Московский государственный технический университет им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

Выполнил:
студент группы ИУ5 - 32Б:
Васильев Н. Д.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2025 г.

ОГЛАВЛЕНИЕ

Задание для домашнего задания

1. Выберите язык программирования и реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Проект может быть посвящен отдельному аспекту (аспектам) языка или содержать
решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор) этого языка и произвольную среду разработки.
4. В случае создания проекта необходимо детально комментировать код.
5. При создании проекта необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.

Описание, листинг кода и скриншоты работы

Выбор языка и платформы:
В качестве языка программирования для реализации проекта был выбран **Python**. Разработка велась в интегрированной среде разработки (IDE) **PyCharm** с использованием библиотеки **Pygame** для создания мультимедийных приложений и игр. Выбор Python и Pygame обусловлен их высокой читаемостью, простотой реализации объектно-ориентированного подхода (ООП), а также широкими возможностями для работы с 2D-графикой, обработкой событий и физикой движения без использования тяжеловесных игровых движков.
Архитектура проекта: Проект структурирован в соответствии с классическим паттерном игрового цикла (Game Loop) и принципами ООП:

- **Инициализация и настройка окружения:** Импорт библиотек, задание глобальных констант (разрешение экрана, FPS, цветовая палитра) и настройка дисплея.
- **Реализация игровых сущностей (ООП):** Создание классов Player (игрок), EnemyShip (ИИ-враги), Mob (препятствия), Bullet (снаряды) и Particle (система частиц), наследуемых от базового класса pygame.sprite.Sprite.
- **Игровой цикл (Main Loop):**
  - **Обработка событий:** Считывание ввода с клавиатуры (WASD/Стрелки, стрельба) и системных событий.
  - **Обновление состояний (Update):** Расчет физики движения, проверка коллизий (столкновений), обновление системы частиц и логики врагов.
  - **Рендеринг (Draw):** Отрисовка кадров, процедурная генерация графики и отображение интерфейса (HUD).
- **Система сохранения данных:** Реализация функционала записи и чтения лучшего результата (High Score) через файловую систему.

