Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base repository: c0a24147d3/ProjExD_Group15 base: main

head repository: c0c2400957/ProjExD_Group15 compare: C0C24009/item_drop

✕ Can't automatically merge. Don't worry, you can still create the pull request.

Discuss and review the changes in this comparison with others. Learn about pull requests **Create pull request**

3 commits 6 files changed 2 contributors

Commits on Oct 28, 2025

アイテム機能の最終調整とREADMEの更新
c0c2400957 committed 37 minutes ago — ae60c6c

アイテム機能の最終調整とREADMEの更新(修正)
c0c2400957 committed 30 minutes ago — 841867e

C0C24009_説明追加.md
c0c2400957 authored 19 minutes ago — Verified — ed17a01

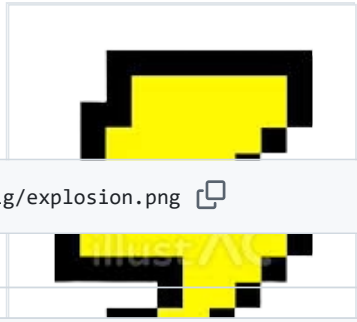
Showing 6 changed files with 313 additions and 349 deletions. Split Unified

42 README.md

```
5  5   * pygame >= 2.1
6  6
7  7   ## ゲームの概要
8      - * 主人公の戦闘機が、迫りくる敵機や障害物をビームで粉砕し突き進むゲーム
9      - * 参考URL：[サイトタイトル](https://www.nintendo.com/jp/games/feature/nintendo-classics/clv-p-hafcj/index.html)
   8   + * 主人公の戦闘機が、迫りくる敵機をビームで粉砕しながら突き進む縦スクロールシューティングゲーム。
   9   + * アイテムを取得することで自機を強化し、敵機を殲滅することが目標。
10 10
11 11  ## ゲームの遊び方
12     - * 矢印キーで機体を操作し，スペースキー押下による発射するビームで敵機や障害物を破壊する
13     - * 敵機の攻撃や障害物との衝突で、機体に一定以上のダメージが蓄積するとゲームオーバーとなる
   12  + * 矢印キーで機体を左右に操作し，スペースキー押下でビームを発射して敵機を破壊。
   13  + * スペースキーを長押しすると、パワーアップ段階に応じて強力なレーザーを照射し続けることができる。
   14  + * 敵機と衝突するとダメージを受け、体力が0になるとゲームオーバーとなる。
14 15
```

```
15 16   ## ゲームの実装
16 17   ### 共通基本機能
17    - * 背景画像とメイン機体、敵機の描画
   18 + * 背景（星空）のスクロールと、自機および敵機の描画
   19 + * プレイヤーの操作（左右移動、弾の発射）
   20 + * 敵機の自動生成と画面内での動き
18 21
19    - ### 分担追加機能(仮)
20    - * ???（担当：ソエタ）：障害物の出現
21    - * ???（担当：うめちゃん）：アイテムの出現
22    - * ???（担当：かつみ）：回避などの特殊コマンド
23    - * ???（担当：のうち）：大ボス
24    - * ???（担当：オウ）：中ボス
   22 + ### 分担追加機能
   23 + * 障害物の出現（担当：ソエタ）
   24 + * **アイテム機能（担当：梅村章太）：敵機を破壊すると、確率でアイテムが出現。取得すると体力が回復したり、一定時間ビームが3方向に発射されたり、強力なレーザーを照射できるようになったりする。**
   25 + * 回避などの特殊コマンド（担当：かつみ）
   26 + * 大ボス（担当：のうち）
   27 + * 中ボス（担当：オウ）
25 28
26 29   ### ToDo
27    - - [ ] ほげほげ機能
28    - - [ ] ふがふが関数内の変数名の統一
   30 + - [ ] 敵機の攻撃（ビーム発射）機能の実装
   31 + - [ ] ゲームバランスの調整（敵の出現頻度、アイテムドロップ率など）
29 32
30 33   ### メモ
31 34   * クラス内の変数は，すべて，「get_変数名」という名前のメソッドを介してアクセスするように設計してある
32 35   * すべてのクラスに関係する関数は，クラスの外で定義してある
33    - *
34    - ・でっかいボス
35    - ・ライフ制
36    - ・アイテム降らす
37    - ・もう一種類のたまに出てくる強敵
38    - ・難易度選択
39    - ・障害物
   36 + * でっかいボス
   37 + * ライフ制
   38 + * アイテム降らす
   39 + * 中ボス、大ボス
   40 + * ナインイド選択
   41 + * 障害物
```

