```
import pygame
import random
import math
import json
import os

# --- Game Constants ---
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
FPS = 60
PLAYER_SIZE = 30
OBSTACLE_SIZE = 40
REWARD_SIZE = 25
POWERUP_SIZE = 30
PLAYER_SPEED = 7
INITIAL_SCROLL_SPEED = 3
SPEED_INCREASE_RATE = 0.01
SPAWN_INTERVAL_MIN = 160
SPAWN_INTERVAL_MAX = 240
BOSS_SPAWN_SCORE = 500

# Colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
GRAY = (100, 100, 100)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
YELLOW = (255, 255, 0)
BLUE = (0, 0, 255)
ORANGE = (255, 165, 0)
PURPLE = (128, 0, 128)
```

```
CYAN = (0, 255, 255)
PINK = (255, 20, 147)
GOLD = (255, 215, 0)

# --- Pygame Initialization ---
pygame.init()
pygame.mixer.init()
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Enhanced Space Runner")
clock = pygame.time.Clock()
font_path = pygame.font.match_font('dejavusansmono')
font = pygame.font.Font(font_path, 36)
small_font = pygame.font.Font(font_path, 24)
tiny_font = pygame.font.Font(font_path, 18)

# --- Sound System ---
class SoundManager:
    """Manages all game sounds"""
    def __init__(self):
        self.sounds = {}
        self.music_volume = 0.3
        self.sfx_volume = 0.5
        self.create_sounds()

    def create_sounds(self):
        """Create simple beep sounds using pygame.mixer"""
        # Create sound effects programmatically
        try:
            # Coin collect sound
            self.sounds['coin'] = self.create_tone(440, 0.1)
            # Power-up sound
```

```
            self.sounds['powerup'] = self.create_tone(880, 0.15)
            # Explosion sound
            self.sounds['explosion'] = self.create_tone(110, 0.2)
            # Boss appear sound
            self.sounds['boss_appear'] = self.create_tone(220, 0.3)
            # Achievement sound
            self.sounds['achievement'] = self.create_tone(660, 0.2)
            # Level up sound
            self.sounds['levelup'] = self.create_tone(550, 0.25)
        except:
            pass

    def create_tone(self, frequency, duration):
        """Create a simple tone"""
        sample_rate = 22050
        n_samples = int(round(duration * sample_rate))
        buf = []
        for i in range(n_samples):
            value = int(4096 * math.sin(2 * math.pi * frequency * i / sample_rate))
            buf.append([value, value])
        sound = pygame.sndarray.make_sound(buf)
        sound.set_volume(self.sfx_volume)
        return sound

    def play(self, sound_name):
        """Play a sound effect"""
        if sound_name in self.sounds:
            try:
                self.sounds[sound_name].play()
            except:
                pass
```

```
sound_manager = SoundManager()


# --- Achievement System ---
ACHIEVEMENTS = {
    'first_blood': {'name': 'First Blood', 'desc': 'Score 100 points', 'requirement': 100, 'icon': '🎯'},
    'coin_collector': {'name': 'Coin Collector', 'desc': 'Collect 50 coins total', 'requirement': 50, 'icon': '💰'},
    'survivor': {'name': 'Survivor', 'desc': 'Score 500 points', 'requirement': 500, 'icon': '🛡️'},
    'boss_slayer': {'name': 'Boss Slayer', 'desc': 'Defeat a boss', 'requirement': 1, 'icon': '⚔️'},
    'millionaire': {'name': 'Millionaire', 'desc': 'Score 1000 points', 'requirement': 1000, 'icon': '👑'},
    'power_user': {'name': 'Power User', 'desc': 'Collect 10 power-ups', 'requirement': 10, 'icon': '⚡'},
    'treasure_hunter': {'name': 'Treasure Hunter', 'desc': 'Collect 5 treasure chests', 'requirement': 5, 'icon': '📦'},
    'speed_demon': {'name': 'Speed Demon', 'desc': 'Reach speed level 10', 'requirement': 10, 'icon': '🚀'},
}

# --- Game State Variables ---
high_score = 0
coins = 0
unlocked_skins = ["default"]
current_skin = "default"
achievements_unlocked = []
stats = {
    'total_coins': 0,
    'bosses_defeated': 0,
    'powerups_collected': 0,
    'treasures_collected': 0,
    'max_speed_level': 0
}
```

```
# --- Save/Load System ---
def save_game_data():
    """Save game progress"""
    data = {
        'high_score': high_score,
        'coins': coins,
        'unlocked_skins': unlocked_skins,
        'current_skin': current_skin,
        'achievements': achievements_unlocked,
        'stats': stats
    }
    with open('space_runner_save.json', 'w') as f:
        json.dump(data, f)

def load_game_data():
    """Load game progress"""
    global high_score, coins, unlocked_skins, current_skin, achievements_unlocked, stats
    try:
        if os.path.exists('space_runner_save.json'):
            with open('space_runner_save.json', 'r') as f:
                data = json.load(f)
                high_score = data.get('high_score', 0)
                coins = data.get('coins', 0)
                unlocked_skins = data.get('unlocked_skins', ["default"])
                current_skin = data.get('current_skin', "default")
                achievements_unlocked = data.get('achievements', [])
                stats = data.get('stats', {'total_coins': 0, 'bosses_defeated': 0, 'powerups_collected': 0,
