sprite):
        def __init__(self,bird: Bird,life : int):
             super().__init__()
             self.image = pg.Surface((20,bird.rect.height*2))
             pg.draw.rect(self.image,(0,0,0),pg.Rect(0,0, 20, bird.rect.height*2))
             self.rect = self.image.get_rect()
             self.rect.centerx = bird.rect.centerx+50
             self.rect.centery = bird.rect.centery
             self.life = life

        def update(self):
            self.life -= 1
            if self.life < 0:
                self.kill() #Shiledsグループからの削除


    class Enemy(pg.sprite.Sprite):
        """
        敵機に関するクラス
        """
        imgs = [pg.image.load(f"ex04/fig/alien{i}.png") for i in range(1, 4)]

        def __init__(self):
            super().__init__()
            self.image = random.choice(__class__.imgs)
            self.rect = self.image.get_rect()
            self.rect.center = random.randint(0, WIDTH), 0
            self.vy = +6
            self.bound = random.randint(50, HEIGHT/2)  # 停止位置
            self.state = "down"  # 降下状態or停止状態
            self.interval = random.randint(50, 300)  # 爆弾投下インターバル

        def update(self):
            """
            敵機を速度ベクトルself.vyに基づき移動（降下）させる
            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
            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 = 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)
    class Reload:
        """
        時間を計測するして、表示するクラス
        """
        def __init__(self, start,fr):
            """
            start = 初期値
            fr = フレームレート
            """
            self.font = pg.font.Font(None, 50)
            self.color = (0, 0, 255)
            self.start = start//fr
            self.image = self.font.render(f"Reloadtime: {self.start}", 0, self.color)
            self.rect = self.image.get_rect()
            self.rect.center = 125, HEIGHT-25

        def time_up(self, add):
            self.start += add

        def update(self, screen: pg.Surface):
            self.image = self.font.render(f"Reloadtime: {self.start}", 0, self.color)
            screen.blit(self.image, self.rect)
    class Gravity(pg.sprite.Sprite):
        """
        重力球の追加
        """
        def __init__(self, bird, size, life):
            super().__init__()
            #self.rad =  size # ジュウリョクダマの半径：size
            color = (1, 1, 1)  # 重力タマの色：黒
            self.image = pg.Surface((2*size, 2*size))
            pg.draw.circle(self.image, color, (size, size), size)
            self.image.set_colorkey((0, 0, 0))
            self.image.set_alpha(200)
            self.rect = self.image.get_rect()
            self.rect.centerx = bird.rect.centerx
            self.rect.centery = bird.rect.centery
            self.life = life
            #self.speed = bird.speed
        def update(self, ):#key_lst)これを追加すれば、球がついてくる機能を有効化できる。:
            """
            以下は球がついてくるようになる追加機能である。
            sum_mv = [0, 0]
            for k, mv in Bird.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 Bird.delta.items():
                    if key_lst[k]:
                        self.rect.move_ip(-self.speed*mv[0], -self.speed*mv[1])
```

```
            """
            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("ex04/fig/pg_bg.jpg")
    score = Score()

    bird = Bird(3, (900, 400))
    bombs = pg.sprite.Group()
    beams = pg.sprite.Group()
    exps = pg.sprite.Group()
    emys = pg.sprite.Group()
    Shields = pg.sprite.Group()

    gravity = pg.sprite.Group()
    tmr = 0
    clock = pg.time.Clock()
    re = 0
    count = 0
    re_time = False
    while True:
        key_lst = pg.key.get_pressed()
        shift_pressed = False
        for event in pg.event.get():
            if event.type == pg.QUIT:
                return 0
            if event.type == pg.KEYDOWN and event.key == pg.K_SPACE and (re ==0 or tmr/50>re+5) :#ビームを５回
                beams.add(Beam(bird))
                count += 1#出した数ビームの数
                if count >= 5:#出したビームの数が５いじょうなら
                    re = tmr/50#時間を記録
                    re_time = Reload(re-tmr//50, 50)#Reloadクラスのインスタンス作成
                    count = 0#出したビームの数を０にする
                if pg.key.get_mods() & pg.KMOD_LSHIFT :
                    count += 5#出した数ビームの数
                    if count >= 5:
                        count = 0
                        re = tmr/50
                        re_time = Reload(re-tmr//50, 50)
                    shift_pressed = True
            if event.type == pg.KEYDOWN and event.key == pg.K_CAPSLOCK:
                if score.score >= 10 and len(Shields) == 0:
                    Shields.add(Shield(bird,400))
                    score.score -= 50

