



c0c2400957 / ProjExD_Group15



Code

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

ProjExD_Group15

/ README.md

in c0c24009/item_drop

Cancel changes

Commit changes...

Edit

Preview

 Show Diff

スペースインベーダーらしきもの

実行環境の必要条件

- python >= 3.10
- pygame >= 2.1

ゲームの概要

- 主人公の戦闘機が、迫りくる敵機をビームで粉碎しながら突き進む縦スクロールシューティングゲーム。
- 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。

ゲームの遊び方

- 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。
- 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。
- 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。

ゲームの実装

共通基本機能

- 背景（星空）のスクロールと、自機および敵機の描画
- プレイヤーの操作（左右移動、弾の発射）
- 敵機の自動生成と画面内での動き

分担追加機能

- 障害物の出現（担当：ソエタ）

- 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。
- 回避などの特殊コマンド（担当：かつみ）
- 大ボス（担当：のうち）
- 中ボス（担当：オウ）

ToDo

- 敵機の攻撃（ビーム発射）機能の実装
- 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）

メモ

- 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設計してある
- すべてのクラスに関する関数は、クラスの外で定義してある
- でっかいボス
- ライフ制
- アイテム降らす
- 中ボス、大ボス
- ナインイド選択
- 障害物



c0c2400957 / ProjExD_Group15



Code

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

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



```
1 import pygame
2 import sys
3 import random
4 import os
5 import math
6
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
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
11 fig_dir = os.path.join(script_dir, "fig")
12
13 # --- 定数 (Constants) ---
14 SCREEN_WIDTH = 600
15 SCREEN_HEIGHT = 800
16 FPS = 60
17
18 # 色 (Colors)
19 WHITE = (255, 255, 255)
20 BLACK = (0, 0, 0)
21 RED = (255, 0, 0)
22 YELLOW = (255, 255, 0)
23 GREEN = (0, 255, 0)
24 CYAN = (0, 255, 255)
25
26 # --- ゲームの初期化 (Game Initialization) ---
27 pygame.init()
28 pygame.font.init() # フォントモジュールを初期化
29 screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
30 pygame.display.set_caption("Xevious Style Shooter")
31 clock = pygame.time.Clock() # FPSを管理するためのClockオブジェクト
32
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
35 if not os.path.exists(fig_dir):
36     os.makedirs(fig_dir)
37     print(f"Warning: '{fig_dir}' directory not found. Please place images inside.")
38
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
41     PLAYER_IMAGE = pygame.image.load(os.path.join(fig_dir, "koukaton.png")).convert_alpha()
42     ENEMY_IMAGE = pygame.image.load(os.path.join(fig_dir, "enemy.png")).convert_alpha()
```

```
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, SCR
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(
53     except pygame.error as e:
54         print(f"Error loading image: {e}"); pygame.quit(); sys.exit()
55
56     # --- クラス定義 ---
57
58     class Player(pygame.sprite.Sprite):
59         """プレイヤー機体を管理するクラス"""
60         def __init__(self):
61             """プレイヤーの初期設定を行う"""
62             super().__init__()
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)
65             self.speed_x = 0
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
77
78         def update(self):
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
82             self.speed_x = 0
83             keys = pygame.key.get_pressed()
84             if keys[pygame.K_LEFT]: self.speed_x = -7
85             if keys[pygame.K_RIGHT]: self.speed_x = 7
86             self.rect.x += self.speed_x
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:
92                 self.powerup_level = 0
93                 if self.active_laser: # レーザーが出ていれば消す
94                     self.active_laser.kill()
95                     self.active_laser = None
96                     print("Power-up ended.")
97
98         def shoot(self, all_sprites, bullets_group):
99             """弾を発射する"""
100            if self.hidden: return
```

```
101     now = pygame.time.get_ticks()
102     # 発射間隔(shoot_delay)を守って発射
103     if now - self.last_shot > self.shoot_delay:
104         self.last_shot = now
105         # パワーアップレベルに応じて弾の種類を変える
106         if self.powerup_level == 0: # 通常弾
107             bullet = PlayerBullet(self.rect.centerx, self.rect.top)
108             all_sprites.add(bullet); bullets_group.add(bullet)
109         elif self.powerup_level == 1: # 3方向弾
110             bullet1 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=0)
111             bullet2 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=-3)
112             bullet3 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=3)
113             all_sprites.add(bullet1, bullet2, bullet3); bullets_group.add(bullet1, bullet2, bullet3)
114         elif self.powerup_level >= 2: # レーザー照射中はサイドの弾だけ発射
115             bullet2 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=-4)
116             bullet3 = PlayerBullet(self.rect.centerx, self.rect.top, speed_x=4)
117             all_sprites.add(bullet2, bullet3); bullets_group.add(bullet2, bullet3)
118
119     def take_damage(self, amount):
120         """ダメージを受ける処理"""
121         self.health -= amount
122         self.health = max(0, self.health) # 体力がマイナスにならないように