'treasures_collected': 0, 'max_speed_level': 0})
    except:
        pass
```

```
def check_achievement(achievement_id, value):
    """Check if achievement should be unlocked"""
    if achievement_id not in achievements_unlocked:
        if value >= ACHIEVEMENTS[achievement_id]['requirement']:
            achievements_unlocked.append(achievement_id)
            sound_manager.play('achievement')
            return True
    return False

# --- Particle System ---
class Particle(pygame.sprite.Sprite):
    """Visual effect particle"""
    def __init__(self, x, y, color):
        super().__init__()
        self.image = pygame.Surface((4, 4), pygame.SRCALPHA)
        pygame.draw.circle(self.image, color, (2, 2), 2)
        self.rect = self.image.get_rect(center=(x, y))
        self.speedx = random.randint(-5, 5)
        self.speedy = random.randint(-5, 5)
        self.lifetime = 30

    def update(self):
        self.rect.x += self.speedx
        self.rect.y += self.speedy
        self.lifetime -= 1
        if self.lifetime <= 0:
            self.kill()

# --- Player Class ---
class Player(pygame.sprite.Sprite):
```

```
    """Enhanced player with power-ups and different skins"""
    def __init__(self, skin="default"):
        super().__init__()
        self.skin = skin
        self.image = pygame.Surface((PLAYER_SIZE, PLAYER_SIZE), pygame.SRCALPHA)
        self.rect = self.image.get_rect()
        self.rect.centerx = 100
        self.rect.centery = SCREEN_HEIGHT // 2
        self.speedy = 0
        self.shield_active = False
        self.shield_timer = 0
        self.magnet_active = False
        self.magnet_timer = 0
        self.speed_boost_active = False
        self.speed_boost_timer = 0
        self.invincible = False
        self.invincible_timer = 0
        self.draw_character()

    def draw_character(self):
        """Draw player based on selected skin"""
        self.image.fill((0, 0, 0, 0))

        if self.skin == "default":
            pygame.draw.circle(self.image, WHITE, (PLAYER_SIZE // 2, PLAYER_SIZE // 2), PLAYER_SIZE // 2
- 2, 2)
            pygame.draw.rect(self.image, GRAY, (0, PLAYER_SIZE // 2, PLAYER_SIZE, PLAYER_SIZE // 2 - 2))
            pygame.draw.circle(self.image, GRAY, (PLAYER_SIZE // 2, PLAYER_SIZE // 2), PLAYER_SIZE // 2 -
2)
            pygame.draw.circle(self.image, CYAN, (PLAYER_SIZE // 2, PLAYER_SIZE // 2), PLAYER_SIZE // 2 -
5)
```

```
            visor_rect = pygame.Rect(PLAYER_SIZE // 4, PLAYER_SIZE // 4, PLAYER_SIZE // 2, PLAYER_SIZE
// 4)
            pygame.draw.rect(self.image, BLUE, visor_rect)
            pygame.draw.line(self.image, GRAY, (PLAYER_SIZE // 2, 0), (PLAYER_SIZE // 2, 5), 2)
            pygame.draw.circle(self.image, GRAY, (PLAYER_SIZE // 2, 0), 2)
        elif self.skin == "golden":
            pygame.draw.circle(self.image, GOLD, (PLAYER_SIZE // 2, PLAYER_SIZE // 2), PLAYER_SIZE // 2 -
2)
            pygame.draw.circle(self.image, YELLOW, (PLAYER_SIZE // 2, PLAYER_SIZE // 2), PLAYER_SIZE //
2 - 5)
            visor_rect = pygame.Rect(PLAYER_SIZE // 4, PLAYER_SIZE // 4, PLAYER_SIZE // 2, PLAYER_SIZE
// 4)
            pygame.draw.rect(self.image, ORANGE, visor_rect)
        elif self.skin == "robot":
            pygame.draw.rect(self.image, GRAY, (5, 5, PLAYER_SIZE - 10, PLAYER_SIZE - 10))
            pygame.draw.rect(self.image, RED, (8, 8, 6, 6))
            pygame.draw.rect(self.image, RED, (PLAYER_SIZE - 14, 8, 6, 6))
            pygame.draw.rect(self.image, CYAN, (10, PLAYER_SIZE - 12, PLAYER_SIZE - 20, 4))
        elif self.skin == "alien":
            pygame.draw.ellipse(self.image, GREEN, (5, 3, PLAYER_SIZE - 10, PLAYER_SIZE - 6))
            pygame.draw.circle(self.image, BLACK, (12, 12), 4)
            pygame.draw.circle(self.image, BLACK, (PLAYER_SIZE - 12, 12), 4)

    def update(self):
        """Update player position and power-up timers"""
        speed_multiplier = 1.5 if self.speed_boost_active else 1
        self.rect.y += self.speedy * speed_multiplier

        if self.rect.top < 0:
            self.rect.top = 0
        if self.rect.bottom > SCREEN_HEIGHT:
            self.rect.bottom = SCREEN_HEIGHT
```

```
    if self.shield_active:
        self.shield_timer -= 1
        if self.shield_timer <= 0:
            self.shield_active = False

    if self.magnet_active:
        self.magnet_timer -= 1
        if self.magnet_timer <= 0:
            self.magnet_active = False

    if self.speed_boost_active:
        self.speed_boost_timer -= 1
        if self.speed_boost_timer <= 0:
            self.speed_boost_active = False

    if self.invincible:
        self.invincible_timer -= 1
        if self.invincible_timer <= 0:
            self.invincible = False

def activate_shield(self):
    self.shield_active = True
    self.shield_timer = 300

def activate_magnet(self):
    self.magnet_active = True
    self.magnet_timer = 360

def activate_speed_boost(self):
    self.speed_boost_active = True
```

```
        self.speed_boost_timer = 240

