ProjExD_Group15 / game.py

c0c2400957 アイテム機能の最終調整とREADMEの更新(修正) 841867e · 14 minutes ago

402 lines (355 loc) · 18.9 KB

Code Blame Raw

```python
import pygame
import sys
import random
import os
import math

# --- スクリプトのパスを基準にディレクトリを設定 ---
# このスクリプト(game.py)があるフォルダの絶対パスを取得
script_dir = os.path.dirname(os.path.abspath(__file__))
# 画像が保存されている'fig'フォルダへのパスを作成
fig_dir = os.path.join(script_dir, "fig")

# --- 定数 (Constants) ---
SCREEN_WIDTH = 600
SCREEN_HEIGHT = 800
FPS = 60

# 色 (Colors)
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 255, 0)
GREEN = (0, 255, 0)
CYAN = (0, 255, 255)

# --- ゲームの初期化 (Game Initialization) ---
pygame.init()
pygame.font.init() # フォントモジュールを初期化
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Xevious Style Shooter")
clock = pygame.time.Clock() # FPSを管理するためのClockオブジェクト

# --- 画像ファイルの読み込み (Load Images) ---
# 'fig'フォルダが存在しない場合は作成する
if not os.path.exists(fig_dir):
    os.makedirs(fig_dir)
    print(f"Warning: '{fig_dir}' directory not found. Please place images inside.")

try:
    # 各種画像の読み込み。convert_alpha()で透明度を扱えるようにする
    PLAYER_IMAGE = pygame.image.load(os.path.join(fig_dir, "koukaton.png")).convert_alpha()
    ENEMY_IMAGE = pygame.image.load(os.path.join(fig_dir, "enemy.png")).convert_alpha()
```

```
        PLAYER_BULLET_IMAGE = pygame.image.load(os.path.join(fig_dir, "beam.png")).convert_alpha()
        # try-exceptで画像ファイルがなくてもエラーで落ちないようにする（フォールバック処理）
        try: LAZER_IMAGE = pygame.image.load(os.path.join(fig_dir, "lazer.png")).convert_alpha()
        except pygame.error: print("Warning: lazer.png not found."); LAZER_IMAGE = pygame.Surface((20, SCR
        try: HEAL_ITEM_IMAGE = pygame.image.load(os.path.join(fig_dir, "heal.png")).convert_alpha()
        except pygame.error: HEAL_ITEM_IMAGE = pygame.Surface((25, 25)); HEAL_ITEM_IMAGE.fill(GREEN)
        try: ATTACK_ITEM_IMAGE = pygame.image.load(os.path.join(fig_dir, "attack.png")).convert_alpha()
        except pygame.error: ATTACK_ITEM_IMAGE = pygame.Surface((25, 25)); ATTACK_ITEM_IMAGE.fill(YELLOW)
        try: EXPLOSION_IMAGE = pygame.image.load(os.path.join(fig_dir, "explosion.png")).convert_alpha()
        except pygame.error: print("Warning: explosion.png not found."); EXPLOSION_IMAGE = pygame.Surface(
    except pygame.error as e:
        print(f"Error loading image: {e}"); pygame.quit(); sys.exit()

    # --- クラス定義 ---

    class Player(pygame.sprite.Sprite):
        """プレイヤー機体を管理するクラス"""
        def __init__(self):
            """プレイヤーの初期設定を行う"""
            super().__init__()
            self.image = pygame.transform.scale(PLAYER_IMAGE, (40, 40))
            self.rect = self.image.get_rect(centerx=SCREEN_WIDTH // 2, bottom=SCREEN_HEIGHT - 30)
            self.speed_x = 0
            self.hidden = False # プレイヤーが非表示（やられた後など）かどうかのフラグ
            # 体力関連
            self.max_health = 100
            self.health = self.max_health
            # 攻撃関連
            self.shoot_delay = 250 # 弾の発射間隔（ミリ秒）
            self.last_shot = pygame.time.get_ticks() # 最後に弾を撃った時刻
            self.powerup_level = 0 # 0:通常, 1:3方向, 2:レーザー
            self.powerup_duration = 7000 # パワーアップの持続時間（ミリ秒）
            self.powerup_end_time = 0 # パワーアップの終了時刻
            self.active_laser = None # 照射中のレーザーオブジェクトを保持

        def update(self):
            """プレイヤーの状態を毎フレーム更新する"""
            if self.hidden: return # 非表示中は更新しない
            # キー入力に応じた左右移動
            self.speed_x = 0
