c0a2407017 / ProjExD_4

Code Issues Pull requests Actions Projects Security

Commits

main All users All time

Commits on May 13, 2025

Merge branch 'main' of https://github.com/c0a2407017/ProjExD_4
a5df568
c0a24017 committed 2 days ago

最終修正
d17fa0f
c0a24017 committed 2 days ago

Merge pull request #5 from TOKYOLIUNIAN/main
Verified 37fad87
c0a2407017 authored 2 days ago

feature3
8d20d9d
TOKYO_LIUNIAN committed 2 days ago

feature3
6a45937
TOKYO_LIUNIAN committed 2 days ago

test
4dced30
TOKYO_LIUNIAN committed 2 days ago

test
cceb9f0
c0a24017 committed 2 days ago

test]
1ae5c2d
TOKYO_LIUNIAN committed 2 days ago

追加３]
45c06a5
TOKYO_LIUNIAN committed 2 days ago

Merge branch 'c0a2407017:main' into main

Verified 1670c3f

TOKYOLIUNIAN authored 2 days ago

Merge pull request #4 from c0a241589e/main

Verified 5cfc333

c0a2407017 authored 2 days ago

五人目

bdd3a47

c0a241589e committed 2 days ago

追加機能6

0d38012

c0a241589e committed 2 days ago

Merge pull request #3 from c0c24020ae/main

Verified c68ada0

c0a2407017 authored 2 days ago

Merge branch 'C0C24020/feature1'