    def draw_powerup_indicators(self, surface):
        """Draw active power-up indicators"""
        y_offset = 60
        if self.shield_active:
            draw_text("SHIELD", tiny_font, CYAN, 70, y_offset)
            y_offset += 25
        if self.magnet_active:
            draw_text("MAGNET", tiny_font, PURPLE, 70, y_offset)
            y_offset += 25
        if self.speed_boost_active:
            draw_text("SPEED", tiny_font, ORANGE, 70, y_offset)

# --- Boss Types ---
class Boss(pygame.sprite.Sprite):
    """Boss enemy with different types"""
    def __init__(self, boss_type="alien"):
        super().__init__()
        self.boss_type = boss_type
        self.image = pygame.Surface((80, 80), pygame.SRCALPHA)
        self.rect = self.image.get_rect()
        self.rect.right = SCREEN_WIDTH + 100
        self.rect.centery = SCREEN_HEIGHT // 2

        if boss_type == "alien":
            self.health = 5
            self.speedx = -2
            self.speedy = 2
        elif boss_type == "asteroid":
            self.health = 8
```

```
            self.speedx = -1.5
            self.speedy = 1.5
        elif boss_type == "mothership":
            self.health = 10
            self.speedx = -1
            self.speedy = 1

        self.max_health = self.health
        self.shoot_timer = 0
        self.draw_boss()

    def draw_boss(self):
        """Draw boss based on type"""
        if self.boss_type == "alien":
            pygame.draw.circle(self.image, RED, (40, 40), 35)
            pygame.draw.circle(self.image, (150, 0, 0), (40, 40), 30)
            pygame.draw.circle(self.image, YELLOW, (30, 30), 8)
            pygame.draw.circle(self.image, YELLOW, (50, 30), 8)
            pygame.draw.circle(self.image, BLACK, (30, 30), 4)
            pygame.draw.circle(self.image, BLACK, (50, 30), 4)
        elif self.boss_type == "asteroid":
            pygame.draw.circle(self.image, (80, 80, 80), (40, 40), 38)
            pygame.draw.circle(self.image, (60, 60, 60), (25, 25), 12)
            pygame.draw.circle(self.image, (60, 60, 60), (55, 30), 8)
            pygame.draw.circle(self.image, (60, 60, 60), (35, 55), 10)
        elif self.boss_type == "mothership":
            pygame.draw.ellipse(self.image, PURPLE, (10, 25, 60, 30))
            pygame.draw.circle(self.image, (100, 0, 100), (40, 40), 20)
            pygame.draw.rect(self.image, CYAN, (15, 35, 10, 10))
            pygame.draw.rect(self.image, CYAN, (55, 35, 10, 10))
```

```
    def update(self):
        """Update boss movement"""
        self.rect.y += self.speedy
        if self.rect.top < 0 or self.rect.bottom > SCREEN_HEIGHT:
            self.speedy *= -1
        self.shoot_timer += 1

    def take_damage(self):
        """Boss takes damage"""
        self.health -= 1
        if self.health <= 0:
            self.kill()
            return True
        return False

class BossProjectile(pygame.sprite.Sprite):
    """Projectiles fired by boss"""
    def __init__(self, x, y):
        super().__init__()
        self.image = pygame.Surface((15, 15), pygame.SRCALPHA)
        pygame.draw.circle(self.image, RED, (7, 7), 7)
        self.rect = self.image.get_rect(center=(x, y))
        self.speedx = -5

    def update(self):
        self.rect.x += self.speedx
        if self.rect.right < 0:
            self.kill()

# --- Power-up Class ---
class PowerUp(pygame.sprite.Sprite):
```

```
    """Power-ups that give special abilities"""
    def __init__(self, power_type):
        super().__init__()
        self.power_type = power_type
        self.image = pygame.Surface((POWERUP_SIZE, POWERUP_SIZE), pygame.SRCALPHA)
        self.rect = self.image.get_rect()
        self.rect.right = SCREEN_WIDTH + random.randint(50, 100)
        self.rect.y = random.randint(0, SCREEN_HEIGHT - POWERUP_SIZE)
        self.speedx = -INITIAL_SCROLL_SPEED
        self.draw_powerup()

    def draw_powerup(self):
        """Draw power-up based on type"""
        if self.power_type == "shield":
            pygame.draw.circle(self.image, CYAN, (POWERUP_SIZE // 2, POWERUP_SIZE // 2),
POWERUP_SIZE // 2 - 2, 3)
            pygame.draw.circle(self.image, CYAN, (POWERUP_SIZE // 2, POWERUP_SIZE // 2),
POWERUP_SIZE // 3, 3)
        elif self.power_type == "magnet":
            pygame.draw.rect(self.image, PURPLE, (5, 8, POWERUP_SIZE - 10, 8))
            pygame.draw.arc(self.image, PURPLE, (5, 5, POWERUP_SIZE // 2 - 5, 15), 0, 3.14, 3)
            pygame.draw.arc(self.image, PURPLE, (POWERUP_SIZE // 2, 5, POWERUP_SIZE // 2 - 5, 15), 0,
3.14, 3)
        elif self.power_type == "speed":
            pygame.draw.polygon(self.image, ORANGE, [(5, POWERUP_SIZE // 2),
                                                     (POWERUP_SIZE - 5, 5),
                                                     (POWERUP_SIZE - 5, POWERUP_SIZE - 5)])

    def update(self, scroll_speed):
        """Update power-up position"""
        self.rect.x += -scroll_speed
        if self.rect.right < 0:
```

```
            self.kill()


