ゲームのタイトル

エイリアンの襲来

実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

ゲームの概要

UFOが上から爆弾を落としてくる。そのUFOを撃ち落とす

ゲームの実装

共通基本機能

- 画像を読み込む関数
- 音を読み込む関数
- プレイヤーに関するクラス
- 敵に関するクラス
- 爆発に関するクラス
- 弾丸に関するクラス
- 爆弾に関するクラス
- 爆弾に関するクラス
- スコアに関するクラス
- メインの関数

担当追加機能

- 右shiftを押すと短時間弾の数が増える　(松村春輝 c0b22136)
- 戦車の数を２台に増やした (坂田匡弥 c0a22070)
- スコアが１０増えるごとに画面の敵をすべて倒す (山本昂 c0a22146)
- ゲームが終了時Gameoverと表示　(東杏澄 c0a21144)
- 各残機を１から３に増加　(福元悠太　c0b20136)

ToDo

- スタート画面の作成
- スコアのボーナスの選択肢の増加
- それぞれ違う機能の戦車を作成
- 弾の種類を増加
- 残機の数を表示

メモ

- playr1の移動は右がA左はD弾の発射はW
- playr2の移動は右がJ左はL弾の発射はU
- 右Shiftで一時的に弾を増やす

c0a22070 / ProjExD_05 Public

Code Issues Pull requests Actions Projects Wiki Security Insights Se

main ProjExD_05 / aliens.py Go to file t

c0a22070 2 hours ago

466 lines (371 loc) · 13.9 KB

Code Blame

```python
#!/usr/bin/env python
""" pygame.examples.aliens

Shows a mini game where you have to defend against aliens.

What does it show you about pygame?

* pg.sprite, the difference between Sprite and Group.
* dirty rectangle optimization for processing for speed.
* music with pg.mixer.music, including fadeout
* sound effects with pg.Sound
* event processing, keyboard handling, QUIT handling.
* a main loop frame limited with a game clock from pg.time.Clock
* fullscreen switching.


Controls
--------

* Left and right arrows to move.
* Space bar to shoot
* f key to toggle between fullscreen.

"""

import random
import os

# import basic pygame modules
import pygame as pg

# see if we can load more than standard BMP
if not pg.image.get_extended():
    raise SystemExit("Sorry, extended image module required")


# game constants
MAX_SHOTS =  4 # most player bullets onscreen
ALIEN_ODDS = 22  # chances a new alien appears
BOMB_ODDS = 60  # chances a new bomb will drop
ALIEN_RELOAD = 12  # frames between new aliens
SCREENRECT = pg.Rect(0, 0, 640, 480)
SCORE = 0