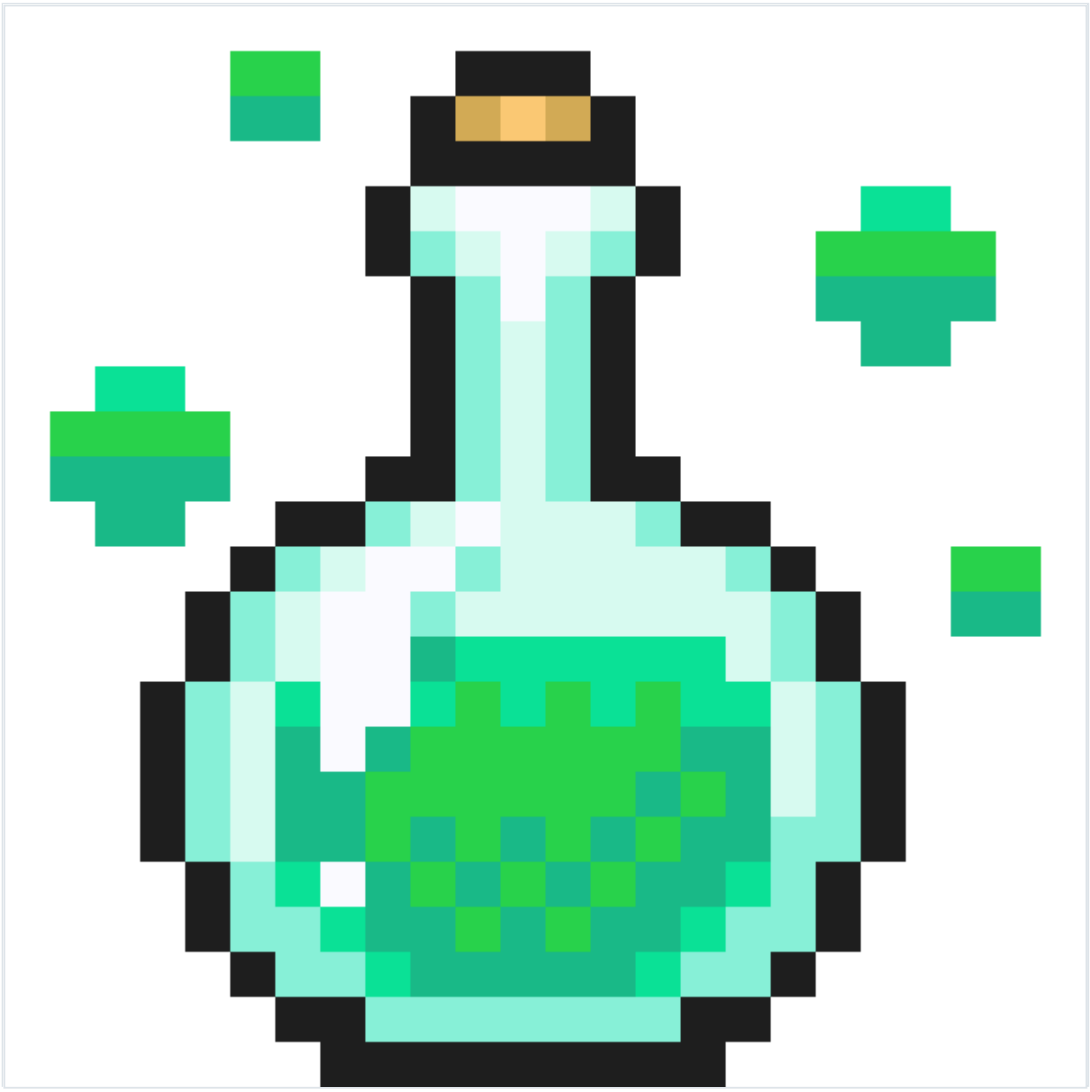
BIN +24.9 KB fig/attack.png



0 fig/explosion.gif → fig/explosion.png

File renamed without changes

BIN +19.9 KB fig/heal.png



BIN +7.38 KB fig/lazer.png

620 game.py

```
...  ...  @@ -1,11 +1,13 @@
 1    1   import pygame
```

```
2   2     import sys
3   3     import random
4       - import os # osモジュールをインポート
5       - import math # 角度計算のために math をインポート
        4 + import os
        5 + import math
6   6
7   7     # --- スクリプトのパスを基準にディレクトリを設定 ---
        8 + # このスクリプト(game.py)があるフォルダの絶対パスを取得
8   9     script_dir = os.path.dirname(os.path.abspath(__file__))
       10 + # 画像が保存されている'fig'フォルダへのパスを作成
9   11    fig_dir = os.path.join(script_dir, "fig")
10  12
11  13    # --- 定数 (Constants) ---
16  18    # 色 (Colors)
17  19    WHITE = (255, 255, 255)
18  20    BLACK = (0, 0, 0)
19      - RED = (255, 0, 0)
20      - YELLOW = (255, 255, 0)
       21 + RED = (255, 0, 0)
       22 + YELLOW = (255, 255, 0)
       23 + GREEN = (0, 255, 0)
       24 + CYAN = (0, 255, 255)
21  25
22  26    # --- ゲームの初期化 (Game Initialization) ---
23      - # ★★重要: 画像をロードする前に初期化と画面設定を完了させます★★
24  27    pygame.init()
25      - pygame.font.init()
       28 + pygame.font.init() # フォントモジュールを初期化
26  29    screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
27  30    pygame.display.set_caption("Xevious Style Shooter")
28      - clock = pygame.time.Clock()
29      -
       31 + clock = pygame.time.Clock() # FPSを管理するためのClockオブジェクト
30  32
31  33    # --- 画像ファイルの読み込み (Load Images) ---
       34 + # 'fig'フォルダが存在しない場合は作成する
32  35    if not os.path.exists(fig_dir):
33  36        os.makedirs(fig_dir)
34      -     print(f"Warning: '{fig_dir}' directory not found. Created an empty one.")
35      -     print("Please place 'koukaton.png', 'enemy.png', 'beam.png' and explosion frames
          (e.g., 'explosion_00.png', 'explosion_01.png'...) inside 'fig' folder.")
       37 +     print(f"Warning: '{fig_dir}' directory not found. Please place images inside.")
36  38
37  39    try:
       40 +     # 各種画像の読み込み。convert_alpha()で透明度を扱えるようにする
38  41        PLAYER_IMAGE = pygame.image.load(os.path.join(fig_dir,
          "koukaton.png")).convert_alpha()
39  42        ENEMY_IMAGE = pygame.image.load(os.path.join(fig_dir,
          "enemy.png")).convert_alpha()
40      -     # プレイヤーの弾も beam.png を読み込むようにする
41      -     PLAYER_BULLET_IMAGE = pygame.image.load(os.path.join(fig_dir,
          "beam.png")).convert_alpha()
42      -     ENEMY_BULLET_IMAGE = pygame.image.load(os.path.join(fig_dir,
          "beam.png")).convert_alpha()
43      -
44      -     # explosion.gif のアニメーションフレームを読み込む
45      -     EXPLOSION_FRAMES = []
```

```
46      -     # 例えば explosion_00.png, explosion_01.png, ... のように連番で保存されていると仮定
47      -     for i in range(10): # 例として10フレーム (explosion_00.png から explosion_09.png)
48      -         frame_filename = os.path.join(fig_dir, f"explosion_{i:02d}.png")
49      -         if os.path.exists(frame_filename):
50      -             EXPLOSION_FRAMES.append(pygame.image.load(frame_filename).convert_alpha())
51      -         else:
52      -             # フレームが見つからない場合、単一のgifファイルを読むか、エラーにする
53      -             # 今回は単一のgifファイルを読み込み、後でそれを使うようにフォールバックする
54      -             print(f"Warning: Explosion frame {frame_filename} not found. Trying to
          load explosion.gif as a single image.")
55      -             try:
56      -                 single_explosion_image = pygame.image.load(os.path.join(fig_dir,
          "explosion.gif")).convert_alpha()
57      -                 EXPLOSION_FRAMES = [single_explosion_image] # 単一フレームとしてセット
58      -             except pygame.error as gif_e:
59      -                 print(f"Error loading explosion.gif: {gif_e}")
60      -             break # 最初のフレームが見つからない時点でループを抜ける
61      -
62      -     if not EXPLOSION_FRAMES: # 何も読み込めなかった場合
63      -         print("Warning: No explosion images (frames or gif) found. Using a fallback
          red circle.")
64      -         # フォールバックとして赤い円を作成
65      -         fallback_image = pygame.Surface((60, 60), pygame.SRCALPHA)
66      -         pygame.draw.circle(fallback_image, RED, (30, 30), 30)
67      -         EXPLOSION_FRAMES = [fallback_image]
68      -
        43  +     PLAYER_BULLET_IMAGE = pygame.image.load(os.path.join(fig_dir,
              "beam.png")).convert_alpha()
        44  +     # 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）
        45  +     try: LAZER_IMAGE = pygame.image.load(os.path.join(fig_dir,
              "lazer.png")).convert_alpha()
        46  +     except pygame.error: print("Warning: lazer.png not found."); LAZER_IMAGE =
              pygame.Surface((20, SCREEN_HEIGHT)); LAZER_IMAGE.fill(CYAN);
              LAZER_IMAGE.set_colorkey(BLACK)