c5fff01

c0c24020ae committed 2 days ago

三人目

dfee162

TOKYO_LIUNIAN committed 2 days ago

三人目

e9e8d5d

TOKYO_LIUNIAN committed 2 days ago

Merge pull request #2 from c0a241589e/main

Verified 8c099b9

c0a2407017 authored 2 days ago

二人目

2af33e7

c0a241589e committed 2 days ago

追加機能１実装完了

e35daae

c0c24020ae committed 2 days ago

featuer3 の保存

28b117d

TOKYO_LIUNIAN committed 2 days ago

Merge pull request #1 from c0a2419825/main

Verified 9e44f36

c0a2407017 authored 2 days ago

merge一人目

d9fc7a9

c0a24198 committed 2 days ago

追加機能6：弾幕

c93e1cd

c0a241589e committed 2 days ago

追加機能6：弾幕

727c7ef

c0a241589e committed 2 days ago

追加機能2：重力場

dd84f1e

c0a241589e committed 2 days ago

追加機能6：弾幕

02e2b0a

c0a241589e committed 2 days ago

追加機能4実装完了

99988af

c0a24198 committed 2 days ago

追加機能 3 EMP実装完了

2d5a274

TOKYO_LIUNIAN committed 2 days ago

追加機能2 : 重力場

6e1c2a9

c0a241589e committed 2 days ago

追加機能5

b808fe0

c0a24017 committed 2 days ago

初期状態

Verified e3a6fbc

c0a2407017 authored 2 days ago

501 lines (427 loc) · 18.3 KB


```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/{num}.png"), 0, 0.9)
        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, 0.9),  # 右上
            (0, -1): pg.transform.rotozoom(img, 90, 0.9),  # 上
            (-1, -1): pg.transform.rotozoom(img0, -45, 0.9),  # 左上
            (-1, 0): img0,  # 左
            (-1, +1): pg.transform.rotozoom(img0, 45, 0.9),  # 左下
            (0, +1): pg.transform.rotozoom(img, -90, 0.9),  # 下
            (+1, +1): pg.transform.rotozoom(img, -45, 0.9),  # 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"
        self.hyper_life = 500


    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, 0.9)
        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
        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]
        speed = 20 if key_lst[pg.K_LSHIFT] else 10 #左Shiftキーで加速 李
        for k, mv in __class__.delta.items():
            if key_lst[k]:
                sum_mv[0] += mv[0]
                sum_mv[1] += mv[1]
```

```
            self.rect.move_ip(self.speed*sum_mv[0], self.speed*sum_mv[1])
            if check_bound(self.rect) != (True, True):
                self.rect.move_ip(-self.speed*sum_mv[0], -self.speed*sum_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":
                if self.hyper_life<0:
                    self.state="normal"
                self.hyper_life-=1
                self.image = pg.transform.laplacian(self.image)

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

            self.speed = 14 if key_lst[pg.K_LSHIFT] else 7


    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__()
            self.active = True #后加的添加状态标记
            #后加的↓
            def update(self):
                if self.active:
                    self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
                    if check_bound(self.rect) != (True, True):
                        self.kill()
            #后加的↑


            rad = random.randint(10, 50)  # 爆弾円の半径：10以上50以下の乱数
            self.image = pg.Surface((2*rad, 2*rad))
            color = random.choice(__class__.colors)  # 爆弾円の色：クラス変数からランダム選択
            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
            """
```

```
159             self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
160             if check_bound(self.rect) != (True, True):
161                 self.kill()
162
163
164 class Beam(pg.sprite.Sprite):
165     """
166     ビームに関するクラス
167     """
168     def __init__(self, bird: Bird, angle_offset: int = 0):
169         """
170         ビーム画像Surfaceを生成する
171         引数 bird：ビームを放つこうかとん
172         引数 angle_offset：ビームの角度オフセット
173         """
174         super().__init__()
175         self.vx, self.vy = bird.dire
176         angle = math.degrees(math.atan2(-self.vy, self.vx))
177         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle , 1.0)
178         base_angle = math.degrees(math.atan2(-self.vy, self.vx))
179         angle = base_angle + angle_offset  # 角度オフセットを追加
180         self.image = pg.transform.rotozoom(pg.image.load(f"fig/beam.png"), angle, 1.0)
181         self.vx = math.cos(math.radians(angle))
182         self.vy = -math.sin(math.radians(angle))
183         self.rect = self.image.get_rect()
184         self.rect.centery = bird.rect.centery+bird.rect.height*self.vy
185         self.rect.centerx = bird.rect.centerx+bird.rect.width*self.vx
186         self.rect.centery = bird.rect.centery + bird.rect.height * self.vy
187         self.rect.centerx = bird.rect.centerx + bird.rect.width * self.vx
188         self.speed = 10
189
190     def update(self):
191         """
192         ビームを速度ベクトルself.vx, self.vyに基づき移動させる
193         引数 screen：画面Surface
194         """
195         self.rect.move_ip(self.speed*self.vx, self.speed*self.vy)
196         if check_bound(self.rect) != (True, True):
197             self.kill()
198
199
200 class NeoBeam(pg.sprite.Sprite):
201     """
202     ネオビームに関するクラス
203     """
204     def __init__(self, bird: Bird, num : int):
205         super().__init__()
206         self.bird = bird
207         self.num = num
208
209     def gen_beams(self):
210         beam_ls = []
211         for i in range(-50, +51, int  (100 / (self.num - 1))) :
212             beam_ls += [i]
213         return beam_ls
214
215
216 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)]
            self.image = self.imgs[0]
            self.rect = self.image.get_rect(center=obj.rect.center)
            self.life = life

        def update(self):
            """
            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
            爆発エフェクトを表現する
            """
            self.life -= 1
            self.image = self.imgs[self.life//10%2]
            if self.life < 0:
                self.kill()

    class Gravity(pg.sprite.Sprite):
        """
        重力に関するクラス
        """
        def __init__(self, life: int):
            super().__init__()
            self.image = pg.Surface((WIDTH, HEIGHT))
            self.rect = self.image.get_rect()
            self.rect.center = WIDTH//2, HEIGHT//2
            pg.draw.rect(self.image, (0, 0, 0), (0, 0, WIDTH, HEIGHT))
            self.image.set_alpha(128)
            self.life = life

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



    # 追加機能３：電磁パルス（EMP）李
    class EMP(pg.sprite.Sprite):
        def __init__(self, emys: pg.sprite.Group, bombs:pg.sprite.Group,screen : pg.Surface):
            super().__init__()
            self.emys = emys
            self.bombs = bombs
            self.screen = screen
            self.life = 30 #0.5秒

            #创建黄色半透明矩形
            self.image = pg.Surface((WIDTH, HEIGHT))
            self.image.fill((255, 255, 0))#黄色
```

```
275             self.image.set_alpha(128)#半透明
276             self.rect = self.image.get_rect()
277             # 立即调用无效化方法
278             self.deactivate_enemies()
279             self.deactivate_bombs()
280
281
282         def deactivate_enemies(self):
283             # Enemyインスタンスを無効化する, Bombインスタンスを無効化する
284             #敌机无效化,禁止投弹
285             for emy in self.emys:
286                 emy.interval = float("inf")  #停止投弹
287                 emy.image = pg.transform.laplacian(emy.image)#滤镜效果
288                 emy.disable = True #敌机状态为无效化
289                 emy.speed = 0 #敌机速度为0
290                 emy.state = "disable" #敌机状态为无效化
291         def deactivate_bombs(self):
292             #炸弹无效化
293             for bomb in self.bombs:
294                 bomb.speed = 0 #完全停止
295                 bomb.active = False #炸弹状态为无效化
296
297         def update(self):
298             #EMP持续时间
299             self.life -= 1
300             if self.life < 0:
301                 self.kill() #持续时间结合后删除EMP效果
302
303     #以上是EMP的实现
304
305
306
307 class Enemy(pg.sprite.Sprite):
308     """
309     敵機に関するクラス
310     """
311     imgs = [pg.image.load(f"fig/alien{i}.png") for i in range(1, 4)]
312
313     def __init__(self):
314         super().__init__()
315         self.disable = False #后加的
316         self.image = pg.transform.rotozoom(random.choice(__class__.imgs), 0, 0.8)
317         self.rect = self.image.get_rect()
318         self.rect.center = random.randint(0, WIDTH), 0
319         self.vx, self.vy = 0, +6
320         self.bound = random.randint(50, HEIGHT//2)  # 停止位置
321         self.state = "down"  # 降下状態or停止状態
322         self.interval = random.randint(50, 300)  # 爆弾投下インターバル
323
324     def update(self):
325         """
326         敵機を速度ベクトルself.vyに基づき移動（降下）させる
327         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
328         引数 screen：画面Surface
329         """
330         if not self.disable: #后加的
331             if self.rect.centery > self.bound:
332                 self.vy = 0
```

```
                self.state = "stop"
            self.rect.move_ip(self.vx, 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.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 Shield(pg.sprite.Sprite):
    """
    こうかとんの防御シールド
    """
    def __init__(self, bird: Bird, life: int):
        super().__init__()
        self.bird = bird
        self.life = life

        height = self.bird.rect.height * 2
        self.original_image = pg.Surface((20, height), flags=pg.SRCALPHA)
        pg.draw.rect(self.original_image, (0, 0, 255), (0, 0, 20, height))

        self.image = self.original_image
        self.rect = self.image.get_rect(center=self.bird.rect.center)


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

        vx, vy = self.bird.dire
        angle = math.degrees(math.atan2(-vy, vx))

        self.image = pg.transform.rotozoom(self.original_image, angle, 0.9)
        self.rect = self.image.get_rect(center=self.bird.rect.center)

        self.rect.centerx += vx * self.bird.rect.width
        self.rect.centery += vy * self.bird.rect.height


def main():
    pg.display.set_caption("真！こうかとん無双")
```

```python
    screen = pg.display.set_mode((WIDTH, HEIGHT))
    bg_img = pg.image.load(f"fig/pg_bg.jpg")
    score = Score()
    emps = pg.sprite.Group()  # EMP后加的
    bird = Bird(3, (900, 400))
    bombs = pg.sprite.Group()
    beams = pg.sprite.Group()
    exps = pg.sprite.Group()
    emys = pg.sprite.Group()
    shield = pg.sprite.Group()
    gravity = pg.sprite.Group()

    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_SPACE:
                beams.add(Beam(bird))
            if event.type == pg.KEYDOWN and event.key == pg.K_s and score.value >= 50 and len(shield)
                shield.add(Shield(bird, 400))  # 防御壁を発動（400フレーム）
                score.value -= 50  # スコアを50消費
            if event.type == pg.KEYDOWN and event.key == pg.K_RSHIFT:
                if score.value>100:
                    score.value-=100
                    bird.state="hyper"
                    bird.hyper_life=500

            if event.type == pg.KEYDOWN and event.key == pg.K_RETURN:
                if score.value >= 200:
                    score.value -=200
                    gravity.add(Gravity(400))
            if event.type == pg.KEYDOWN and key_lst[pg.K_LSHIFT] and key_lst[pg.K_SPACE]:
                neobeam = NeoBeam(bird, 5)
                neobeam_ls = neobeam.gen_beams()
                for i in neobeam_ls:
                    beams.add(Beam(bird, i))

            elif event.type == pg.KEYDOWN and event.key ==pg.K_e and score.value >=20:# EMP判断是否可以
                    emps.add(EMP(emys,bombs,screen))
                    score.value -= 20    #消耗分数
        screen.blit(bg_img, [0, 0])

        if tmr%200 == 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))

        for emy in pg.sprite.groupcollide(emys, beams, True, True).keys():  # ビームと衝突した敵機リスト
            exps.add(Explosion(emy, 100))  # 爆発エフェクト
            score.value += 10  # 10点アップ
            bird.change_img(6, screen)  # こうかとん喜びエフェクト