```

```python
main_dir = os.path.split(os.path.abspath(__file__))[0]


def load_image(file):
    """loads an image, prepares it for play"""
    file = os.path.join(main_dir, "data", file)
    try:
        surface = pg.image.load(file)
    except pg.error:
        raise SystemExit('Could not load image "%s" %s' % (file, pg.get_error()))
    return surface.convert()


def load_sound(file):
    """because pygame can be be compiled without mixer."""
    if not pg.mixer:
        return None
    file = os.path.join(main_dir, "data", file)
    try:
        sound = pg.mixer.Sound(file)
        return sound
    except pg.error:
        print("Warning, unable to load, %s" % file)
    return None


# Each type of game object gets an init and an update function.
# The update function is called once per frame, and it is when each object should
# change its current position and state.
#
# The Player object actually gets a "move" function instead of update,
# since it is passed extra information about the keyboard.


class Player1(pg.sprite.Sprite):
    """Representing the player as a moon buggy type car."""

    speed = 10
    bounce = 24
    gun_offset = -11
    images = []

    def __init__(self):
        pg.sprite.Sprite.__init__(self, self.containers)
        self.image = self.images[0]
        self.rect = self.image.get_rect(midbottom=SCREENRECT.midbottom)
        self.reloading = 0
        self.origtop = self.rect.top
        self.facing = -1

    def move(self, direction):
        if direction:
            self.facing = direction
        self.rect.move_ip(direction * self.speed, 0)
        self.rect = self.rect.clamp(SCREENRECT)
        if direction < 0:
            self.image = self.images[0]
        elif direction > 0:
            self.image = self.images[1]
        self.rect.top = self.origtop - (self.rect.left // self.bounce % 2)
```

```python

        def gunpos(self):
            pos = self.facing * self.gun_offset + self.rect.centerx
            return pos, self.rect.top

    class Player2(pg.sprite.Sprite): #２台目の戦車
        """Representing the player as a moon buggy type car."""

        speed = 10
        bounce = 24
        gun_offset = -11
        images = []

        def __init__(self):
            pg.sprite.Sprite.__init__(self, self.containers)
            self.image = self.images[0]
            self.rect = self.image.get_rect(midbottom=SCREENRECT.midbottom)
            self.reloading = 0
            self.origtop = self.rect.top
            self.facing = 1

        def move(self, direction):
            if direction:
                self.facing = direction
            self.rect.move_ip(direction * self.speed, 0)
            self.rect = self.rect.clamp(SCREENRECT)
            if direction < 0:
                self.image = self.images[0]
            elif direction > 0:
                self.image = self.images[1]
            self.rect.top = self.origtop - (self.rect.left // self.bounce % 2)

        def gunpos(self):
            pos = self.facing * self.gun_offset + self.rect.centerx
            return pos, self.rect.top


    class Alien(pg.sprite.Sprite):
        """An alien space ship. That slowly moves down the screen."""

        speed = 13
        animcycle = 12
        images = []

        def __init__(self):
            pg.sprite.Sprite.__init__(self, self.containers)
            self.image = self.images[0]
            self.rect = self.image.get_rect()
            self.facing = random.choice((-1, 1)) * Alien.speed
            self.frame = 0
            if self.facing < 0:
                self.rect.right = SCREENRECT.right

        def update(self):
            self.rect.move_ip(self.facing, 0)
            if not SCREENRECT.contains(self.rect):
                self.facing = -self.facing
                self.rect.top = self.rect.bottom + 1
                self.rect = self.rect.clamp(SCREENRECT)
            self.frame = self.frame + 1
```

```python
            self.image = self.images[self.frame // self.animcycle % 3]


class Explosion(pg.sprite.Sprite):
    """An explosion. Hopefully the Alien and not the player!"""

    defaultlife = 12
    animcycle = 3
    images = []

    def __init__(self, actor):
        pg.sprite.Sprite.__init__(self, self.containers)
        self.image = self.images[0]
        self.rect = self.image.get_rect(center=actor.rect.center)
        self.life = self.defaultlife

    def update(self):
        """called every time around the game loop.

        Show the explosion surface for 'defaultlife'.
        Every game tick(update), we decrease the 'life'.

        Also we animate the explosion.
        """
        self.life = self.life - 1
        self.image = self.images[self.life // self.animcycle % 2]
        if self.life <= 0:
            self.kill()


class Shot(pg.sprite.Sprite):
    """a bullet the Player sprite fires."""

    speed = -11
    images = []

    def __init__(self, pos):
        pg.sprite.Sprite.__init__(self, self.containers)
        self.image = self.images[0]
        self.rect = self.image.get_rect(midbottom=pos)

    def update(self):
        """called every time around the game loop.

        Every tick we move the shot upwards.
        """
        self.rect.move_ip(0, self.speed)
        if self.rect.top <= 0:
            self.kill()


class Bomb(pg.sprite.Sprite):
    """A bomb the aliens drop."""

    speed = 9
    images = []

    def __init__(self, alien):
        pg.sprite.Sprite.__init__(self, self.containers)
        self.image = self.images[0]
```

```
            self.rect = self.image.get_rect(midbottom=alien.rect.move(0, 5).midbottom)

        def update(self):
            """called every time around the game loop.

            Every frame we move the sprite 'rect' down.
            When it reaches the bottom we:

            - make an explosion.
            - remove the Bomb.
            """
            self.rect.move_ip(0, self.speed)
            if self.rect.bottom >= 470:
                Explosion(self)
                self.kill()