        47  +     try: HEAL_ITEM_IMAGE = pygame.image.load(os.path.join(fig_dir,
              "heal.png")).convert_alpha()
        48  +     except pygame.error: HEAL_ITEM_IMAGE = pygame.Surface((25, 25));
              HEAL_ITEM_IMAGE.fill(GREEN)
        49  +     try: ATTACK_ITEM_IMAGE = pygame.image.load(os.path.join(fig_dir,
              "attack.png")).convert_alpha()
        50  +     except pygame.error: ATTACK_ITEM_IMAGE = pygame.Surface((25, 25));
              ATTACK_ITEM_IMAGE.fill(YELLOW)
        51  +     try: EXPLOSION_IMAGE = pygame.image.load(os.path.join(fig_dir,
              "explosion.png")).convert_alpha()
        52  +     except pygame.error: print("Warning: explosion.png not found."); EXPLOSION_IMAGE =
              pygame.Surface((60, 60), pygame.SRCALPHA); pygame.draw.circle(EXPLOSION_IMAGE, RED,
              (30, 30), 30)
69      53    except pygame.error as e:
70      -     print(f"Error loading image: {e}")
71      -     if "cannot convert without pygame.display initialized" in str(e):
72      -         print("Pygame display was not initialized before image conversion.")
73      -     print("Make sure 'koukaton.png', 'enemy.png', 'beam.png' and explosion frames
          (e.g., 'explosion_00.png', 'explosion_01.png'...) or 'explosion.gif' exist in 'fig'
          folder.")
74      -     pygame.quit()
75      -     sys.exit()
        54  +     print(f"Error loading image: {e}"); pygame.quit(); sys.exit()
76      55
```

```
                                  56 + # --- クラス定義 ---
77                                57
78                                - # --- プレイヤー クラス (Player Class) ---
79                                58   class Player(pygame.sprite.Sprite):
                                  59 +     """プレイヤー機体を管理するクラス"""
80                                60       def __init__(self):
                                  61 +         """プレイヤーの初期設定を行う"""
81                                62           super().__init__()
82                                -            self.image = pygame.transform.scale(PLAYER_IMAGE, (40, 40))
83                                -            self.rect = self.image.get_rect()
84                                -            self.rect.centerx = SCREEN_WIDTH // 2
85                                -            self.rect.bottom = SCREEN_HEIGHT - 30
                                  63 +         self.image = pygame.transform.scale(PLAYER_IMAGE, (40, 40))
                                  64 +         self.rect = self.image.get_rect(centerx=SCREEN_WIDTH // 2,
                                     bottom=SCREEN_HEIGHT - 30)
86                                65           self.speed_x = 0
87                                -            self.shoot_delay = 200
88                                -            self.last_shot = pygame.time.get_ticks()
89                                -            self.hidden = False
                                  66 +         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
                                  67 +         # 体力関連
                                  68 +         self.max_health = 100
                                  69 +         self.health = self.max_health
                                  70 +         # 攻撃関連
                                  71 +         self.shoot_delay = 250 # 弾の発射間隔（ミリ秒）
                                  72 +         self.last_shot = pygame.time.get_ticks() # 最後に弾を撃った時刻
                                  73 +         self.powerup_level = 0 # 0:通常, 1:3方向, 2:レーザー
                                  74 +         self.powerup_duration = 7000 # パワーアップの持続時間（ミリ秒）
                                  75 +         self.powerup_end_time = 0 # パワーアップの終了時刻
                                  76 +         self.active_laser = None # 照射中のレーザーオブジェクトを保持
90                                77
91                                78       def update(self):
92                                -            if self.hidden:
93                                -                return
94                                -
                                  79 +         """プレイヤーの状態を毎フレーム更新する"""
                                  80 +         if self.hidden: return # 非表示中は更新しない
                                  81 +         # キー入力に応じた左右移動
95                                82           self.speed_x = 0
96                                83           keys = pygame.key.get_pressed()
97                                -            if keys[pygame.K_LEFT]:
98                                -                self.speed_x = -7
99                                -            if keys[pygame.K_RIGHT]:
100                               -                self.speed_x = 7
101                               -