# --- Obstacle Class ---
class Obstacle(pygame.sprite.Sprite):
    """Enhanced obstacle with rotation"""
    def __init__(self, type):
        super().__init__()
        self.type = type
        self.image = pygame.Surface((OBSTACLE_SIZE, OBSTACLE_SIZE), pygame.SRCALPHA)
        self.original_image = self.image
        self.rect = self.image.get_rect()
        self.rect.right = SCREEN_WIDTH + random.randint(50, 100)
        self.rect.y = random.randint(0, SCREEN_HEIGHT - OBSTACLE_SIZE)
        self.speedx = -INITIAL_SCROLL_SPEED
        self.rotation = 0
        self.draw_obstacle()
        self.original_image = self.image.copy()


    def draw_obstacle(self):
        """Draw obstacle based on type"""
        if self.type == "asteroid":
            pygame.draw.circle(self.image, GRAY, (OBSTACLE_SIZE // 2, OBSTACLE_SIZE // 2),
OBSTACLE_SIZE // 2)
            pygame.draw.circle(self.image, (80, 80, 80), (OBSTACLE_SIZE // 4, OBSTACLE_SIZE // 4),
OBSTACLE_SIZE // 8)
        elif self.type == "alien":
            pygame.draw.polygon(self.image, GREEN, [(OBSTACLE_SIZE // 2, 0), (0, OBSTACLE_SIZE),
(OBSTACLE_SIZE, OBSTACLE_SIZE)])
            pygame.draw.circle(self.image, WHITE, (OBSTACLE_SIZE // 2, OBSTACLE_SIZE // 2),
OBSTACLE_SIZE // 8)
        elif self.type == "black_hole":
            pygame.draw.circle(self.image, BLACK, (OBSTACLE_SIZE // 2, OBSTACLE_SIZE // 2),
OBSTACLE_SIZE // 2)
```

```
            pygame.draw.circle(self.image, PURPLE, (OBSTACLE_SIZE // 2, OBSTACLE_SIZE // 2),
OBSTACLE_SIZE // 3, 2)
        elif self.type == "debris":
            pygame.draw.rect(self.image, GRAY, (0, 0, OBSTACLE_SIZE, OBSTACLE_SIZE))

    def update(self, scroll_speed):
        """Update obstacle position with rotation"""
        self.rect.x += -scroll_speed
        self.rotation += 2
        self.image = pygame.transform.rotate(self.original_image, self.rotation)
        self.rect = self.image.get_rect(center=self.rect.center)
        if self.rect.right < 0:
            self.kill()

# --- Reward Class ---
class Reward(pygame.sprite.Sprite):
    """Enhanced reward with coin collection"""
    def __init__(self, type):
        super().__init__()
        self.type = type
        self.image = pygame.Surface((REWARD_SIZE, REWARD_SIZE), pygame.SRCALPHA)
        self.rect = self.image.get_rect()
        self.rect.right = SCREEN_WIDTH + random.randint(50, 100)
        self.rect.y = random.randint(0, SCREEN_HEIGHT - REWARD_SIZE)
        self.speedx = -INITIAL_SCROLL_SPEED
        self.points = 0
        self.coin_value = 0
        self.draw_reward()

    def draw_reward(self):
        """Draw reward and assign values"""
```

```
        if self.type == "star":
            self.points = 10
            self.coin_value = 1
            pygame.draw.circle(self.image, YELLOW, (REWARD_SIZE // 2, REWARD_SIZE // 2),
REWARD_SIZE // 2)
        elif self.type == "planet":
            self.points = 50
            self.coin_value = 5
            planet_color = random.choice([BLUE, GREEN, ORANGE, PURPLE])
            pygame.draw.circle(self.image, planet_color, (REWARD_SIZE // 2, REWARD_SIZE // 2),
REWARD_SIZE // 2)
        elif self.type == "treasure":
            self.points = 100
            self.coin_value = 10
            pygame.draw.rect(self.image, YELLOW, (0, 0, REWARD_SIZE, REWARD_SIZE), border_radius=5)


    def update(self, scroll_speed, player=None):
        """Update reward position with magnet effect"""
        if player and player.magnet_active:
            dx = player.rect.centerx - self.rect.centerx
            dy = player.rect.centery - self.rect.centery
            dist = math.sqrt(dx**2 + dy**2)
            if dist < 200 and dist > 0:
                self.rect.x += dx / dist * 3
                self.rect.y += dy / dist * 3


        self.rect.x += -scroll_speed
        if self.rect.right < 0:
            self.kill()


# --- Background System ---
class BackgroundManager:
```

```
"""Manages different background themes for levels"""
def __init__(self):
    self.current_theme = "space"
    self.stars = []
    self.nebula_particles = []
    self.reset_background()

def reset_background(self):
    """Reset background elements"""
    self.stars = []
    for _ in range(200):
        self.stars.append([random.randint(0, SCREEN_WIDTH),
                           random.randint(0, SCREEN_HEIGHT),
                           random.randint(1, 3)])

    self.nebula_particles = []
    for _ in range(50):
        self.nebula_particles.append([random.randint(0, SCREEN_WIDTH),
                                      random.randint(0, SCREEN_HEIGHT),
                                      random.randint(10, 30),
                                      random.choice([PURPLE, BLUE, PINK, CYAN])])

def change_theme(self, level):
    """Change background based on level"""
    if level < 3:
        self.current_theme = "space"
    elif level < 6:
        self.current_theme = "nebula"
    elif level < 9:
        self.current_theme = "asteroid_field"
    else:
```

```
        self.current_theme = "deep_space"

def draw(self, surface, scroll_speed):
    """Draw background based on theme"""
    if self.current_theme == "space":
        surface.fill(BLACK)
        for star in self.stars:
            pygame.draw.circle(surface, WHITE, (int(star[0]), int(star[1])), star[2])
            star[0] -= scroll_speed * 0.5
            if star[0] < 0:
                star[0] = SCREEN_WIDTH
                star[1] = random.randint(0, SCREEN_HEIGHT)

    elif self.current_theme == "nebula":
        surface.fill((10, 0, 20))
        for nebula in self.nebula_particles:
            s = pygame.Surface((nebula[2] * 2, nebula[2] * 2), pygame.SRCALPHA)
            pygame.draw.circle(s, (*nebula[3][:3], 30), (nebula[2], nebula[2]), nebula[2])
            surface.blit(s, (int(nebula[0]), int(nebula[1])))
            nebula[0] -= scroll_speed * 0.3
            if nebula[0] < -50:
                nebula[0] = SCREEN_WIDTH + 50
        for star in self.stars[:100]:
            pygame.draw.circle(surface, WHITE, (int(star[0]), int(star[1])), star[2])
            star[0] -= scroll_speed * 0.5
            if star[0] < 0:
                star[0] = SCREEN_WIDTH

    elif self.current_theme == "asteroid_field":
        surface.fill((20, 10, 0))
        for star in self.stars:
```

```
            color = random.choice([WHITE, GRAY, ORANGE])
            pygame.draw.circle(surface, color, (int(star[0]), int(star[1])), star[2])
            star[0] -= scroll_speed * 0.6
            if star[0] < 0:
                star[0] = SCREEN_WIDTH

        else:  # deep_space
            surface.fill((5, 0, 15))
            for star in self.stars:
                brightness = random.randint(100, 255)
                color = (brightness, brightness, 255)
                pygame.draw.circle(surface, color, (int(star[0]), int(star[1])), star[2])
                star[0] -= scroll_speed * 0.4
                if star[0] < 0:
                    star[0] = SCREEN_WIDTH

# --- Utility Functions ---
def draw_text(text, font, color, x, y):
    """Render text to screen"""
    text_surface = font.render(text, True, color)
    text_rect = text_surface.get_rect()
    text_rect.midtop = (x, y)
    screen.blit(text_surface, text_rect)

def create_particles(x, y, color, group):
    """Create particle explosion effect"""
    for _ in range(10):
        particle = Particle(x, y, color)
        group.add(particle)

# --- Achievement Notification ---
```

```
class AchievementNotification:
    """Shows achievement unlock notifications"""
    def __init__(self):
        self.active = False
        self.achievement_id = None
        self.timer = 0

    def show(self, achievement_id):
        """Display achievement"""
        self.active = True
        self.achievement_id = achievement_id
        self.timer = 180  # 3 seconds

    def update(self):
        """Update notification timer"""
        if self.active:
            self.timer -= 1
            if self.timer <= 0:
                self.active = False

    def draw(self, surface):
        """Draw notification"""
        if self.active and self.achievement_id:
            achievement = ACHIEVEMENTS[self.achievement_id]
            y = 100

            # Background box
            box_width = 350
            box_height = 80
            box_x = SCREEN_WIDTH // 2 - box_width // 2
            box_y = y - 10
```

```
        s = pygame.Surface((box_width, box_height), pygame.SRCALPHA)
        pygame.draw.rect(s, (*GOLD, 200), (0, 0, box_width, box_height), border_radius=10)
        pygame.draw.rect(s, GOLD, (0, 0, box_width, box_height), 3, border_radius=10)
        surface.blit(s, (box_x, box_y))

        draw_text("ACHIEVEMENT UNLOCKED!", tiny_font, WHITE, SCREEN_WIDTH // 2, y)
        draw_text(f"{achievement['icon']} {achievement['name']}", small_font, GOLD, SCREEN_WIDTH
// 2, y + 30)

# --- Menu and UI Functions ---
def show_main_menu():
    """Display main menu"""
    load_game_data()
    menu_running = True
    selected_option = 0
    options = ["PLAY", "SKINS", "ACHIEVEMENTS", "QUIT"]

    bg_stars = []
    for _ in range(100):
        bg_stars.append([random.randint(0, SCREEN_WIDTH), random.randint(0, SCREEN_HEIGHT),
random.randint(1, 2)])

    while menu_running:
        screen.fill(BLACK)

        for star in bg_stars:
            pygame.draw.circle(screen, WHITE, (int(star[0]), int(star[1])), star[2])
            star[0] -= 0.5
            if star[0] < 0:
                star[0] = SCREEN_WIDTH
```

```
        draw_text("ENHANCED SPACE RUNNER", font, CYAN, SCREEN_WIDTH // 2, 80)
        draw_text(f"High Score: {high_score}", small_font, YELLOW, SCREEN_WIDTH // 2, 150)
        draw_text(f"Coins: {coins}", small_font, GOLD, SCREEN_WIDTH // 2, 190)

        for i, option in enumerate(options):
            color = YELLOW if i == selected_option else WHITE
            draw_text(option, small_font, color, SCREEN_WIDTH // 2, 280 + i * 60)

        pygame.display.flip()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return "quit"