123         return self.health <= 0 # 体力が0以下になったらTrueを返す
124
125     def heal(self, amount):
126         """体力を回復する処理"""
127         self.health += amount
128         self.health = min(self.health, self.max_health) # 最大体力を超えないように
129
130     def power_up(self):
131         """攻撃をパワーアップさせる処理"""
132         self.powerup_level = min(2, self.powerup_level + 1) # 最大レベル2まで
133         self.powerup_end_time = pygame.time.get_ticks() + self.powerup_duration # 終了タイマーをリセット
134         print(f"Power-up! Level {self.powerup_level}")
135
136     def hide(self):
137         """プレイヤーを非表示にする（ゲームオーバー時）"""
138         self.hidden = True
139         self.kill()
140
141     class Enemy(pygame.sprite.Sprite):
142         """敵機を管理するクラス"""
143         def __init__(self, speed_level=0, player_ref=None):
144             """敵の初期設定を行う"""
145             super().__init__()
146             self.image = pygame.transform.scale(ENEMY_IMAGE, (40, 40))
147             self.rect = self.image.get_rect(x=random.randrange(0, SCREEN_WIDTH - 40), y=random.randrange(-
148             # ゲームレベルに応じて落下速度を上げる
149             speed_increase = speed_level * 0.4
150             self.speed_y = random.randrange(int(2 + speed_increase), int(5 + speed_increase) + 1)
151             self.player = player_ref
152             self.health = 1
153             self.score_value = 1
154
155         def update(self):
156             """敵の状態を毎フレーム更新する（下に移動）"""
157             self.rect.y += self.speed_y
158             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
```

```
217    class AttackUpItem(Item):
218        """攻撃力アップアイテムのクラス"""
219        def __init__(self, center):
220            self.image = pygame.transform.scale(ATTACK_ITEM_IMAGE, (30, 30))
221            super().__init__(center)
222
223        def apply_effect(self, player):
224            """取得した際のパワーアップ効果を適用する"""
225            player.power_up()
226
227    class Explosion(pygame.sprite.Sprite):
228        """爆発エフェクトを管理するクラス"""
229        def __init__(self, center, size="normal"):
230            """爆発の初期設定を行う"""
231            super().__init__()
232            self.original_image = EXPLOSION_IMAGE
233            # "large"か"normal"かでサイズを変える
234            scale = (90, 90) if size == "large" else (60, 60)
235            self.image = pygame.transform.scale(self.original_image, scale)
236            self.rect = self.image.get_rect(center=center)
237            self.duration = 400 # 表示時間 (ミリ秒)
238            self.creation_time = pygame.time.get_ticks() # 生成された時刻を記録
239
240        def update(self):
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() # 一定時間経つたら消える
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
250        for star in stars:
251            pygame.draw.circle(surface, WHITE, (star[0], star[1]), star[3])
252            star[1] += star[2] * speed_modifier
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
255
256    def draw_text(surface, text, font, color, x, y, align="topright"):
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)
259        text_rect = text_surface.get_rect()
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)
263        surface.blit(text_surface, text_rect)
264
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(1,10)] for i 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
288     player = Player()
289     all_sprites.add(player)
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
291     ADD_ENEMY = pygame.USEREVENT + 1
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
295
296     # --- メインゲームループ (Main Game Loop) ---
297     while running:
298         # フレームレートの制御
299         clock.tick(FPS)
300
301         # --- イベント処理 ---
302         for event in pygame.event.get():
303             # ウィンドウのxボタンが押されたら終了
304             if event.type == pygame.QUIT:
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
310             elif event.type == ADD_ENEMY and not game_over:
311                 enemy = Enemy(game_speed_level, player)
312                 all_sprites.add(enemy)
313                 enemies_group.add(enemy)
314
315         # --- 操作処理 ---
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
318         if keys[pygame.K_SPACE] and not game_over:
319             player.shoot(all_sprites, player_bullets_group)
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 ---
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) # レーザーは消す
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, "center")
395         draw_text(screen, "Press any key to exit", info_font, WHITE, SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2, "center")
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()
```



c0a24147d3 / ProjExD_Group15



Code

Issues

Pull requests

Actions

Projects

Security

Insights

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](#)

-o 3 commits

6 files changed

2 contributors

-o 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.