                                  84 +         if keys[pygame.K_LEFT]: self.speed_x = -7
                                  85 +         if keys[pygame.K_RIGHT]: self.speed_x = 7
102                               86           self.rect.x += self.speed_x
103                               -
104                               -            if self.rect.right > SCREEN_WIDTH:
105                               -                self.rect.right = SCREEN_WIDTH
106                               -            if self.rect.left < 0:
107                               -                self.rect.left = 0
                                  87 +         # 画面端からはみ出ないように制御
                                  88 +         if self.rect.right > SCREEN_WIDTH: self.rect.right = SCREEN_WIDTH
                                  89 +         if self.rect.left < 0: self.rect.left = 0
                                  90 +         # パワーアップの持続時間が過ぎたら元に戻す
                                  91 +         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):
-       if self.hidden:
-           return
-
+       """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
-           bullet = PlayerBullet(self.rect.centerx, self.rect.top)
-           all_sprites.add(bullet)
-           bullets_group.add(bullet)
+           # 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, bullet3)
+           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()
+       self.kill()
```

```
-
- # --- 敵 クラス (Enemy Class) ---
  class Enemy(pygame.sprite.Sprite):
-     # player_ref を引数に追加
-     def __init__(self, speed_level=0, all_sprites_ref=None,
  enemy_bullets_group_ref=None, player_ref=None):
+     """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()
-         self.rect.x = random.randrange(0, SCREEN_WIDTH - self.rect.width)
-         self.rect.y = random.randrange(-100, -40)
-
-         base_speed_min = 2
-         base_speed_max = 5
-         speed_increase = speed_level * 0.4
-         min_speed = int(base_speed_min + speed_increase)
-         max_speed = int(base_speed_max + speed_increase)
-         if max_speed <= min_speed:
-             max_speed = min_speed + 1
-
-         self.speed_y = random.randrange(min_speed, max_speed)
-
-         self.all_sprites = all_sprites_ref
-         self.enemy_bullets_group = enemy_bullets_group_ref
-         self.enemy_shoot_delay = random.randrange(1000, 2500) # 1秒～2.5秒に変更
-         self.last_shot = pygame.time.get_ticks()
-         self.player = player_ref # プレイヤーへの参照を保存
-
-         self.health = 1
-         self.score_value = 1
+         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(-100, -40))
+         # 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:
+         if self.rect.top > SCREEN_HEIGHT + 10: # 画面外に出たら消える
              self.kill()

-         self.shoot()
-
-     def shoot(self):
-         now = pygame.time.get_ticks()
-         if now - self.last_shot > self.enemy_shoot_delay:
-             self.last_shot = now
-             # EnemyBullet に self.speed_y と self.player を渡す
```

```
166                 -            enemy_bullet = EnemyBullet(self.rect.centerx, self.rect.bottom,
                    self.speed_y, self.player)
167                 -            if self.all_sprites and self.enemy_bullets_group:
168                 -                self.all_sprites.add(enemy_bullet)
169                 -                self.enemy_bullets_group.add(enemy_bullet)
170                 -
171                 -    def hit(self):
172                 -        self.health -= 1
173                 -        if self.health <= 0:
174                 -            return True
175                 -        return False
176                 -
177                 - # --- プレイヤー弾 クラス (Player Bullet Class) ---
178 161               class PlayerBullet(pygame.sprite.Sprite):
179                 -    def __init__(self, x, y):
    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"""
180 165                   super().__init__()
181                 -        # beam.png をスケーリングし、上下反転させる
182                 -        # 元画像をリサイズ (敵の弾と合わせる)
183                 -        raw_image = pygame.transform.scale(PLAYER_BULLET_IMAGE, (15, 15))
184                 -        # Y軸(垂直)方向に反転させて上向きにする
185                 -        self.image = pygame.transform.flip(raw_image, False, True)
186                 -        self.rect = self.image.get_rect()
187                 -        self.rect.bottom = y
188                 -        self.rect.centerx = x
189                 -        self.speed_y = -10
    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）
190 170
191 171               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）"""
192 173                   self.rect.y += self.speed_y
193                 -        if self.rect.bottom < 0:
    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
194 176                       self.kill()
195 177
196                 - # --- 敵のビーム クラス (Enemy Bullet Class) ---
197                 - class EnemyBullet(pygame.sprite.Sprite):
198                 -     # enemy_speed_y と player_obj を引数に追加
199                 -     def __init__(self, x, y, enemy_speed_y, player_obj):
    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"""
200 182                   super().__init__()
201                 -        self.image = pygame.transform.scale(ENEMY_BULLET_IMAGE, (15, 15))
    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))
202 185                   self.rect = self.image.get_rect()
203                 -        self.rect.top = y
204                 -        self.rect.centerx = x
205                 -
206                 -        # --- プレイヤーへの方向と速度を計算 ---
```

```
207     -
208     -         # 1. ターゲット座標を設定
209     -         target_x, target_y = self.rect.centerx, self.rect.bottom + 100 # デフォルト（真下）
210     -         if player_obj and not player_obj.hidden:
211     -             # プレイヤーが有効なら、プレイヤーの中心を狙う
212     -             target_x = player_obj.rect.centerx
213     -             target_y = player_obj.rect.centery
214     -
215     -         # 2. 差分（ベクトル）を計算
216     -         dx = target_x - self.rect.centerx
217     -         dy = target_y - self.rect.centery
218     -
219     -         # 3. 距離を計算 (math.hypot は 0 除算を回避するのに便利)
220     -         distance = math.hypot(dx, dy)
221     -
222     -         # 4. (要件1) 速度を計算 (敵機の2.5倍、ただし最低速度は7)
223     -         total_speed = max(7.0, enemy_speed_y * 2.5)
224     -
225     -         # 5. (要件2) 速度ベクトルを計算
226     -         if distance == 0:
227     -             # ターゲットが真上（あり得ないが念のため）なら真下に撃つ
228     -             self.speed_x = 0
229     -             self.speed_y = total_speed
230     -         else:
231     -             # ベクトルを正規化（長さを1に）してから速度を掛ける
232     -             self.speed_x = (dx / distance) * total_speed
233     -             self.speed_y = (dy / distance) * total_speed
234     -         # --- 計算ここまで ---
        186 +         self.update() # 初期位置をプレイヤーに合わせる
235     187
236     188     def update(self):
237     -         # 浮動小数点数で座標を更新（より滑らかになる）
238     -         self.rect.x += self.speed_x
        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）"""
239     203             self.rect.y += self.speed_y
240     -         # 画面外に出たら削除
241     -         if self.rect.top > SCREEN_HEIGHT or self.rect.bottom < 0 or self.rect.left > SCREEN_WIDTH or self.rect.right < 0:
        204 +         if self.rect.top > SCREEN_HEIGHT: # 画面外に出たら消える
242     205             self.kill()
243     206
244     - # --- 爆発エフェクト クラス (Explosion Class) ---
        207 + class HealItem(Item):
```

```
                +     """回復アイテムのクラス"""
                +     def __init__(self, center):
                +         self.image = pygame.transform.scale(HEAL_ITEM_IMAGE, (30, 30))
                +         super().__init__(center) # 親クラスの初期化処理を呼び出す
                +
                +     def apply_effect(self, player):
                +         """取得した際の回復効果を適用する"""
                +         player.heal(25)
                +
                + 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()
                +
245             227   class Explosion(pygame.sprite.Sprite):
                +     """爆発エフェクトを管理するクラス"""
246             229       def __init__(self, center, size="normal"):
                +         """爆発の初期設定を行う"""
247             231           super().__init__()
248             -         self.frames = EXPLOSION_FRAMES
249             -
250             -         if size == "large":
251             -             # 大きな爆発の場合、フレームを大きくスケール
252             -             self.frames = [pygame.transform.scale(f, (90, 90)) for f in self.frames]
253             -             self.frame_rate = 100 # フレーム表示速度（ミリ秒）
254             -         else:
255             -             self.frames = [pygame.transform.scale(f, (60, 60)) for f in self.frames]
256             -             self.frame_rate = 70 # フレーム表示速度（ミリ秒）
257             -
258             -         self.current_frame = 0
259             -         self.image = self.frames[self.current_frame]
                +         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)
260             236           self.rect = self.image.get_rect(center=center)
261             -         self.last_update = pygame.time.get_ticks()
                +         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
262             239
263             240       def update(self):
264             -         now = pygame.time.get_ticks()
265             -         if now - self.last_update > self.frame_rate:
266             -             self.last_update = now
267             -             self.current_frame += 1
268             -             if self.current_frame >= len(self.frames):
269             -                 self.kill() # 全フレーム表示したら消滅
270             -             else:
271             -                 center = self.rect.center # 中心座標を保持
272             -                 self.image = self.frames[self.current_frame]
273             -                 self.rect = self.image.get_rect(center=center)
274             -
275             -
```

```
276                 - # --- 星（背景）の管理 (Star Background Management) ---
277                 - def create_stars(number):
278                 -     stars = []
279                 -     for _ in range(number):
280                 -         star_x = random.randrange(0, SCREEN_WIDTH)
281                 -         star_y = random.randrange(0, SCREEN_HEIGHT)
282                 -         star_speed = random.randrange(1, 4)
283                 -         star_size = random.randrange(1, 4)
284                 -         stars.append([star_x, star_y, star_speed, star_size])
285                 -     return stars
286                 -
287                 - def draw_stars(surface, stars, speed_level=0):
288                 -     speed_modifier = 1.0 + speed_level * 0.15
289                 -
        241         +         """毎フレーム、表示時間が過ぎたかチェックする"""
        242         +         if pygame.time.get_ticks() - self.creation_time > self.duration:
        243         +             self.kill() # 一定時間経ったら消える
        244         +
        245         + # --- 描画関数群 ---
        246         +
        247         + def draw_stars(surface, stars, speed_level=0):
        248         +     """背景の星を描画し、スクロールさせる"""
        249         +     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
290     250             for star in stars:
291     251                 pygame.draw.circle(surface, WHITE, (star[0], star[1]), star[3])
292     252                 star[1] += star[2] * speed_modifier
293                 -
294                 -         if star[1] > SCREEN_HEIGHT:
295                 -             star[1] = 0
296                 -             star[0] = random.randrange(0, SCREEN_WIDTH)
        253         +         if star[1] > SCREEN_HEIGHT: # 画面下に出たら上に戻す
        254         +             star[0] = random.randrange(0, SCREEN_WIDTH); star[1] = 0
297     255
298                 - # --- テキスト描画用のヘルパー関数 (Helper function for drawing text) ---
299     256         def draw_text(surface, text, font, color, x, y, align="topright"):
300                 -     text_surface = font.render(text, True, color)
        257         +     """指定された位置にテキストを描画する"""
        258         +     text_surface = font.render(text, True, color)
301     259             text_rect = text_surface.get_rect()
302                 -     if align == "topright":
303                 -         text_rect.topright = (x, y)
304                 -     elif align == "center":
305                 -         text_rect.center = (x, y)
306                 -     elif align == "topleft":
307                 -         text_rect.topleft = (x, y)
        260         +     if align == "topright": text_rect.topright = (x, y)
        261         +     elif align == "center": text_rect.center = (x, y)
        262         +     elif align == "topleft": text_rect.topleft = (x, y)
308     263             surface.blit(text_surface, text_rect)
309     264
310                 - # --- フォントの設定 ---
311                 - score_font = pygame.font.SysFont(None, 36)
312                 - game_over_font = pygame.font.SysFont(None, 64, bold=True)
313                 -
314                 - # --- 背景用の星を作成 ---
315                 - stars = create_stars(100)
316                 -
```

```
317                - # --- スプライトグループの作成 (Sprite Groups) ---
318                - all_sprites = pygame.sprite.Group()
319                - enemies_group = pygame.sprite.Group()
320                - player_bullets_group = pygame.sprite.Group()
321                - enemy_bullets_group = pygame.sprite.Group()
322                - 
323                - # --- プレイヤーの作成 ---
                   265 + def draw_health_bar(surface, x, y, pct):
                   266 +     """プレイヤーの体力バーを描画する"""
                   267 +     pct = max(0, pct)
                   268 +     BAR_LENGTH, BAR_HEIGHT = 150, 15
                   269 +     fill = (pct / 100) * BAR_LENGTH
                   270 +     bar_color = GREEN if pct > 60 else YELLOW if pct > 30 else RED # 体力に応じて色を変
                         える
                   271 +     pygame.draw.rect(surface, bar_color, (x, y, fill, BAR_HEIGHT))
                   272 +     pygame.draw.rect(surface, WHITE, (x, y, BAR_LENGTH, BAR_HEIGHT), 2) # 白い枠線