Код:

```
import pygame
import random
import os

# --- КОНСТАНТЫ И НАСТРОЙКИ ---
WIDTH = 800
HEIGHT = 600
FPS = 60
POWERUP_TIME = 5000
BAR_LENGTH = 100
BAR_HEIGHT = 10
HS_FILE = "highscore.txt"  # Имя файла для сохранения рекорда

# Цвета
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 30, 30)
GREEN = (0, 255, 0)
BLUE = (0, 100, 255)
YELLOW = (255, 255, 0)
ORANGE = (255, 165, 0)
CYAN = (0, 255, 255)
PURPLE = (200, 0, 200)
```

```
# --- ФУНКЦИИ ЗАГРУЗКИ РЕКОРДА ---
def load_high_score():
    """Загружает рекорд из файла. Если файла нет - возвращает 0"""
    dir_path = os.path.dirname(__file__)
    file_path = os.path.join(dir_path, HS_FILE)
    try:
        with open(file_path, 'r') as f:
            return int(f.read())
    except:
        return 0


def save_high_score(score):
    """Сохраняет рекорд в файл"""
    dir_path = os.path.dirname(__file__)
    file_path = os.path.join(dir_path, HS_FILE)
    with open(file_path, 'w') as f:
        f.write(str(score))


# --- КЛАССЫ ---

class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface((50, 40))
        self.image.set_colorkey(BLACK)
        # Корпус
        pygame.draw.polygon(self.image, BLUE, [(0, 40), (25, 0), (50, 40)])
        pygame.draw.circle(self.image, CYAN, (25, 25), 8)
        # Двигатели
        pygame.draw.rect(self.image, ORANGE, (10, 38, 10, 5))
        pygame.draw.rect(self.image, ORANGE, (30, 38, 10, 5))

        self.rect = self.image.get_rect()
        self.radius = 20
        self.rect.centerx = WIDTH / 2
        self.rect.bottom = HEIGHT - 10
        self.speedx = 0
        self.speedy = 0
        self.shield = 100
        self.shoot_delay = 250
        self.last_shot = pygame.time.get_ticks()
        self.power = 1
        self.power_time = pygame.time.get_ticks()
        self.last_damage_time = pygame.time.get_ticks()

    def update(self):
        # Таймер улучшения
        if self.power >= 2 and pygame.time.get_ticks() - self.power_time >
POWERUP_TIME:
            self.power -= 1
            self.power_time = pygame.time.get_ticks()

        # Регенерация
        now = pygame.time.get_ticks()
        if now - self.last_damage_time > 3000 and self.shield < 100:
            self.shield += 0.5

        # Управление (ТЕПЕРЬ С WASD!)
        self.speedx = 0
        self.speedy = 0
```

```
        keystate = pygame.key.get_pressed()

        # Влево (Стрелка ИЛИ 'A')
        if keystate[pygame.K_LEFT] or keystate[pygame.K_a]:
            self.speedx = -8
        # Вправо (Стрелка ИЛИ 'D')
        if keystate[pygame.K_RIGHT] or keystate[pygame.K_d]:
            self.speedx = 8
        # Вверх (Стрелка ИЛИ 'W')
        if keystate[pygame.K_UP] or keystate[pygame.K_w]:
            self.speedy = -8
            self.spawn_exhaust()
        # Вниз (Стрелка ИЛИ 'S')
        if keystate[pygame.K_DOWN] or keystate[pygame.K_s]:
            self.speedy = 8

        self.rect.x += self.speedx
        self.rect.y += self.speedy

        # Ограничения экрана
        if self.rect.right > WIDTH: self.rect.right = WIDTH
        if self.rect.left < 0: self.rect.left = 0
        if self.rect.bottom > HEIGHT: self.rect.bottom = HEIGHT
        if self.rect.top < 0: self.rect.top = 0

    def spawn_exhaust(self):
        if random.random() > 0.5:
            p = Particle((self.rect.centerx - 10, self.rect.bottom), ORANGE, 2)
            all_sprites.add(p)
            p = Particle((self.rect.centerx + 10, self.rect.bottom), ORANGE, 2)
            all_sprites.add(p)

    def shoot(self):
        now = pygame.time.get_ticks()
        if now - self.last_shot > self.shoot_delay:
            self.last_shot = now
            if self.power == 1:
                bullet = Bullet(self.rect.centerx, self.rect.top, -10, YELLOW,
'player')
                all_sprites.add(bullet)
                bullets.add(bullet)
            if self.power >= 2:
                bullet1 = Bullet(self.rect.left, self.rect.centery, -10, YELLOW,
'player')
                bullet2 = Bullet(self.rect.right, self.rect.centery, -10, YELLOW,
'player')
                all_sprites.add(bullet1)
                all_sprites.add(bullet2)
                bullets.add(bullet1)
                bullets.add(bullet2)

    def powerup(self):
        self.power += 1
        self.power_time = pygame.time.get_ticks()

    def hit(self, damage):
        self.shield -= damage
        self.last_damage_time = pygame.time.get_ticks()


class EnemyShip(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface((40, 30))