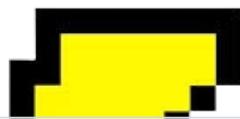
[Split](#) [Unified](#)

▼ ⌂ 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碎し突き進むゲーム
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碎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	18	- * 背景画像とメイン機体、敵機の描画
	19	+ * 背景（星空）のスクロールと、自機および敵機の描画
	20	+ * プレイヤーの操作（左右移動、弾の発射）
		+ * 敵機の自動生成と画面内での動き
18	21	
19	21	- *** 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	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ります。**
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	29	- - [] ほげほげ機能
28		- - [] ふがふが関数内の変数名の統一
30	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いる
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いる
33	35	- *
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	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
6   6   # --- スクリプトのパスを基準にディレクトリを設定 ---
7   7   + # 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   8   script_dir = os.path.dirname(os.path.abspath(__file__))
9   9   + # 画像が保存されている'fig'フォルダへのパスを作成
10 10  fig_dir = os.path.join(script_dir, "fig")
11 11  # --- 定数 (Constants) ---
12 12  # 色 (Colors)
13 13  WHITE = (255, 255, 255)
14 14  BLACK = (0, 0, 0)
15 15  - RED = (255, 0, 0)
16 16  - YELLOW = (255, 255, 0)
17 17  + RED = (255, 0, 0)
18 18  + YELLOW = (255, 255, 0)
19 19  + GREEN = (0, 255, 0)
20 20  + CYAN = (0, 255, 255)
21 21  # --- ゲームの初期化 (Game Initialization) ---
22 22  - # ★★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★★
23 23  pygame.init()
24 24  - pygame.font.init()
25 25  + pygame.font.init() # フォントモジュールを初期化
26 26  screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
27 27  pygame.display.set_caption("Xevious Style Shooter")
28 28  - clock = pygame.time.Clock()
29 29  -
30 30  + clock = pygame.time.Clock() # FPSを管理するためのClockオブジェクト
31 31  # --- 画像ファイルの読み込み (Load Images) ---
32 32  + # 'fig' フォルダが存在しない場合は作成する
33 33  if not os.path.exists(fig_dir):
34 34      os.makedirs(fig_dir)
35 35  -     print(f"Warning: '{fig_dir}' directory not found. Created an empty one.")
36 36  -     print("Please place 'koukaton.png', 'enemy.png', 'beam.png' and explosion frames
37 37  +     print(f"Warning: '{fig_dir}' directory not found. Please place images inside.")
38 38
39 39  try:
40 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
41 41      PLAYER_IMAGE = pygame.image.load(os.path.join(fig_dir,
42 42          "koukaton.png")).convert_alpha()
43 43      ENEMY_IMAGE = pygame.image.load(os.path.join(fig_dir,
44 44          "enemy.png")).convert_alpha()
45 45  -     # プレイヤーの弾も beam.png を読み込むようにする
46 46  -     PLAYER_BULLET_IMAGE = pygame.image.load(os.path.join(fig_dir,
47 47          "beam.png")).convert_alpha()
48 48  -     ENEMY_BULLET_IMAGE = pygame.image.load(os.path.join(fig_dir,
49 49          "beam.png")).convert_alpha()
50 50  -
51 51  -     # explosion.gif のアニメーションフレームを読み込む
52 52  -     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
55     - load explosion.gif as a single image.")
56     -             try:
57     -                 single_explosion_image = pygame.image.load(os.path.join(fig_dir,
58     - "explosion.gif")).convert_alpha()
59     -                 EXPLOSION_FRAMES = [single_explosion_image] # 単一フレームとしてセット
60     -             except pygame.error as gif_e:
61     -                 print(f"Error loading explosion.gif: {gif_e}")
62     -             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
63
64     -     if not EXPLOSION_FRAMES: # 何も読み込めなかった場合
65     -         print("Warning: No explosion images (frames or gif) found. Using a fallback
66     - red circle.")
67     -         # フォールバックとして赤い円を作成
68     -         fallback_image = pygame.Surface((60, 60), pygame.SRCALPHA)
69     -         pygame.draw.circle(fallback_image, RED, (30, 30), 30)
70     -         EXPLOSION_FRAMES = [fallback_image]
71
72
73
74
75
76
```