                   273 + 
                   274 + # --- ゲームのセットアップ ---
                   275 + # フォントの準備
                   276 + score_font = pygame.font.SysFont(None, 36)
                   277 + game_over_font = pygame.font.SysFont(None, 64, bold=True)
                   278 + info_font = pygame.font.SysFont(None, 30)
                   279 + # 背景の星をランダムに生成
                   280 + stars =
                         [[random.randrange(0,SCREEN_WIDTH),random.randrange(0,SCREEN_HEIGHT),random.randrange(
                         1,4),random.randrange(1,4)] for _ in range(100)]
                   281 + # スプライトを管理するためのグループを作成
                   282 + all_sprites = pygame.sprite.Group() # 全てのスプライト（描画・更新用）
                   283 + enemies_group = pygame.sprite.Group() # 敵（衝突判定用）
                   284 + player_bullets_group = pygame.sprite.Group() # プレイヤーの弾（衝突判定用）
                   285 + items_group = pygame.sprite.Group() # アイテム（衝突判定用）
                   286 + laser_group = pygame.sprite.Group() # レーザー（衝突判定用）
                   287 + # プレイヤーインスタンスを生成し、グループに追加
324                288   player = Player()
325                289   all_sprites.add(player)
326                - 
327                - # --- 敵を定期的に生成するためのカスタムイベント ---
                   290 + # 敵を定期的に生成するためのカスタムイベントを設定
328                291   ADD_ENEMY = pygame.USEREVENT + 1
329                - initial_spawn_rate = 1000
330                - current_spawn_rate = initial_spawn_rate
331                - pygame.time.set_timer(ADD_ENEMY, current_spawn_rate)
332                - 
333                - # --- スコアとゲームレベル ---
334                - score = 0
335                - game_speed_level = 0
336                - game_over = False
                   292 + pygame.time.set_timer(ADD_ENEMY, 1000) # 1000ミリ秒(1秒)ごとに発生
                   293 + # ゲームの状態を管理する変数を初期化
                   294 + score, game_speed_level, game_over, running = 0, 0, False, True
337                295 
338                296   # --- メインゲームループ (Main Game Loop) ---
339                - running = True
340                297   while running:
341                -     # 1. フレームレートの制御 (Control frame rate)
                   298 +     # フレームレートの制御
342                299       clock.tick(FPS)
343                300 
```

```
344     -     # 2. イベント処理 (Event handling)
    301 +     # --- イベント処理 ---
345 302       for event in pygame.event.get():
    303 +         # ウィンドウの×ボタンが押されたら終了
346 304           if event.type == pygame.QUIT:
347 305               running = False
    306 +         # ゲームオーバー中に、何かキーが押されたら終了
    307 +         elif game_over and event.type == pygame.KEYDOWN:
    308 +             running = False
    309 +         # ADD_ENEMYイベントが発生したら、新しい敵を生成
348 310           elif event.type == ADD_ENEMY and not game_over:
349     -             # Enemy生成時に player オブジェクトを渡す
350     -             new_enemy = Enemy(game_speed_level, all_sprites, enemy_bullets_group,
      player)
351     -             all_sprites.add(new_enemy)
352     -             enemies_group.add(new_enemy)
    311 +             enemy = Enemy(game_speed_level, player)
    312 +             all_sprites.add(enemy)
    313 +             enemies_group.add(enemy)
353 314
354     -     # 射撃（スペースキーが押され続けているかチェック）
    315 +     # --- 操作処理 ---
355 316       keys = pygame.key.get_pressed()
    317 +     # スペースキーが押されていたら弾を発射
356 318       if keys[pygame.K_SPACE] and not game_over:
357 319           player.shoot(all_sprites, player_bullets_group)
358     -
359     -     # 3. 更新 (Update)
    320 +
    321 +     # 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）
    322 +     if player.powerup_level >= 2 and not game_over:
    323 +         if keys[pygame.K_SPACE]:
    324 +             if not player.active_laser: # レーザーがなければ生成
    325 +                 player.active_laser = SuperLaser(player)
    326 +                 all_sprites.add(player.active_laser);
      laser_group.add(player.active_laser)
    327 +         else: # スペースを離したら消す
    328 +             if player.active_laser:
    329 +                 player.active_laser.kill(); player.active_laser = None
    330 +     else: # パワーアップ中でなければ消す
    331 +         if player.active_laser:
    332 +             player.active_laser.kill(); player.active_laser = None
    333 +
    334 +     # --- 更新処理 ---
360 335       if not game_over:
361     -         all_sprites.update()
362     -     else:
363     -         # ゲームオーバー後も爆発エフェクトは更新する
364     -         # explosionオブジェクトがkillされるまでupdateを続ける
365     -         for sprite in all_sprites:
366     -             if isinstance(sprite, Explosion):
367     -                 sprite.update()
368     -
369     -     # 4. 衝突判定 (Collision Detection)
    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):
```

```