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP:
                    selected_option = (selected_option - 1) % len(options)
                elif event.key == pygame.K_DOWN:
                    selected_option = (selected_option + 1) % len(options)
                elif event.key == pygame.K_RETURN:
                    if selected_option == 0:
                        return "play"
                    elif selected_option == 1:
                        return "skins"
                    elif selected_option == 2:
                        return "achievements"
                    elif selected_option == 3:
                        return "quit"

        clock.tick(30)

def show_achievements_screen():
```

```
"""Display achievements"""
running = True

while running:
    screen.fill(BLACK)
    draw_text("ACHIEVEMENTS", font, CYAN, SCREEN_WIDTH // 2, 30)

    y_offset = 100
    col = 0
    for achievement_id, achievement in ACHIEVEMENTS.items():
        unlocked = achievement_id in achievements_unlocked
        color = GOLD if unlocked else GRAY

        x_pos = 200 if col == 0 else 600

        draw_text(achievement['icon'], small_font, color, x_pos, y_offset)
        draw_text(achievement['name'], tiny_font, color, x_pos, y_offset + 35)

        if unlocked:
            draw_text("✓", tiny_font, GREEN, x_pos, y_offset + 60)
        else:
            draw_text(f"{achievement['requirement']}", tiny_font, GRAY, x_pos, y_offset + 60)

        col += 1
        if col >= 2:
            col = 0
            y_offset += 100

    draw_text("Press ESC to return", tiny_font, WHITE, SCREEN_WIDTH // 2, 550)
    pygame.display.flip()
```

```
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    return

        clock.tick(30)

def show_skin_shop():
    """Display skin selection/shop"""
    global current_skin, coins, unlocked_skins

    skins = {
        "default": {"name": "Classic", "cost": 0, "unlocked": True},
        "golden": {"name": "Golden", "cost": 100, "unlocked": "golden" in unlocked_skins},
        "robot": {"name": "Robot", "cost": 200, "unlocked": "robot" in unlocked_skins},
        "alien": {"name": "Alien", "cost": 150, "unlocked": "alien" in unlocked_skins}
    }

    skin_list = list(skins.keys())
    selected_index = 0
    shop_running = True

    while shop_running:
        screen.fill(BLACK)
        draw_text("SKIN SHOP", font, CYAN, SCREEN_WIDTH // 2, 50)
        draw_text(f"Coins: {coins}", small_font, GOLD, SCREEN_WIDTH // 2, 110)

        selected_skin = skin_list[selected_index]
        skin_info = skins[selected_skin]
```

```
preview_player = Player(selected_skin)
preview_rect = preview_player.image.get_rect(center=(SCREEN_WIDTH // 2, 250))
screen.blit(preview_player.image, preview_rect)

draw_text(skin_info["name"], small_font, WHITE, SCREEN_WIDTH // 2, 320)

if skin_info["unlocked"]:
    if current_skin == selected_skin:
        draw_text("EQUIPPED", small_font, GREEN, SCREEN_WIDTH // 2, 370)
    else:
        draw_text("Press ENTER to Equip", tiny_font, YELLOW, SCREEN_WIDTH // 2, 370)
else:
    draw_text(f"Cost: {skin_info['cost']} coins", small_font, YELLOW, SCREEN_WIDTH // 2, 370)
    if coins >= skin_info['cost']:
        draw_text("Press ENTER to Buy", tiny_font, GREEN, SCREEN_WIDTH // 2, 410)
    else:
        draw_text("Not enough coins", tiny_font, RED, SCREEN_WIDTH // 2, 410)

draw_text("Use ARROW KEYS | ESC to return", tiny_font, GRAY, SCREEN_WIDTH // 2, 520)
pygame.display.flip()

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        return
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_ESCAPE:
            save_game_data()
            return
        elif event.key == pygame.K_LEFT:
            selected_index = (selected_index - 1) % len(skin_list)
```

```
            elif event.key == pygame.K_RIGHT:
                selected_index = (selected_index + 1) % len(skin_list)
            elif event.key == pygame.K_RETURN:
                if skin_info["unlocked"]:
                    current_skin = selected_skin
                    save_game_data()
                elif coins >= skin_info['cost']:
                    coins -= skin_info['cost']
                    unlocked_skins.append(selected_skin)
                    skins[selected_skin]["unlocked"] = True
                    current_skin = selected_skin
                    save_game_data()

        clock.tick(30)

def show_game_over_screen(score, coins_earned, level, new_achievements):
    """Display enhanced game over screen"""
    global high_score, coins

    if score > high_score:
        high_score = score
        new_record = True
    else:
        new_record = False

    coins += coins_earned
    save_game_data()

    screen.fill(BLACK)

    y_pos = 120
```

```
    if new_record:
        draw_text("NEW RECORD!", font, GOLD, SCREEN_WIDTH // 2, y_pos)
        y_pos += 60

    draw_text("GAME OVER", font, WHITE, SCREEN_WIDTH // 2, y_pos)
    y_pos += 70