```
43     +     PLAYER_BULLET_IMAGE = pygame.image.load(os.path.join(fig_dir,
74     - "beam.png")).convert_alpha()
75     +     # 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）
76     +     try: LAZER_IMAGE = pygame.image.load(os.path.join(fig_dir,
77     - "lazer.png")).convert_alpha()
78     +     except pygame.error: print("Warning: lazer.png not found."); LAZER_IMAGE =
79     -     pygame.Surface((20, SCREEN_HEIGHT)); LAZER_IMAGE.fill(CYAN);
80     -     LAZER_IMAGE.set_colorkey(BLACK)
81     +     try: HEAL_ITEM_IMAGE = pygame.image.load(os.path.join(fig_dir,
82     - "heal.png")).convert_alpha()
83     +     except pygame.error: HEAL_ITEM_IMAGE = pygame.Surface((25, 25));
84     -     HEAL_ITEM_IMAGE.fill(GREEN)
85     +     try: ATTACK_ITEM_IMAGE = pygame.image.load(os.path.join(fig_dir,
86     - "attack.png")).convert_alpha()
87     +     except pygame.error: ATTACK_ITEM_IMAGE = pygame.Surface((25, 25));
88     -     ATTACK_ITEM_IMAGE.fill(YELLOW)
89     +     try: EXPLOSION_IMAGE = pygame.image.load(os.path.join(fig_dir,
90     - "explosion.png")).convert_alpha()
91     +     except pygame.error: print("Warning: explosion.png not found."); EXPLOSION_IMAGE =
92     -     pygame.Surface((60, 60), pygame.SRCALPHA); pygame.draw.circle(EXPLOSION_IMAGE, RED,
93     - (30, 30), 30)
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
```

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

```
92 +         self.powerup_level = 0
93 +         if self.active_laser: # レーザーが出ていれば消す
94 +             self.active_laser.kill()
95 +             self.active_laser = None
96 +             print("Power-up ended.")

108 97
109 98     def shoot(self, all_sprites, bullets_group):
110 -         if self.hidden:
111 -             return
112 -
113 99     """弾を発射する"""
114 100    if self.hidden: return
115 101    now = pygame.time.get_ticks()
116 102    # 発射間隔(shoot_delay)を守って発射
117 103    if now - self.last_shot > self.shoot_delay:
118 104        self.last_shot = now
119 105    bullet = PlayerBullet(self.rect.centerx, self.rect.top)
120 106    all_sprites.add(bullet)
121 107    bullets_group.add(bullet)

122 108
123 109
124 110
125 111
126 112
127 113
128 114
129 115
130 116
131 117
132 118
133 119
134 120
135 121
136 122
137 123
138 139
139 140

92 +         self.powerup_level = 0
93 +         if self.active_laser: # レーザーが出ていれば消す
94 +             self.active_laser.kill()
95 +             self.active_laser = None
96 +             print("Power-up ended.")

108 97
109 98     def shoot(self, all_sprites, bullets_group):
110 -         if self.hidden:
111 -             return
112 -
113 99     """弾を発射する"""
114 100    if self.hidden: return
115 101    now = pygame.time.get_ticks()
116 102    # 発射間隔(shoot_delay)を守って発射
117 103    if now - self.last_shot > self.shoot_delay:
118 104        self.last_shot = now
119 105    bullet = PlayerBullet(self.rect.centerx, self.rect.top)
120 106    all_sprites.add(bullet)
121 107    bullets_group.add(bullet)

122 108
123 109
124 110
125 111
126 112
127 113
128 114
129 115
130 116
131 117
132 118
133 119
134 120
135 121
136 122
137 123
138 139
139 140