            keys = pygame.key.get_pressed()
            if keys[pygame.K_LEFT]: self.speed_x = -7
            if keys[pygame.K_RIGHT]: self.speed_x = 7
            self.rect.x += self.speed_x
            # 画面端からはみ出ないように制御
            if self.rect.right > SCREEN_WIDTH: self.rect.right = SCREEN_WIDTH
            if self.rect.left < 0: self.rect.left = 0
            # パワーアップの持続時間が過ぎたら元に戻す
            if self.powerup_level > 0 and pygame.time.get_ticks() > self.powerup_end_time:
                self.powerup_level = 0
                if self.active_laser: # レーザーが出ていれば消す
                    self.active_laser.kill()
                    self.active_laser = None
                print("Power-up ended.")

        def shoot(self, all_sprites, bullets_group):
            """弾を発射する"""
            if self.hidden: return
```

```
            now = pygame.time.get_ticks()
            # 発射間隔(shoot_delay)を守って発射
            if now - self.last_shot > self.shoot_delay:
                self.last_shot = now
                # パワーアップレベルに応じて弾の種類を変える
                if self.powerup_level == 0: # 通常弾
                    bullet = PlayerBullet(self.rect.centerx, self.rect.top)
                    all_sprites.add(bullet); bullets_group.add(bullet)
                elif self.powerup_level == 1: # 3方向弾
                    bullet1 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=0)
                    bullet2 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=-3)
                    bullet3 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=3)
                    all_sprites.add(bullet1, bullet2, bullet3); bullets_group.add(bullet1, bullet2, bullet
                elif self.powerup_level >= 2: # レーザー照射中はサイドの弾だけ発射
                    bullet2 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=-4)
                    bullet3 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=4)
                    all_sprites.add(bullet2, bullet3); bullets_group.add(bullet2, bullet3)

        def take_damage(self, amount):
            """ダメージを受ける処理"""
            self.health -= amount
            self.health = max(0, self.health) # 体力がマイナスにならないように
            return self.health <= 0 # 体力が0以下になったらTrueを返す

        def heal(self, amount):
            """体力を回復する処理"""
            self.health += amount
            self.health = min(self.health, self.max_health) # 最大体力を超えないように

        def power_up(self):
            """攻撃をパワーアップさせる処理"""
            self.powerup_level = min(2, self.powerup_level + 1) # 最大レベル2まで
            self.powerup_end_time = pygame.time.get_ticks() + self.powerup_duration # 終了タイマーをリセット
            print(f"Power-up! Level {self.powerup_level}")

        def hide(self):
            """プレイヤーを非表示にする（ゲームオーバー時）"""
            self.hidden = True
            self.kill()

    class Enemy(pygame.sprite.Sprite):
        """敵機を管理するクラス"""
        def __init__(self, speed_level=0, player_ref=None):
            """敵の初期設定を行う"""
            super().__init__()
            self.image = pygame.transform.scale(ENEMY_IMAGE, (40, 40))
            self.rect = self.image.get_rect(x=random.randrange(0, SCREEN_WIDTH - 40), y=random.randrange(-
            # ゲームレベルに応じて落下速度を上げる
            speed_increase = speed_level * 0.4
            self.speed_y = random.randrange(int(2 + speed_increase), int(5 + speed_increase) + 1)
            self.player = player_ref
            self.health = 1
            self.score_value = 1