    draw_text(f"Score: {score}", small_font, WHITE, SCREEN_WIDTH // 2, y_pos)
    y_pos += 40
    draw_text(f"Level Reached: {level}", small_font, CYAN, SCREEN_WIDTH // 2, y_pos)
    y_pos += 40
    draw_text(f"Coins Earned: +{coins_earned}", small_font, GOLD, SCREEN_WIDTH // 2, y_pos)
    y_pos += 40

    if new_achievements:
        draw_text(f"{len(new_achievements)} New Achievement(s)!", tiny_font, YELLOW,
SCREEN_WIDTH // 2, y_pos)
        y_pos += 30

    draw_text("Press 'R' to Restart | 'M' for Menu", small_font, WHITE, SCREEN_WIDTH // 2, 520)
    pygame.display.flip()

    waiting = True
    while waiting:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return "quit"
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_r:
                    return "restart"
                if event.key == pygame.K_m:
```

```
                return "menu"
        clock.tick(15)

# --- Main Game Function ---
def run_game():
    """Main game loop with all enhancements"""
    global coins, stats

    all_sprites = pygame.sprite.Group()
    obstacles = pygame.sprite.Group()
    rewards = pygame.sprite.Group()
    powerups = pygame.sprite.Group()
    particles = pygame.sprite.Group()
    bosses = pygame.sprite.Group()
    boss_projectiles = pygame.sprite.Group()

    player = Player(current_skin)
    all_sprites.add(player)

    score = 0
    coins_earned = 0
    scroll_speed = INITIAL_SCROLL_SPEED
    level = 1

    last_obstacle_spawn = pygame.time.get_ticks()
    last_reward_spawn = pygame.time.get_ticks()
    last_powerup_spawn = pygame.time.get_ticks()

    next_boss_score = BOSS_SPAWN_SCORE
    boss_active = False
    boss_types = ["alien", "asteroid", "mothership"]
```

```
current_boss_type = 0

background = BackgroundManager()
achievement_notification = AchievementNotification()

game_powerups_collected = 0
game_treasures_collected = 0
new_achievements = []

game_running = True

while game_running:
    clock.tick(FPS)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            return "quit"
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP or event.key == pygame.K_w:
                player.speedy = -PLAYER_SPEED
            elif event.key == pygame.K_DOWN or event.key == pygame.K_s:
                player.speedy = PLAYER_SPEED
            elif event.key == pygame.K_ESCAPE:
                return "menu"
        elif event.type == pygame.KEYUP:
            if event.key in [pygame.K_UP, pygame.K_w, pygame.K_DOWN, pygame.K_s]:
                player.speedy = 0

    player.update()
    scroll_speed += SPEED_INCREASE_RATE * 0.01
```

```
        # Calculate level
        new_level = (score // 300) + 1
        if new_level > level:
            level = new_level
            background.change_theme(level)
            sound_manager.play('levelup')
            if level > stats['max_speed_level']:
                stats['max_speed_level'] = level

        for obstacle in obstacles:
            obstacle.update(scroll_speed)
        for reward in rewards:
            reward.update(scroll_speed, player)
        for powerup in powerups:
            powerup.update(scroll_speed)
        for boss in bosses:
            boss.update()
        for projectile in boss_projectiles:
            projectile.update()
        particles.update()
        achievement_notification.update()

        now = pygame.time.get_ticks()
        if not boss_active and now - last_obstacle_spawn > random.randint(SPAWN_INTERVAL_MIN,
SPAWN_INTERVAL_MAX):
            last_obstacle_spawn = now
            obstacle_type = random.choice(["asteroid", "alien", "black_hole", "debris"])
            new_obstacle = Obstacle(obstacle_type)
            all_sprites.add(new_obstacle)
            obstacles.add(new_obstacle)
```

```
if now - last_reward_spawn > random.randint(80, 150):
    last_reward_spawn = now
    reward_type = random.choice(["star", "star", "planet", "treasure"])
    new_reward = Reward(reward_type)
    all_sprites.add(new_reward)
    rewards.add(new_reward)

if now - last_powerup_spawn > random.randint(400, 600):
    last_powerup_spawn = now
    powerup_type = random.choice(["shield", "magnet", "speed"])
    new_powerup = PowerUp(powerup_type)
    all_sprites.add(new_powerup)
    powerups.add(new_powerup)

if score >= next_boss_score and not boss_active:
    boss_type = boss_types[current_boss_type % len(boss_types)]
    boss = Boss(boss_type)
    all_sprites.add(boss)
    bosses.add(boss)
    boss_active = True
    current_boss_type += 1
    sound_manager.play('boss_appear')

for boss in bosses:
    if boss.shoot_timer > 90:
        boss.shoot_timer = 0
        projectile = BossProjectile(boss.rect.left, boss.rect.centery)
        all_sprites.add(projectile)
        boss_projectiles.add(projectile)

if not player.shield_active and not player.invincible:
```

```
    hits = pygame.sprite.spritecollide(player, obstacles, False)
    if hits:
        sound_manager.play('explosion')
        create_particles(player.rect.centerx, player.rect.centery, RED, particles)