# 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()
```

```
124 |          -
125 |          - # --- 敵 クラス (Enemy Class) ---
126 | 141   class Enemy(pygame.sprite.Sprite):
127 |          -     # player_ref を引数に追加
128 |          -     def __init__(self, speed_level=0, all_sprites_ref=None,
129 |                           enemy_bullets_group_ref=None, player_ref=None):
130 |          -         """敵機を管理するクラス"""
131 |          -         def __init__(self, speed_level=0, player_ref=None):
132 |          -             """敵の初期設定を行う"""
133 |          -             super().__init__()
134 |          -             self.image = pygame.transform.scale(ENEMY_IMAGE, (40, 40))
135 |          -             self.rect = self.image.get_rect()
136 |          -             self.rect.x = random.randrange(0, SCREEN_WIDTH - self.rect.width)
137 |          -             self.rect.y = random.randrange(-100, -40)
138 |          -             base_speed_min = 2
139 |          -             base_speed_max = 5
140 |          -             speed_increase = speed_level * 0.4
141 |          -             min_speed = int(base_speed_min + speed_increase)
142 |          -             max_speed = int(base_speed_max + speed_increase)
143 |          -             if max_speed <= min_speed:
144 |          -                 max_speed = min_speed + 1
145 |          -             self.speed_y = random.randrange(min_speed, max_speed)
146 |          -             self.all_sprites = all_sprites_ref
147 |          -             self.enemy_bullets_group = enemy_bullets_group_ref
148 |          -             self.enemy_shoot_delay = random.randrange(1000, 2500) # 1秒～2.5秒に変更
149 |          -             self.last_shot = pygame.time.get_ticks()
150 |          -             self.player = player_ref # プレイヤーへの参照を保存
151 |          -             self.health = 1
152 |          -             self.score_value = 1
153 | 146   +
154 | 147   +     self.image = pygame.transform.scale(ENEMY_IMAGE, (40, 40))
155 | 148   +     self.rect = self.image.get_rect(x=random.randrange(0, SCREEN_WIDTH - 40),
156 |                                         y=random.randrange(-100, -40))
157 | 149   +         # ゲームレベルに応じて落下速度を上げる
158 | 150   +         speed_increase = speed_level * 0.4
159 | 151   +         self.speed_y = random.randrange(int(2 + speed_increase), int(5 +
160 |                                         speed_increase) + 1)
161 | 152   +         self.player = player_ref
162 | 153   +         self.health = 1
163 | 154   +         self.score_value = 1
164 | 155   def update(self):
165 | 156   +             """敵の状態を毎フレーム更新する（下に移動）"""
166 | 157   +             self.rect.y += self.speed_y
167 | 158   -             if self.rect.top > SCREEN_HEIGHT + 10:
168 | 159   +                 if self.rect.top > SCREEN_HEIGHT + 10: # 画面外に出たら消える
169 | 160   +                     self.kill()
170 | 161   -             self.shoot()
171 | 162   -             def shoot(self):
172 | 163   -                 now = pygame.time.get_ticks()
173 | 164   -                 if now - self.last_shot > self.enemy_shoot_delay:
174 | 165   -                     self.last_shot = now
175 | 166   -                     # EnemyBullet に self.speed_y と self.player を渡す
```