        def update(self):
            """敵の状態を毎フレーム更新する（下に移動）"""
            self.rect.y += self.speed_y
            if self.rect.top > SCREEN_HEIGHT + 10: # 画面外に出たら消える
```

```
159                 self.kill()
160
161 class PlayerBullet(pygame.sprite.Sprite):
162     """プレイヤーの弾を管理するクラス"""
163     def __init__(self, x, y, speed_x=0):
164         """弾の初期設定を行う"""
165         super().__init__()
166         self.image = pygame.transform.scale(PLAYER_BULLET_IMAGE, (10, 25))
167         self.rect = self.image.get_rect(center=(x, y))
168         self.speed_y = -10 # 上方向への速度
169         self.speed_x = speed_x # 横方向への速度（3方向弾用）
170
171     def update(self):
172         """弾の状態を毎フレーム更新する（指定された速度で移動）"""
173         self.rect.y += self.speed_y
174         self.rect.x += self.speed_x
175         if not screen.get_rect().colliderect(self.rect): # 画面外に出たら消える
176             self.kill()
177
178 class SuperLaser(pygame.sprite.Sprite):
179     """照射式レーザーを管理するクラス"""
180     def __init__(self, player_obj):
181         """レーザーの初期設定を行う"""
182         super().__init__()
183         self.player = player_obj # プレイヤーへの参照を保持
184         self.image = pygame.transform.scale(LAZER_IMAGE, (20, SCREEN_HEIGHT))
185         self.rect = self.image.get_rect()
186         self.update() # 初期位置をプレイヤーに合わせる
187
188     def update(self):
189         """毎フレーム、プレイヤーの位置に追従する"""
190         self.rect.centerx = self.player.rect.centerx
191         self.rect.bottom = self.player.rect.top
192
193 class Item(pygame.sprite.Sprite):
194     """全てのアイテムの親となる基本クラス"""
195     def __init__(self, center):
196         """アイテムの基本設定"""
197         super().__init__()
198         self.rect = self.image.get_rect(center=center) # 画像(self.image)は子クラスで設定
199         self.speed_y = 3
200
201     def update(self):
202         """アイテムの状態を毎フレーム更新する（下に移動）"""
203         self.rect.y += self.speed_y
204         if self.rect.top > SCREEN_HEIGHT: # 画面外に出たら消える
205             self.kill()
206
207 class HealItem(Item):
208     """回復アイテムのクラス"""
209     def __init__(self, center):
210         self.image = pygame.transform.scale(HEAL_ITEM_IMAGE, (30, 30))
211         super().__init__(center) # 親クラスの初期化処理を呼び出す
212
213     def apply_effect(self, player):
214         """取得した際の回復効果を適用する"""
215         player.heal(25)
216
```

```python
class AttackUpItem(Item):
    """攻撃力アップアイテムのクラス"""
    def __init__(self, center):
        self.image = pygame.transform.scale(ATTACK_ITEM_IMAGE, (30, 30))
        super().__init__(center)

    def apply_effect(self, player):
        """取得した際のパワーアップ効果を適用する"""
        player.power_up()

class Explosion(pygame.sprite.Sprite):
    """爆発エフェクトを管理するクラス"""
    def __init__(self, center, size="normal"):
        """爆発の初期設定を行う"""
        super().__init__()
        self.original_image = EXPLOSION_IMAGE
        # "large"か"normal"かでサイズを変える
        scale = (90, 90) if size == "large" else (60, 60)
        self.image = pygame.transform.scale(self.original_image, scale)
        self.rect = self.image.get_rect(center=center)
        self.duration = 400 # 表示時間（ミリ秒）
        self.creation_time = pygame.time.get_ticks() # 生成された時刻を記録

    def update(self):
        """毎フレーム、表示時間が過ぎたかチェックする"""
        if pygame.time.get_ticks() - self.creation_time > self.duration:
            self.kill() # 一定時間経ったら消える

# --- 描画関数群 ---

def draw_stars(surface, stars, speed_level=0):
    """背景の星を描画し、スクロールさせる"""