```

```
        self.image.set_colorkey(BLACK)
        pygame.draw.ellipse(self.image, PURPLE, [0, 5, 40, 20])
        pygame.draw.circle(self.image, GREEN, (20, 15), 5)
        self.rect = self.image.get_rect()
        self.rect.x = random.randrange(WIDTH - self.rect.width)
        self.rect.y = random.randrange(-100, -40)
        self.speedy = random.randrange(1, 4)
        self.speedx = random.randrange(-2, 2)
        self.shoot_delay = random.randrange(1000, 3000)
        self.last_shot = pygame.time.get_ticks()

    def update(self):
        self.rect.x += self.speedx
        self.rect.y += self.speedy
        if self.rect.right > WIDTH or self.rect.left < 0:
            self.speedx *= -1
        now = pygame.time.get_ticks()
        if now - self.last_shot > self.shoot_delay:
            self.last_shot = now
            self.shoot()
        if self.rect.top > HEIGHT + 10:
            self.kill()

    def shoot(self):
        bullet = Bullet(self.rect.centerx, self.rect.bottom, 6, RED, 'enemy')
        all_sprites.add(bullet)
        enemy_bullets.add(bullet)


class Mob(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.width = random.randrange(30, 50)
        self.image = pygame.Surface((self.width, self.width))
        self.image.set_colorkey(BLACK)
        pygame.draw.circle(self.image, RED, (self.width // 2, self.width // 2),
self.width // 2)
        pygame.draw.circle(self.image, (150, 50, 50), (self.width // 2 - 5,
self.width // 2 - 5), self.width // 4)
        self.rect = self.image.get_rect()
        self.radius = int(self.rect.width * .85 / 2)
        self.rect.x = random.randrange(WIDTH - self.rect.width)
        self.rect.y = random.randrange(-150, -100)
        self.speedy = random.randrange(2, 8)
        self.speedx = random.randrange(-3, 3)

    def update(self):
        self.rect.x += self.speedx
        self.rect.y += self.speedy
        if self.rect.top > HEIGHT + 10 or self.rect.left < -50 or self.rect.right
> WIDTH + 50:
            self.rect.x = random.randrange(WIDTH - self.rect.width)
            self.rect.y = random.randrange(-100, -40)
            self.speedy = random.randrange(2, 8)


class Bullet(pygame.sprite.Sprite):
    def __init__(self, x, y, speed, color, owner):
        super().__init__()
        self.image = pygame.Surface((6, 15))
        self.image.set_colorkey(BLACK)
        pygame.draw.ellipse(self.image, color, [0, 0, 6, 15])
        self.rect = self.image.get_rect()
        self.rect.bottom = y
```

```
        self.rect.centerx = x
        self.speedy = speed
        self.owner = owner

    def update(self):
        self.rect.y += self.speedy
        if self.rect.bottom < 0 or self.rect.top > HEIGHT:
            self.kill()


class Particle(pygame.sprite.Sprite):
    def __init__(self, center, color, size_range=8):
        super().__init__()
        self.size = random.randint(2, size_range)
        self.image = pygame.Surface((self.size, self.size))
        self.image.set_colorkey(BLACK)
        pygame.draw.circle(self.image, color, (self.size // 2, self.size // 2),
self.size // 2)
        self.rect = self.image.get_rect()
        self.rect.center = center
        self.vel_x = random.uniform(-2, 2)
        self.vel_y = random.uniform(-2, 2)
        self.lifetime = 30

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


class PowerUp(pygame.sprite.Sprite):
    def __init__(self, center):
        super().__init__()
        self.type = random.choice(['shield', 'gun'])
        self.image = pygame.Surface((20, 20))
        self.image.set_colorkey(BLACK)
        if self.type == 'shield':
            color = GREEN
            pygame.draw.circle(self.image, color, (10, 10), 10)
            pygame.draw.line(self.image, WHITE, (10, 3), (10, 17), 2)
            pygame.draw.line(self.image, WHITE, (3, 10), (17, 10), 2)
        else:
            color = YELLOW
            pygame.draw.rect(self.image, color, (5, 5, 10, 10))
        self.rect = self.image.get_rect()
        self.rect.center = center
        self.speedy = 3

    def update(self):
        self.rect.y += self.speedy
        if self.rect.top > HEIGHT:
            self.kill()


class Star():
    def __init__(self):
        self.x = random.randint(0, WIDTH)
        self.y = random.randint(0, HEIGHT)
        self.speed = random.uniform(0.5, 3.0)
        self.size = random.randint(1, 3)
        shade = int(255 * (self.speed / 3))
        self.color = (shade, shade, shade)
```