```

166             enemy_bullet = EnemyBullet(self.rect.centerx, self.rect.bottom,
167                                         self.speed_y, self.player)
168             if self.all_sprites and self.enemy_bullets_group:
169                 self.all_sprites.add(enemy_bullet)
170                 self.enemy_bullets_group.add(enemy_bullet)
171
172     def hit(self):
173         self.health -= 1
174         if self.health <= 0:
175             return True
176         return False
177
178 # --- プレイヤー弾 クラス (Player Bullet Class) ---
179 class PlayerBullet(pygame.sprite.Sprite):
180
181     def __init__(self, x, y):
182
183         """プレイヤーの弾を管理するクラス"""
184
185         def __init__(self, x, y, speed_x=0):
186
187             """弾の初期設定を行う"""
188
189             super().__init__()
190
191             # beam.png をスケーリングし、上下反転させる
192             # 元画像をリサイズ（敵の弾と合わせる）
193             raw_image = pygame.transform.scale(PLAYER_BULLET_IMAGE, (15, 15))
194             # Y軸（垂直）方向に反転させて上向きにする
195             self.image = pygame.transform.flip(raw_image, False, True)
196             self.rect = self.image.get_rect()
197             self.rect.bottom = y
198             self.rect.centerx = x
199             self.speed_y = -10
200
201             self.image = pygame.transform.scale(PLAYER_BULLET_IMAGE, (10, 25))
202             self.rect = self.image.get_rect(center=(x, y))
203             self.speed_y = -10 # 上方向への速度
204             self.speed_x = speed_x # 横方向への速度（3方向弾用）
205
206     def update(self):
207
208         """弾の状態を毎フレーム更新する（指定された速度で移動）"""
209
210         self.rect.y += self.speed_y
211
212         if self.rect.bottom < 0:
213
214             self.rect.x += self.speed_x
215
216             if not screen.get_rect().colliderect(self.rect): # 画面外に出たら消える
217
218                 self.kill()
219
220     # --- 敵のビーム クラス (Enemy Bullet Class) ---
221
222     class EnemyBullet(pygame.sprite.Sprite):
223
224         # enemy_speed_y と player_obj を引数に追加
225
226         def __init__(self, x, y, enemy_speed_y, player_obj):
227
228             class SuperLaser(pygame.sprite.Sprite):
229
230                 """照射式レーザーを管理するクラス"""
231
232                 def __init__(self, player_obj):
233
234                     """レーザーの初期設定を行う"""
235
236                     super().__init__()
237
238                     self.image = pygame.transform.scale(ENEMY_BULLET_IMAGE, (15, 15))
239
240                     self.player = player_obj # プレイヤーへの参照を保持
241
242                     self.image = pygame.transform.scale(LAZER_IMAGE, (20, SCREEN_HEIGHT))
243
244                     self.rect = self.image.get_rect()
245
246                     self.rect.top = y
247
248                     self.rect.centerx = x
249
250
251                     # --- 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
210      -    下）
211      -    if player_obj and not player_obj.hidden:
212      -        # プレイヤーが有効なら、プレイヤーの中心を狙う
213      -        target_x = player_obj.rect.centerx
214      -        target_y = player_obj.rect.centery
215      -
216      -    # 2. 差分（ベクトル）を計算
217      -    dx = target_x - self.rect.centerx
218      -    dy = target_y - self.rect.centery
219      -
220      -    # 3. 距離を計算（math.hypot は 0 除算を回避するのに便利）
221      -    distance = math.hypot(dx, dy)
222      -
223      -    # 4. (要件1) 速度を計算（敵機の2.5倍、ただし最低速度は7）
224      -    total_speed = max(7.0, enemy_speed_y * 2.5)
225      -
226      -    # 5. (要件2) 速度ベクトルを計算
227      -    if distance == 0:
228      -        # 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
229      -        self.speed_x = 0
230      -        self.speed_y = total_speed
231      -    else:
232      -        # ベクトルを正規化（長さを1に）してから速度を掛ける
233      -        self.speed_x = (dx / distance) * total_speed
234      -        self.speed_y = (dy / distance) * total_speed
235      186 +    # --- 計算ここまで ---
236      187 +
237      188     def update(self):
238      189     -    # 浮動小数点数で座標を更新（より滑らかになる）
239      190     -    self.rect.x += self.speed_x
240      191     +    """
241      192     +    """毎フレーム、プレイヤーの位置に追従する"""
242      193     +    self.rect.centerx = self.player.rect.centerx
243      194     +    self.rect.bottom = self.player.rect.top
244      195     +    """
245      196     +    class Item(pygame.sprite.Sprite):
246      197     +        """全てのアイテムの親となる基本クラス"""
247      198     +        def __init__(self, center):
248      199     +            """アイテムの基本設定"""
249      200     +            super().__init__()
250      201     +            self.rect = self.image.get_rect(center=center) # 画像(self.image)は子クラスで設
251      202     +            定
252      203     +            self.speed_y = 3
253      204     +    """
254      205     +    def update(self):
255      206     +        """アイテムの状態を毎フレーム更新する（下に移動）"""
256      207     -    self.rect.y += self.speed_y
257      208     -    # 画面外に出たら削除
258      209     -    if self.rect.top > SCREEN_HEIGHT or self.rect.bottom < 0 or self.rect.left >
259      210     -        SCREEN_WIDTH or self.rect.right < 0:
260      211     +    if self.rect.top > SCREEN_HEIGHT: # 画面外に出たら消える
261      212     +        self.kill()
262      213     -    # --- 爆発エフェクト クラス (Explosion Class) ---
263      214     +    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 +
217 + class AttackUpItem(Item):
218 +     """攻撃力アップアイテムのクラス"""
219 +     def __init__(self, center):
220 +         self.image = pygame.transform.scale(ATTACK_ITEM_IMAGE, (30, 30))
221 +         super().__init__(center)
222 +
223 +     def apply_effect(self, player):
224 +         """取得した際のパワーアップ効果を適用する"""
225 +         player.power_up()
226 +
245 227     class Explosion(pygame.sprite.Sprite):
228 +         """爆発エフェクトを管理するクラス"""
246 229         def __init__(self, center, size="normal"):
230 +             """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]
232 +             self.original_image = EXPLOSION_IMAGE
233 +             # "large"か"normal"かでサイズを変える
234 +             scale = (90, 90) if size == "large" else (60, 60)
235 +             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()
237 +             self.duration = 400 # 表示時間 (ミリ秒)
238 +             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
290      241 +     """毎フレーム、表示時間が過ぎたかチェックする"""
291      242 +     if pygame.time.get_ticks() - self.creation_time > self.duration:
292          -         self.kill() # 一定時間経ったら消える
293
294      245 +     # --- 描画関数群 ---
295
296      246 +
297      247 +     def draw_stars(surface, stars, speed_level=0):
298          +     """背景の星を描画し、スクロールさせる"""
299          +     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
299          +     for star in stars:
300              -         pygame.draw.circle(surface, WHITE, (star[0], star[1]), star[3])
301              -         star[1] += star[2] * speed_modifier
302
303          -         if star[1] > SCREEN_HEIGHT:
304              -             star[1] = 0
305              -             star[0] = random.randrange(0, SCREEN_WIDTH)
306
307          253 +         if star[1] > SCREEN_HEIGHT: # 画面下に出たら上に戻す
307          254 +             star[0] = random.randrange(0, SCREEN_WIDTH); star[1] = 0
308
309      255
310
311      256 -     # --- テキスト描画用のヘルパー関数 (Helper function for drawing text) ---
312      256 +     def draw_text(surface, text, font, color, x, y, align="topright"):
313
314          -         text_surface = font.render(text, True, color)
315
316          257 +         """指定された位置にテキストを描画する"""
316          258 +         text_surface = font.render(text, True, color)
317
318          259 +         text_rect = text_surface.get_rect()
319
320          259 -         if align == "topright":
321              -             text_rect.topright = (x, y)
322
323          260 +         elif align == "center":
324              -             text_rect.center = (x, y)
325
326          261 +         elif align == "topleft":
327              -             text_rect.topleft = (x, y)
328
329          260 +         if align == "topright": text_rect.topright = (x, y)
330          261 +         elif align == "center": text_rect.center = (x, y)
331          262 +         elif align == "topleft": text_rect.topleft = (x, y)
332
333      263 +         surface.blit(text_surface, text_rect)
334
335      264
336
337      310 -     # --- フォントの設定 ---
338      311 -     score_font = pygame.font.SysFont(None, 36)
339      312 -     game_over_font = pygame.font.SysFont(None, 64, bold=True)
340
341      313 -
342
343      314 -     # --- 背景用の星を作成 ---
344      315 -     stars = create_stars(100)
345
```