    class Score(pg.sprite.Sprite):
        """to keep track of the score."""

        def __init__(self):
            pg.sprite.Sprite.__init__(self)
            self.font = pg.font.Font(None, 20)
            self.font.set_italic(1)
            self.color = "white"
            self.lastscore = -1
            self.update()
            self.rect = self.image.get_rect().move(10, 450)

        def update(self):
            """We only update the score in update() when it has changed."""
            if SCORE != self.lastscore:
                self.lastscore = SCORE
                msg = "Score: %d" % SCORE
                self.image = self.font.render(msg, 0, self.color)


    def main(winstyle=0):
        # Initialize pygame
        if pg.get_sdl_version()[0] == 2:
            pg.mixer.pre_init(44100, 32, 2, 1024)
        pg.init()
        if pg.mixer and not pg.mixer.get_init():
            print("Warning, no sound")
            pg.mixer = None

        fullscreen = False
        # Set the display mode
        winstyle = 0  # |FULLSCREEN
        bestdepth = pg.display.mode_ok(SCREENRECT.size, winstyle, 32)
        screen = pg.display.set_mode(SCREENRECT.size, winstyle, bestdepth)

        # Load images, assign to sprite classes
        # (do this before the classes are used, after screen setup)
        img = load_image("player1.gif")
        Player1.images = [img, pg.transform.flip(img, 1, 0)]
        img = load_image("player2.gif")  #２台目の戦車
        Player2.images = [img, pg.transform.flip(img, -1, 0)]  # ２台目の戦車
        img = load_image("explosion1.gif")
        Explosion.images = [img, pg.transform.flip(img, 1, 1)]
        Alien.images = [load_image(im) for im in ("alien1.gif", "alien2.gif", "alien3.gif")]
```

```
    Alien.images = [load_image(im) for im in ("alien1.gif", "alien2.gif", "alien3.gif")]
    Bomb.images = [load_image("bomb.gif")]
    Shot.images = [load_image("shot.gif")]

    # decorate the game window
    icon = pg.transform.scale(Alien.images[0], (32, 32))
    pg.display.set_icon(icon)
    pg.display.set_caption("Pygame Aliens")
    pg.mouse.set_visible(0)

    # create the background, tile the bgd image
    bgdtile = load_image("background.gif")
    background = pg.Surface(SCREENRECT.size)
    for x in range(0, SCREENRECT.width, bgdtile.get_width()):
        background.blit(bgdtile, (x, 0))
    screen.blit(background, (0, 0))
    pg.display.flip()

    # load the sound effects
    boom_sound = load_sound("boom.wav")
    shoot_sound = load_sound("car_door.wav")
    if pg.mixer:
        music = os.path.join(main_dir, "data", "house_lo.wav")
        pg.mixer.music.load(music)
        pg.mixer.music.play(-1)

    # Initialize Game Groups
    aliens = pg.sprite.Group()
    shots = pg.sprite.Group()
    bombs = pg.sprite.Group()
    all = pg.sprite.RenderUpdates()
    lastalien = pg.sprite.GroupSingle()

    # assign default groups to each sprite class
    Player1.containers = all
    Player2.containers = all  # ２台目の戦車
    Alien.containers = aliens, all, lastalien
    Shot.containers = shots, all
    Bomb.containers = bombs, all
    Explosion.containers = all
    Score.containers = all

    # Create Some Starting Values
    global score
    alienreload = ALIEN_RELOAD
    clock = pg.time.Clock()

    # initialize our starting sprites
    global SCORE
    player1 = Player1()
    player2 = Player2()  # ２台目の戦車
    Alien()  # note, this 'lives' because it goes into a sprite group
    if pg.font:
        all.add(Score())

    # Run our main loop whilst the player is alive.
    while player1.alive():

        # get input
        for event in pg.event.get():
            if event.type == pg.QUIT:
```

```
                if event.type == pg.QUIT:
                    return
                if event.type == pg.KEYDOWN and event.key == pg.K_ESCAPE:
                    return
                elif event.type == pg.KEYDOWN:
                    if event.key == pg.K_f:
                        if not fullscreen:
                            print("Changing to FULLSCREEN")
                            screen_backup = screen.copy()
                            screen = pg.display.set_mode(
                                SCREENRECT.size, winstyle | pg.FULLSCREEN, bestdepth
                            )
                            screen.blit(screen_backup, (0, 0))
                        else:
                            print("Changing to windowed mode")
                            screen_backup = screen.copy()
                            screen = pg.display.set_mode(
                                SCREENRECT.size, winstyle, bestdepth
                            )
                            screen.blit(screen_backup, (0, 0))
                        pg.display.flip()
                        fullscreen = not fullscreen




            keystate = pg.key.get_pressed()

            # clear/erase the last drawn sprites
            all.clear(screen, background)

            # update all the sprites
            all.update()

            # handle player input
            direction1 = keystate[pg.K_d] - keystate[pg.K_a]
            player1.move(direction1)
            direction2 = keystate[pg.K_l] - keystate[pg.K_j]
            player2.move(direction2)  # ２台目の戦車
            firing1 = keystate[pg.K_w]
            firing2 = keystate[pg.K_i]
            if not player1.reloading and firing1 and len(shots) < MAX_SHOTS:
                Shot(player1.gunpos())
                if pg.mixer:
                    shoot_sound.play()
            player1.reloading = firing1

            if not player2.reloading and firing2 and len(shots) < MAX_SHOTS:  # ２台目の戦車
                Shot(player2.gunpos())
                if pg.mixer:
                    shoot_sound.play()
            player2.reloading = firing2  # ２台目の戦車


            # Create new alien
            if alienreload:
                alienreload = alienreload - 1
            elif not int(random.random() * ALIEN_ODDS):
                Alien()
                alienreload = ALIEN_RELOAD