```

```
            for bomb in pg.sprite.groupcollide(bombs, beams, True, True).keys():  # ビームと衝突した爆弾リス
                exps.add(Explosion(bomb, 50))  # 爆発エフェクト
                score.value += 1  # 1点アップ

            for emy in pg.sprite.groupcollide(emys, gravity, True, False).keys():  # 重力と衝突した敵機リスト
                exps.add(Explosion(emy, 100))  # 爆発エフェクト
                score.value += 10  # 10点アップ
                bird.change_img(6, screen)  # こうかとん喜びエフェクト
            for bomb in pg.sprite.groupcollide(bombs, gravity, True, False).keys():  # 重力と衝突した爆弾リス
                exps.add(Explosion(bomb, 50))  # 爆発エフェクト                score.value += 1  # 1点アップ
                bird.change_img(6, screen)  # こうかとん喜びエフェクト

            for bomb in pg.sprite.spritecollide(bird, bombs, True):  # こうかとんと衝突した爆弾リスト
                if bird.state == "hyper":
                    score.value +=1
                    break
                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 shields in shield:
                for bomb in pg.sprite.spritecollide(shields, bombs, True):  # 防御壁と衝突した爆弾を削除
                    exps.add(Explosion(bomb, 50))  # 爆発エフェクト
                    score.value += 1  # スコアを1加算

            bird.update(key_lst, screen)
            beams.update()
            beams.draw(screen)
            emys.update()
            emys.draw(screen)
            emps.update() # EMP后加的
            emps.draw(screen)#绘制EMP效果 ,后加的
            bombs.update()
            bombs.draw(screen)
            exps.update()
            exps.draw(screen)
            shield.update()
            shield.draw(screen)
            gravity.update()
            gravity.draw(screen)
            score.update(screen)
            pg.display.update()
            tmr += 1
            clock.tick(50)


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