    speed_modifier = 1.0 + speed_level * 0.15 # ゲームレベルに応じてスクロール速度を上げる
    for star in stars:
        pygame.draw.circle(surface, WHITE, (star[0], star[1]), star[3])
        star[1] += star[2] * speed_modifier
        if star[1] > SCREEN_HEIGHT: # 画面下に出たら上に戻す
            star[0] = random.randrange(0, SCREEN_WIDTH); star[1] = 0

def draw_text(surface, text, font, color, x, y, align="topright"):
    """指定された位置にテキストを描画する"""
    text_surface = font.render(text, True, color)
    text_rect = text_surface.get_rect()
    if align == "topright": text_rect.topright = (x, y)
    elif align == "center": text_rect.center = (x, y)
    elif align == "topleft": text_rect.topleft = (x, y)
    surface.blit(text_surface, text_rect)

def draw_health_bar(surface, x, y, pct):
    """プレイヤーの体力バーを描画する"""
    pct = max(0, pct)
    BAR_LENGTH, BAR_HEIGHT = 150, 15
    fill = (pct / 100) * BAR_LENGTH
    bar_color = GREEN if pct > 60 else YELLOW if pct > 30 else RED # 体力に応じて色を変える
    pygame.draw.rect(surface, bar_color, (x, y, fill, BAR_HEIGHT))
    pygame.draw.rect(surface, WHITE, (x, y, BAR_LENGTH, BAR_HEIGHT), 2) # 白い枠線

# --- ゲームのセットアップ ---
```

```
    # フォントの準備
    score_font = pygame.font.SysFont(None, 36)
    game_over_font = pygame.font.SysFont(None, 64, bold=True)
    info_font = pygame.font.SysFont(None, 30)
    # 背景の星をランダムに生成
    stars = [[random.randrange(0,SCREEN_WIDTH),random.randrange(0,SCREEN_HEIGHT),random.randrange(1,4),ran
    # スプライトを管理するためのグループを作成
    all_sprites = pygame.sprite.Group() # 全てのスプライト（描画・更新用）
    enemies_group = pygame.sprite.Group() # 敵（衝突判定用）
    player_bullets_group = pygame.sprite.Group() # プレイヤーの弾（衝突判定用）
    items_group = pygame.sprite.Group() # アイテム（衝突判定用）
    laser_group = pygame.sprite.Group() # レーザー（衝突判定用）
    # プレイヤーインスタンスを生成し、グループに追加
    player = Player()
    all_sprites.add(player)
    # 敵を定期的に生成するためのカスタムイベントを設定
    ADD_ENEMY = pygame.USEREVENT + 1
    pygame.time.set_timer(ADD_ENEMY, 1000) # 1000ミリ秒(1秒)ごとに発生
    # ゲームの状態を管理する変数を初期化
    score, game_speed_level, game_over, running = 0, 0, False, True

    # --- メインゲームループ (Main Game Loop) ---
    while running:
        # フレームレートの制御
        clock.tick(FPS)

        # --- イベント処理 ---
        for event in pygame.event.get():
            # ウィンドウの×ボタンが押されたら終了
            if event.type == pygame.QUIT:
                running = False
            # ゲームオーバー中に、何かキーが押されたら終了
            elif game_over and event.type == pygame.KEYDOWN:
                running = False
            # ADD_ENEMYイベントが発生したら、新しい敵を生成
            elif event.type == ADD_ENEMY and not game_over:
                enemy = Enemy(game_speed_level, player)
                all_sprites.add(enemy)
                enemies_group.add(enemy)

        # --- 操作処理 ---
        keys = pygame.key.get_pressed()
        # スペースキーが押されていたら弾を発射
        if keys[pygame.K_SPACE] and not game_over:
            player.shoot(all_sprites, player_bullets_group)

        # レーザーの照射処理（パワーアップレベル2以上でスペースキー長押し）
        if player.powerup_level >= 2 and not game_over:
            if keys[pygame.K_SPACE]:
                if not player.active_laser: # レーザーがなければ生成
                    player.active_laser = SuperLaser(player)
                    all_sprites.add(player.active_laser); laser_group.add(player.active_laser)
            else: # スペースを離したら消す
                if player.active_laser:
                    player.active_laser.kill(); player.active_laser = None
        else: # パワーアップ中でなければ消す