```

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
        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 =
281          [[random.randrange(0,SCREEN_WIDTH),random.randrange(0,SCREEN_HEIGHT),random.randrange(
282              1,4),random.randrange(1,4)] for _ in range(100)]
283      + # スプライトを管理するためのグループを作成
284      + all_sprites = pygame.sprite.Group() # 全てのスプライト (描画・更新用)
285      + enemies_group = pygame.sprite.Group() # 敵 (衝突判定用)
286      + player_bullets_group = pygame.sprite.Group() # プレイヤーの弾 (衝突判定用)
287      + items_group = pygame.sprite.Group() # アイテム (衝突判定用)
288      + laser_group = pygame.sprite.Group() # レーザー (衝突判定用)
289      + # 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      290      -
327      290      - # --- 敵を定期的に生成するためのカスタムイベント ---
328      291      + # 敵を定期的に生成するためのカスタムイベントを設定
329      291      ADD_ENEMY = pygame.USEREVENT + 1
330      292      - initial_spawn_rate = 1000
331      292      - current_spawn_rate = initial_spawn_rate
332      292      - pygame.time.set_timer(ADD_ENEMY, current_spawn_rate)
333      293      -
334      293      - # --- スコアとゲームレベル ---
335      293      - score = 0
336      293      - game_speed_level = 0
336      294      - game_over = False
337      294      + pygame.time.set_timer(ADD_ENEMY, 1000) # 1000ミリ秒(1秒)ごとに発生
338      294      + # ゲームの状態を管理する変数を初期化
339      294      + score, game_speed_level, game_over, running = 0, 0, False, True
340      295      #
341      295      - --- メインゲームループ (Main Game Loop) ---
342      296      - running = True
343      297      while running:
344      297      - # 1. フレームレートの制御 (Control frame rate)
345      298      + # フレームレートの制御
346      299      clock.tick(FPS)
347      300

```