                s.update()

    # --- 衝突判定 ---
    if not game_over:
-
-       enemies_destroyed_this_frame = 0
-
-       # プレイヤーの弾と敵の衝突
-       hits_normal = pygame.sprite.groupcollide(player_bullets_group, enemies_group, True, False)
-       for bullet, enemies_hit in hits_normal.items():
-           for enemy_hit in enemies_hit:
-               if enemy_hit.hit():
-                   explosion = Explosion(enemy_hit.rect.center, "normal")
-                   all_sprites.add(explosion)
-                   score += enemy_hit.score_value
-                   enemies_destroyed_this_frame += 1
-                   enemy_hit.kill()
-
-       # レベルアップ処理
-       if enemies_destroyed_this_frame > 0:
-           new_speed_level = score // 10
-           if new_speed_level > game_speed_level:
-               game_speed_level = new_speed_level
-               print(f"--- SPEED LEVEL UP! Level: {game_speed_level} ---")
-
-               current_spawn_rate = max(150, int(initial_spawn_rate * (0.9 ** game_speed_level)))
-               pygame.time.set_timer(ADD_ENEMY, 0)
-               pygame.time.set_timer(ADD_ENEMY, current_spawn_rate)
-               print(f"New Spawn Rate: {current_spawn_rate} ms")
+       # プレイヤーの弾/レーザーと敵の衝突
+       # groupcollideで衝突したペアを検出し、敵と弾を消す
+       hits_bullet = pygame.sprite.groupcollide(player_bullets_group, enemies_group, True, True)
+       hits_laser = pygame.sprite.groupcollide(laser_group, enemies_group, False, True) # レーザーは消えない
+       hits_bullet.update(hits_laser) # ２つの衝突結果をマージする
+
+       # 衝突した後処理（スコア加算、爆発、アイテムドロップ）
+       for weapon, enemies_hit in hits_bullet.items():
+           for enemy in enemies_hit:
+               score += enemy.score_value
+               all_sprites.add(Explosion(enemy.rect.center, "normal"))
+               # 20%の確率でアイテムをドロップ
+               if random.random() > 0.8:
+                   item = random.choice([HealItem, AttackUpItem])(enemy.rect.center)
+                   all_sprites.add(item); items_group.add(item)
+
+       # 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）
+       new_level = score // 10
+       if new_level > game_speed_level:
+           game_speed_level = new_level
+           # 敵の出現間隔を短くする
+           rate = max(150, int(1000 * (0.9 ** game_speed_level)))
+           pygame.time.set_timer(ADD_ENEMY, rate)

        # プレイヤーと敵の衝突
```

```
398     -         player_enemy_hits = pygame.sprite.spritecollide(player, enemies_group, True)
    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
399     381
400     -         if player_enemy_hits:
401     -             explosion = Explosion(player.rect.center, "large")
402     -             all_sprites.add(explosion)
403     -             player.hide()
404     -             game_over = True
405     -             print("Game Over! (Collided with enemy)")
406     -             pygame.time.set_timer(ADD_ENEMY, 0)
407     -
408     -         # プレイヤーと敵のビームの衝突
409     -         player_beam_hits = pygame.sprite.spritecollide(player, enemy_bullets_group,
            True)
410     -         if player_beam_hits:
411     -             explosion = Explosion(player.rect.center, "normal")
412     -             all_sprites.add(explosion)
413     -             player.hide()
414     -             game_over = True
415     -             print("Game Over! (Hit by enemy beam)")
416     -             pygame.time.set_timer(ADD_ENEMY, 0)
417     -
418     -
419     -     # 5. 描画 (Draw / Render)
420     -     screen.fill(BLACK)
421     -     draw_stars(screen, stars, game_speed_level)
422     -     all_sprites.draw(screen)
423     -
424     -     # スコアを描画
425     -     draw_text(screen, f"SCORE: {score}", score_font, WHITE, SCREEN_WIDTH - 10, 10,
            align="topright")
    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)
426     391
427     -     # レベルを描画
428     -     draw_text(screen, f"LEVEL: {game_speed_level}", score_font, WHITE, 10, 10,
            align="topleft")
429     -
```

```
430     -     # ゲームオーバー表示
    392 +     # ゲームオーバー画面の表示
431 393       if game_over:
432     -         draw_text(screen, "GAME OVER", game_over_font, RED, SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2, align="center")
433     -
434     -     # 6. 画面のフリップ (Flip display)
    394 +         draw_text(screen, "GAME OVER", game_over_font, RED, SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2, "center")
    395 +         draw_text(screen, "Press any key to exit", info_font, WHITE, SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2 + 50, "center")
    396 +
    397 +     # 描画内容を画面に反映
435 398       pygame.display.flip()
436 399
437     - # --- 終了処理 (Exit) ---
    400 + # --- 終了処理 ---
438 401   pygame.quit()
439     - sys.exit()
440     -
    402 + sys.exit()
```