            if player.active_laser:
                player.active_laser.kill(); player.active_laser = None
```

```
333
334         # --- 更新処理 ---
335         if not game_over:
336             all_sprites.update() # 全てのスプライトの状態を更新
337         else: # ゲームオーバー後は爆発エフェクトだけ更新
338             for s in all_sprites:
339                 if isinstance(s, Explosion):
340                     s.update()
341 
342         # --- 衝突判定 ---
343         if not game_over:
344             # プレイヤーの弾/レーザーと敵の衝突
345             # groupcollideで衝突したペアを検出し、敵と弾を消す
346             hits_bullet = pygame.sprite.groupcollide(player_bullets_group, enemies_group, True, True)
347             hits_laser = pygame.sprite.groupcollide(laser_group, enemies_group, False, True) # レーザーは消
348             hits_bullet.update(hits_laser) # ２つの衝突結果をマージする
349 
350             # 衝突した後処理（スコア加算、爆発、アイテムドロップ）
351             for weapon, enemies_hit in hits_bullet.items():
352                 for enemy in enemies_hit:
353                     score += enemy.score_value
354                     all_sprites.add(Explosion(enemy.rect.center, "normal"))
355                     # 20%の確率でアイテムをドロップ
356                     if random.random() > 0.8:
357                         item = random.choice([HealItem, AttackUpItem])(enemy.rect.center)
358                         all_sprites.add(item); items_group.add(item)
359 
360             # ゲームレベルの更新（スコアが10上がるごとにレベルアップ）
361             new_level = score // 10
362             if new_level > game_speed_level:
363                 game_speed_level = new_level
364                 # 敵の出現間隔を短くする
365                 rate = max(150, int(1000 * (0.9 ** game_speed_level)))
366                 pygame.time.set_timer(ADD_ENEMY, rate)
367 
368             # プレイヤーと敵の衝突
369             if pygame.sprite.spritecollide(player, enemies_group, True):
370                 # 10ダメージ受け、体力が0になったらゲームオーバー
371                 if player.take_damage(10):
372                     game_over = True
373                     all_sprites.add(Explosion(player.rect.center, "large"))
374                     player.hide()
375                 else: # まだ生きている場合は小さな爆発
376                     all_sprites.add(Explosion(player.rect.center, "normal"))
377 
378             # プレイヤーとアイテムの衝突
379             for item in pygame.sprite.spritecollide(player, items_group, True):
380                 item.apply_effect(player) # アイテムの効果を適用
381 
382         # --- 描画処理 ---
383         screen.fill(BLACK) # 画面を黒で塗りつぶす
384         draw_stars(screen, stars, game_speed_level) # 背景の星を描画
385         all_sprites.draw(screen) # 全てのスプライトを描画
386         # UI（スコア、レベル、体力バー）を描画
387         draw_text(screen, f"SCORE: {score}", score_font, WHITE, SCREEN_WIDTH - 10, 10, "topright")
388         draw_text(screen, f"LEVEL: {game_speed_level}", score_font, WHITE, 10, 10, "topleft")
389         draw_text(screen, "HP", score_font, WHITE, 10, 40, "topleft")
390         draw_health_bar(screen, 50, 45, player.health)
```

```
391
392         # ゲームオーバー画面の表示
393         if game_over:
394             draw_text(screen, "GAME OVER", game_over_font, RED, SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2, "ce
395             draw_text(screen, "Press any key to exit", info_font, WHITE, SCREEN_WIDTH // 2, SCREEN_HEIGHT
396
397         # 描画内容を画面に反映
398         pygame.display.flip()
399
400     # --- 終了処理 ---
401     pygame.quit()
402     sys.exit()
```