```

344      -      # 2. イベント処理 (Event handling)
345      301     +      # --- イベント処理 ---
346      302     for event in pygame.event.get():
347      303     +      # ウィンドウのxボタンが押されたら終了
348      304     if event.type == pygame.QUIT:
349      305         running = False
350      306     +      # ゲームオーバー中に、何かキーが押されたら終了
351      307     +      elif game_over and event.type == pygame.KEYDOWN:
352      308         running = False
353      309     +      # ADD_ENEMYイベントが発生したら、新しい敵を生成
354      310     elif event.type == ADD_ENEMY and not game_over:
355      311     -      # Enemy生成時に player オブジェクトを渡す
356      312     -      new_enemy = Enemy(game_speed_level, all_sprites, enemy_bullets_group,
357      313             player)
358      314     -      all_sprites.add(new_enemy)
359      315     -      enemies_group.add(new_enemy)
360      316     +      enemy = Enemy(game_speed_level, player)
361      317     +      all_sprites.add(enemy)
362      318     +      enemies_group.add(enemy)
363      319     -      # 射撃 (スペースキーが押され続けているかチェック)
364      320     -      # 操作処理 ---
365      321     keys = pygame.key.get_pressed()
366      322     +      # スペースキーが押されていたら弾を発射
367      323     if keys[pygame.K_SPACE] and not game_over:
368      324         player.shoot(all_sprites, player_bullets_group)
369      325     -      # 3. 更新 (Update)
370      326     -      # 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)
371      327     +      if player.powerup_level >= 2 and not game_over:
372      328         +      if keys[pygame.K_SPACE]:
373      329             +      if not player.active_laser: # レーザーがなければ生成
374      330                 player.active_laser = SuperLaser(player)
375      331                 all_sprites.add(player.active_laser);
376      332                 laser_group.add(player.active_laser)
377             else: # スペースを離したら消す
378                 if player.active_laser:
379                     player.active_laser.kill(); player.active_laser = None
380             else: # パワーアップ中でなければ消す
381                 if player.active_laser:
382                     player.active_laser.kill(); player.active_laser = None
383             +
384             +      # --- 更新処理 ---
385      335     if not game_over:
386      336     -      all_sprites.update()
387      337     -      else:
388      338     -      # ゲームオーバー後も爆発エフェクトは更新する
389      339     -      # explosionオブジェクトがkillされるまでupdateを続ける
390      340     for sprite in all_sprites:
391      341         if isinstance(sprite, Explosion):
392             sprite.update()
393
394     -      # 4. 衝突判定 (Collision Detection)
395      346     all_sprites.update() # 全てのスプライトの状態を更新
396      347     else: # ゲームオーバー後は爆発エフェクトだけ更新
397      348     for s in all_sprites:
398             if isinstance(s, Explosion):

```

```

340 +         s.update()
341 +
342 +     # --- 衝突判定 ---
370 343     if not game_over:
371 -
372 -         enemies_destroyed_this_frame = 0
373 -
374 -         # プレイヤーの弾と敵の衝突
375 -         hits_normal = pygame.sprite.groupcollide(player_bullets_group, enemies_group,
376 -             True, False)
376 -         for bullet, enemies_hit in hits_normal.items():
377 -             for enemy_hit in enemies_hit:
378 -                 if enemy_hit.hit():
379 -                     explosion = Explosion(enemy_hit.rect.center, "normal")
380 -                     all_sprites.add(explosion)
381 -                     score += enemy_hit.score_value
382 -                     enemies_destroyed_this_frame += 1
383 -                     enemy_hit.kill()
384 -
385 -         # レベルアップ処理
386 -         if enemies_destroyed_this_frame > 0:
387 -             new_speed_level = score // 10
388 -             if new_speed_level > game_speed_level:
389 -                 game_speed_level = new_speed_level
390 -                 print(f"--- SPEED LEVEL UP! Level: {game_speed_level} ---")
391 -
392 -             current_spawn_rate = max(150, int(initial_spawn_rate * (0.9 **
393 -                                         game_speed_level)))
393 -             pygame.time.set_timer(ADD_ENEMY, 0)
394 -             pygame.time.set_timer(ADD_ENEMY, current_spawn_rate)
395 -             print(f"New Spawn Rate: {current_spawn_rate} ms")

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,
347 +         True, True)
347 +     hits_laser = pygame.sprite.groupcollide(laser_group, enemies_group, False,
348 +         True) # レーザーは消えない
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)

396 367
397 368     # 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,
410      -                                         True)
411      -         if player_beam_hits:
412      -             explosion = Explosion(player.rect.center, "normal")
413      -             all_sprites.add(explosion)
414      -             player.hide()
415      -             game_over = True
416      -             print("Game Over! (Hit by enemy beam)")
417      -             pygame.time.set_timer(ADD_ENEMY, 0)
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,
426      -                  align="topright")
427      382      +         # --- 描画処理 ---
428      383      +         screen.fill(BLACK) # 画面を黒で塗りつぶす
429      384      +         draw_stars(screen, stars, game_speed_level) # 背景の星を描画
430      385      +         all_sprites.draw(screen) # 全てのスプライトを描画
431      386      +         # UI (スコア、レベル、体力バー) を描画
432      387      +         draw_text(screen, f"SCORE: {score}", score_font, WHITE, SCREEN_WIDTH - 10, 10,
433      -                  "topright")
434      388      +         draw_text(screen, f"LEVEL: {game_speed_level}", score_font, WHITE, 10, 10,
435      -                  "topleft")
436      389      +         draw_text(screen, "HP", score_font, WHITE, 10, 40, "topleft")
437      390      +         draw_health_bar(screen, 50, 45, player.health)
438
439      426      391
440      427      -         # レベルを描画
441      428      -         draw_text(screen, f"LEVEL: {game_speed_level}", score_font, WHITE, 10, 10,
442      -                  align="topleft")
443      429      -
```

430		- # ゲームオーバー表示 + # 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() ⊖