ProjExD_3 / fight_kokaton.py

c0b2310742 追加機能３：とりあえず完了 1197209 · 50 minutes ago

243 lines (211 loc) · 8.52 KB

Code Blame Raw

```python
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
NUM_OF_BOMBS = 5  # 爆弾の個数
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
class Score:
    """
    スコア表示に関するクラス
    """
    def __init__(self):
        self.fonto = pg.font.SysFont("hgp創英角ﾎﾟｯﾌﾟ体", 30)
        self.color = (0, 0, 255)  # 青色
        self.score = 0
        self.img = self.fonto.render(f"スコア: {self.score}", True, self.color)
        self.rct = self.img.get_rect()
        self.rct.center = (100, HEIGHT - 50)

```

```python
    def update(self, screen: pg.Surface):
        self.img = self.fonto.render(f"スコア: {self.score}", True, self.color)
        screen.blit(self.img, self.rct)

class Bird:
    """
    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, -5),
        pg.K_DOWN: (0, +5),
        pg.K_LEFT: (-5, 0),
        pg.K_RIGHT: (+5, 0),
    }
    img0 = pg.transform.rotozoom(pg.image.load("fig/3.png"), 0, 0.9)
    img = pg.transform.flip(img0, True, False)  # デフォルトのこうかとん（右向き）
    imgs = {  # 0度から反時計回りに定義
        (+5, 0): img,  # 右
        (+5, -5): pg.transform.rotozoom(img, 45, 0.9),  # 右上
        (0, -5): pg.transform.rotozoom(img, 90, 0.9),  # 上
        (-5, -5): pg.transform.rotozoom(img0, -45, 0.9),  # 左上
        (-5, 0): img0,  # 左
        (-5, +5): pg.transform.rotozoom(img0, 45, 0.9),  # 左下
        (0, +5): pg.transform.rotozoom(img, -90, 0.9),  # 下
        (+5, +5): pg.transform.rotozoom(img, -45, 0.9),  # 右下
    }

    def __init__(self, xy: tuple[int, int]):
        """
        こうかとん画像Surfaceを生成する
        引数 xy：こうかとん画像の初期位置座標タプル
        """
        self.img = __class__.imgs[(+5, 0)]
        self.rct: pg.Rect = self.img.get_rect()
        self.rct.center = xy

    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.img = pg.transform.rotozoom(pg.image.load(f"fig/{num}.png"), 0, 0.9)
        screen.blit(self.img, self.rct)

    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]:
                    sum_mv[0] += mv[0]
                    sum_mv[1] += mv[1]
            self.rct.move_ip(sum_mv)
            if check_bound(self.rct) != (True, True):
                self.rct.move_ip(-sum_mv[0], -sum_mv[1])
            if not (sum_mv[0] == 0 and sum_mv[1] == 0):
                self.img = __class__.imgs[tuple(sum_mv)]
            screen.blit(self.img, self.rct)


    class Beam:
        """
        こうかとんが放つビームに関するクラス
        """
        def __init__(self, bird:"Bird"):
            """
            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（Birdインスタンス）
            """
            self.img = pg.image.load(f"fig/beam.png")
            self.rct = self.img.get_rect()
            self.rct.centery = bird.rct.centery  # こうかとんの中心縦座標
            self.rct.left = bird.rct.right  # こうかとんの右座標
            self.vx, self.vy = +5, 0

        def update(self, screen: pg.Surface):
            """
            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
            """
            if check_bound(self.rct) == (True, True):
                self.rct.move_ip(self.vx, self.vy)
                screen.blit(self.img, self.rct)


    class Bomb:
        """
        爆弾に関するクラス
        """
        def __init__(self, color: tuple[int, int, int], rad: int):
            """
            引数に基づき爆弾円Surfaceを生成する
            引数1 color：爆弾円の色タプル
            引数2 rad：爆弾円の半径
            """
            self.img = pg.Surface((2*rad, 2*rad))
            pg.draw.circle(self.img, color, (rad, rad), rad)
            self.img.set_colorkey((0, 0, 0))
            self.rct = self.img.get_rect()
            self.rct.center = random.randint(0, WIDTH), random.randint(0, HEIGHT)
```

```
142             self.vx, self.vy = +5, +5
143
144         def update(self, screen: pg.Surface):
145             """
146             爆弾を速度ベクトルself.vx, self.vyに基づき移動させる
147             引数 screen：画面Surface
148             """
149             yoko, tate = check_bound(self.rct)
150             if not yoko:
151                 self.vx *= -1
152             if not tate:
153                 self.vy *= -1
154             self.rct.move_ip(self.vx, self.vy)
155             screen.blit(self.img, self.rct)
156
157     class Explosion:
158         """
159         爆弾が破壊された時の爆発エフェクトに関するクラス
160         """
161         def __init__(self, center):
162             self.images = [
163                 pg.image.load("fig/explosion.gif")
164             ]
165             self.current_image = 0  # 現在表示している画像のインデックス
166             self.rct = self.images[0].get_rect(center=center)
167             self.life = 10  # 爆発の表示時間
168
169         def update(self, screen):
170             if self.life > 0:
171                 screen.blit(self.images[self.current_image], self.rct)
172                 self.life -= 1  # 表示時間を1フレーム分減算
173                 self.current_image = (self.current_image + 1) % len(self.images)  # 画像を交互に
174
175
176     def main():
177         pg.display.set_caption("たたかえ！こうかとん")
178         screen = pg.display.set_mode((WIDTH, HEIGHT))
179         bg_img = pg.image.load("fig/pg_bg.jpg")
180         bird = Bird((300, 200))
181         bombs = [Bomb((255, 0, 0), 10) for _ in range(NUM_OF_BOMBS)]
182         score = Score()
183         clock = pg.time.Clock()
184         beams = []  # 複数のビームを格納するリスト
185         explosions = []  # 複数の爆発エフェクトを格納するリスト
186
187         while True:
188             for event in pg.event.get():
189                 if event.type == pg.QUIT:
190                     return
191                 if event.type == pg.KEYDOWN and event.key == pg.K_SPACE:
192                     beams.append(Beam(bird))  # スペースキーで新しいビームを追加
193
194             screen.blit(bg_img, [0, 0])
```

```python

            # こうかとんと爆弾の衝突判定
            for bomb in bombs:
                if bird.rct.colliderect(bomb.rct):
                    bird.change_img(8, screen)
                    pg.display.update()
                    time.sleep(1)
                    return

            # ビームの更新と衝突判定
            for beam in beams:
                beam.update(screen)
            beams = [beam for beam in beams if check_bound(beam.rct) == (True, True)]  # 画面外

            for i, bomb in enumerate(bombs):
                for beam in beams:
                    if beam.rct.colliderect(bomb.rct):  # ビームが爆弾に衝突した場合
                        beams.remove(beam)  # 衝突したビームを削除
                        explosions.append(Explosion(bomb.rct.center))  # 爆発エフェクトを追加
                        bombs[i] = None  # 爆弾を削除
                        bird.change_img(6, screen)
                        score.score += 1
                        pg.display.update()

            # 破壊された爆弾をリストから削除
            bombs = [bomb for bomb in bombs if bomb is not None]

            key_lst = pg.key.get_pressed()
            bird.update(key_lst, screen)

            for bomb in bombs:
                bomb.update(screen)

            # 爆発エフェクトの更新
            for explosion in explosions:
                explosion.update(screen)
            explosions = [explosion for explosion in explosions if explosion.life > 0]  # 表示

            score.update(screen)
            pg.display.update()
            clock.tick(50)


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