            if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT and score.score >= 100:
                bird.change_state("hyper",500)
                score.score_up(-100)
            if event.type == pg.KEYDOWN and event.key == pg.K_LSHIFT:
                bird.speed = 20
            if event.type == pg.KEYUP and event.key == pg.K_LSHIFT:
                bird.speed = 10
            if event.type == pg.KEYDOWN and event.key == pg.K_TAB and score.score >= 50:
                score.score_up(-50)
                gravity.add(Gravity(bird, 200, 500))

```

```
406             screen.blit(bg_img, [0, 0])
407
408             if tmr%200 == 0:  # 200フレームに1回，敵機を出現させる
409                 emys.add(Enemy())
410
411             for emy in emys:
412                 if emy.state == "stop" and tmr%emy.interval == 0:
413                     # 敵機が停止状態に入ったら，intervalに応じて爆弾投下
414                     bombs.add(Bomb(emy, bird))
415
416             for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():
417                 exps.add(Explosion(emy, 100))  # 爆発エフェクト
418                 score.score_up(10)  # 10点アップ
419                 bird.change_img(6, screen)  # こうかとん喜びエフェクト
420
421             for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():
422                 exps.add(Explosion(bomb, 50))  # 爆発エフェクト
423                 score.score_up(1)  # 1点アップ
424
425             for bomb in pg.sprite.spritecollide(bird, bombs, True):
426                 if bird.state == "hyper":
427                     exps.add(Explosion(bomb, 50))  # 爆発エフェクト
428                     score.score_up(1)  # 1点アップ
429
430                 else:
431                    bird.change_img(8, screen) # こうかとん悲しみエフェクト
432                    score.update(screen)
433                    pg.display.update()
434                    time.sleep(2)
435                    return
436
437             for bomb in pg.sprite.groupcollide(bombs,gravity, True, False).keys():
438                 exps.add(Explosion(bomb, 50))
439
440             if len(pg.sprite.spritecollide(bird, bombs, True)) != 0:
441                 bird.change_img(8, screen) # こうかとん悲しみエフェクト
442                 score.update(screen)
443                 pg.display.update()
444                 time.sleep(2)
445                 return
446
447             for bomb in pg.sprite.groupcollide(bombs, Shields, True, False).keys():
448                 exps.add(Explosion(bomb, 50))  # 爆発エフェクト
449                 score.score_up(1)
450
451             gravity.update()#key_lst)これを有効化すると、球がついてくる。
452             gravity.draw(screen)
453             if shift_pressed: #左shiftおされたら
454                 if pg.key.get_mods() & pg.KMOD_LSHIFT:
455                     num_beams = 5
456                     neo_beam = NeoBeam(bird, num_beams)
457                     beams.add(*neo_beam.gen_beams())
458
459             bird.update(key_lst, screen)
460             beams.update()
461             beams.draw(screen)
462             emys.update()
463             emys.draw(screen)
464             bombs.update()
465             bombs.draw(screen)
```

```
466             exps.update()
467             exps.draw(screen)
468             Shields.update() #防御壁の更新
469             Shields.draw(screen) #防御壁の描画
470             if re_time:
471                 if tmr % 50 == 0:
472                     re_time.time_up(1)
473                 if re_time.start <= 5:
474                     re_time.update(screen)
475 
476 
477             score.update(screen)
478             pg.display.update()
479             tmr += 1
480             clock.tick(50)
481 
482     if __name__ == "__main__":
483         pg.init()
484         main()
485         pg.quit()
486         sys.exit()
```