        game_running = False

if not player.shield_active and not player.invincible:
    proj_hits = pygame.sprite.spritecollide(player, boss_projectiles, True)
    if proj_hits:
        sound_manager.play('explosion')
        create_particles(player.rect.centerx, player.rect.centery, RED, particles)
        game_running = False

reward_hits = pygame.sprite.spritecollide(player, rewards, True)
for reward in reward_hits:
    score += reward.points
    coins_earned += reward.coin_value
    stats['total_coins'] += reward.coin_value
    sound_manager.play('coin')
    create_particles(reward.rect.centerx, reward.rect.centery, YELLOW, particles)

    if reward.type == "treasure":
        game_treasures_collected += 1

powerup_hits = pygame.sprite.spritecollide(player, powerups, True)
for powerup in powerup_hits:
    if powerup.power_type == "shield":
        player.activate_shield()
    elif powerup.power_type == "magnet":
        player.activate_magnet()
    elif powerup.power_type == "speed":
```

```
            player.activate_speed_boost()
        sound_manager.play('powerup')
        create_particles(powerup.rect.centerx, powerup.rect.centery, PURPLE, particles)
        game_powerups_collected += 1
        stats['powerups_collected'] += 1

    if player.shield_active:
        boss_hits = pygame.sprite.spritecollide(player, bosses, False)
        for boss in boss_hits:
            if boss.take_damage():
                score += 500
                coins_earned += 50
                stats['bosses_defeated'] += 1
                sound_manager.play('explosion')
                create_particles(boss.rect.centerx, boss.rect.centery, GOLD, particles)
                boss_active = False
                next_boss_score = score + BOSS_SPAWN_SCORE

    # Check achievements
    if check_achievement('first_blood', score):
        new_achievements.append('first_blood')
        achievement_notification.show('first_blood')
    if check_achievement('survivor', score):
        new_achievements.append('survivor')
        achievement_notification.show('survivor')
    if check_achievement('millionaire', score):
        new_achievements.append('millionaire')
        achievement_notification.show('millionaire')
    if check_achievement('coin_collector', stats['total_coins']):
        new_achievements.append('coin_collector')
        achievement_notification.show('coin_collector')
```

```
if check_achievement('boss_slayer', stats['bosses_defeated']):
    new_achievements.append('boss_slayer')
    achievement_notification.show('boss_slayer')
if check_achievement('power_user', stats['powerups_collected']):
    new_achievements.append('power_user')
    achievement_notification.show('power_user')
if check_achievement('treasure_hunter', game_treasures_collected):
    new_achievements.append('treasure_hunter')
    achievement_notification.show('treasure_hunter')
if check_achievement('speed_demon', level):
    new_achievements.append('speed_demon')
    achievement_notification.show('speed_demon')

# Drawing
background.draw(screen, scroll_speed)

if player.shield_active:
    pygame.draw.circle(screen, CYAN, player.rect.center, PLAYER_SIZE, 2)

all_sprites.draw(screen)
particles.draw(screen)

for boss in bosses:
    bar_width = 60
    bar_height = 6
    fill = (boss.health / boss.max_health) * bar_width
    bar_x = boss.rect.centerx - bar_width // 2
    bar_y = boss.rect.top - 15
    pygame.draw.rect(screen, RED, (bar_x, bar_y, bar_width, bar_height))
    pygame.draw.rect(screen, GREEN, (bar_x, bar_y, fill, bar_height))
```

```
        draw_text(f"Score: {score}", small_font, WHITE, SCREEN_WIDTH // 2, 10)
        draw_text(f"Level: {level}", tiny_font, CYAN, 650, 10)
        draw_text(f"Coins: {coins_earned}", tiny_font, GOLD, 100, 10)
        player.draw_powerup_indicators(screen)

        if boss_active:
            draw_text("BOSS FIGHT!", small_font, RED, SCREEN_WIDTH // 2, SCREEN_HEIGHT - 40)

        achievement_notification.draw(screen)

        pygame.display.flip()

    save_game_data()
    return show_game_over_screen(score, coins_earned, level, new_achievements)

# --- Main Program ---
def main():
    """Main program loop"""
    load_game_data()

    while True:
        action = show_main_menu()

        if action == "quit":
            break
        elif action == "play":
            result = run_game()
            if result == "quit":
                break
        elif action == "skins":
            show_skin_shop()
```

```
        elif action == "achievements":
            show_achievements_screen()

    pygame.quit()

if __name__ == "__main__":
    main()
```

space_runner_save.json(file)

{"high_score": 5440, "coins": 846, "unlocked_skins": ["default"], "current_skin": "default", "achievements": ["first_blood", "coin_collector", "survivor", "power_user", "treasure_hunter", "millionaire", "speed_demon"], "stats": {"total_coins": 846, "bosses_defeated": 0, "powerups_collected": 36, "treasures_collected": 0, "max_speed_level": 19}}