```

```
            # Drop bombs
            if lastalien and not int(random.random() * BOMB_ODDS):
                Bomb(lastalien.sprite)

            # Detect collisions between aliens and players.
            for alien in pg.sprite.spritecollide(player1, aliens, 1):
                if pg.mixer:
                    boom_sound.play()
                Explosion(alien)
                Explosion(player1)
                SCORE = SCORE + 1
                player1.kill()
            for alien in pg.sprite.spritecollide(player2, aliens, 1):  # ２台目の戦車
                if pg.mixer:
                    boom_sound.play()
                Explosion(alien)
                Explosion(player2)
                SCORE = SCORE + 1
                player2.kill()

            # See if shots hit the aliens.
            for alien in pg.sprite.groupcollide(aliens, shots, 1, 1).keys():
                if pg.mixer:
                    boom_sound.play()
                Explosion(alien)
                SCORE = SCORE + 1

            # See if alien boms hit the player.
            for bomb in pg.sprite.spritecollide(player1, bombs, 1):
                if pg.mixer:
                    boom_sound.play()
                Explosion(player1)
                Explosion(bomb)
                player1.kill()



            for bomb in pg.sprite.spritecollide(player2, bombs, 1):   # ２台目の戦車
                if pg.mixer:
                    boom_sound.play()
                Explosion(player2)
                Explosion(bomb)
                player2.kill()

            # draw the scene
            dirty = all.draw(screen)
            pg.display.update(dirty)

            # cap the framerate at 40fps. Also called 40HZ or 40 times per second.
            clock.tick(40)


        if pg.mixer:
            pg.mixer.music.fadeout(1000)
        pg.time.wait(1000)


    # call the "main" function if running this script
    if __name__ == "__main__":
        main()
```

```
        pg.quit()
```