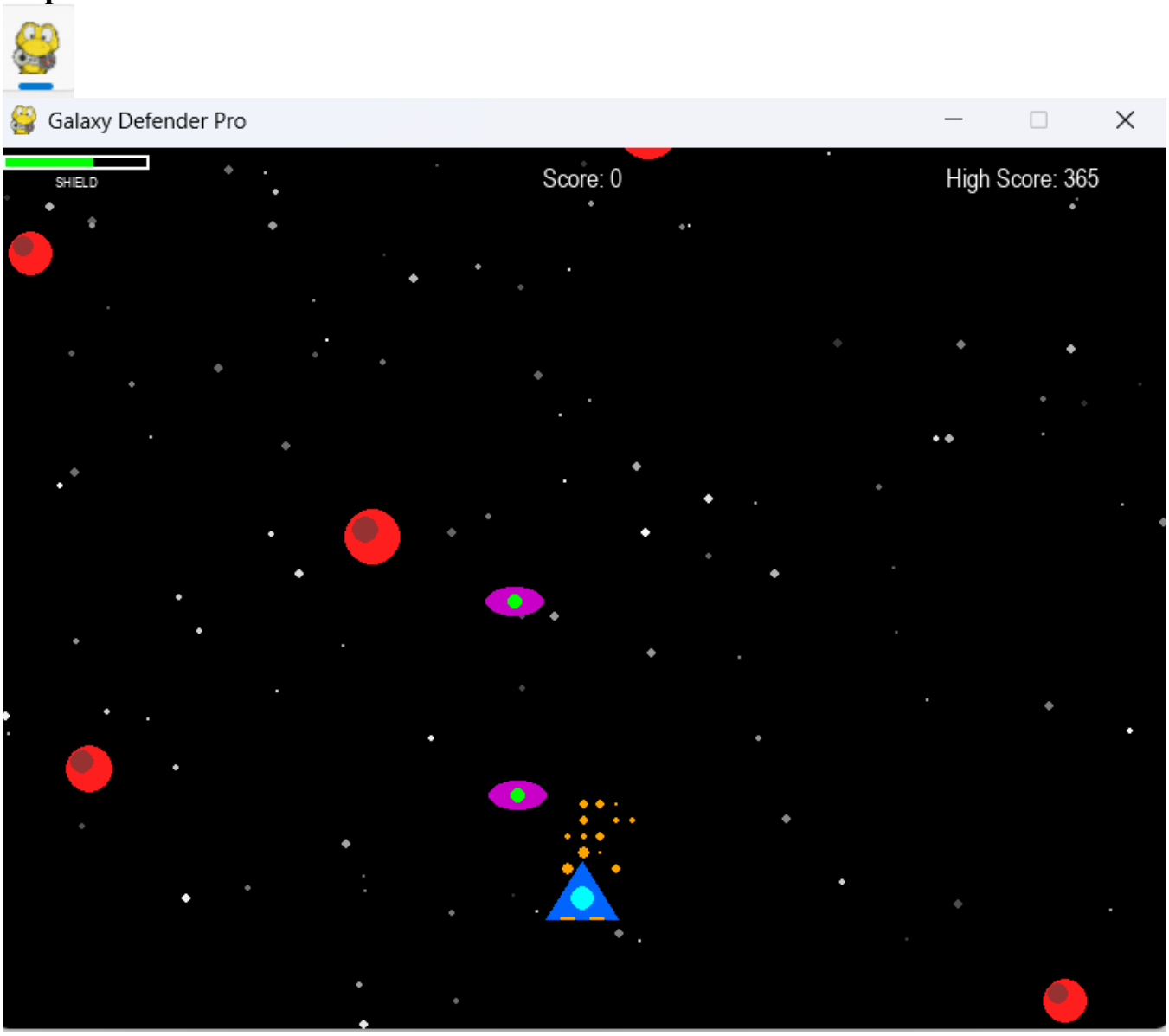
```
    def update(self):
        self.y += self.speed
        if self.y > HEIGHT:
            self.y = 0
            self.x = random.randint(0, WIDTH)

    def draw(self, screen):
        pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)),
self.size)


# --- ФУНКЦИИ ОТРИСОВКИ ---

def draw_text(surf, text, size, x, y):
    font = pygame.font.Font(font_name, size)
    text_surface = font.render(text, True, WHITE)
    text_rect = text_surface.get_rect()
    text_rect.midtop = (x, y)
    surf.blit(text_surface, text_rect)


def draw_shield_bar(surf, x, y, pct):
    if pct < 0: pct = 0
    fill = (pct / 100) * BAR_LENGTH
    outline_rect = pygame.Rect(x, y, BAR_LENGTH, BAR_HEIGHT)
    fill_rect = pygame.Rect(x, y, fill, BAR_HEIGHT)
    pygame.draw.rect(surf, GREEN, fill_rect)
    pygame.draw.rect(surf, WHITE, outline_rect, 2)
    draw_text(surf, "SHIELD", 10, x + BAR_LENGTH / 2, y + 12)


def show_go_screen(score, highscore):
    screen.fill(BLACK)
    draw_text(screen, "GALAXY DEFENDER", 64, WIDTH / 2, HEIGHT / 4)
    draw_text(screen, f"Твой счет: {score}", 22, WIDTH / 2, HEIGHT / 2)
    draw_text(screen, f"Рекорд: {highscore}", 22, WIDTH / 2, HEIGHT / 2 + 40)

    draw_text(screen, "WASD / Стрелки - лететь", 18, WIDTH / 2, HEIGHT * 3 / 4)
    draw_text(screen, "Нажми кнопку для старта", 18, WIDTH / 2, HEIGHT * 3 / 4 +
30)

    pygame.display.flip()
    waiting = True
    while waiting:
        clock.tick(FPS)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                exit()
            if event.type == pygame.KEYUP:
                waiting = False


# --- ИНИЦИАЛИЗАЦИЯ ---

pygame.init()
pygame.mixer.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Galaxy Defender Pro")
clock = pygame.time.Clock()
font_name = pygame.font.match_font('arial')

# Загружаем рекорд один раз при запуске
```

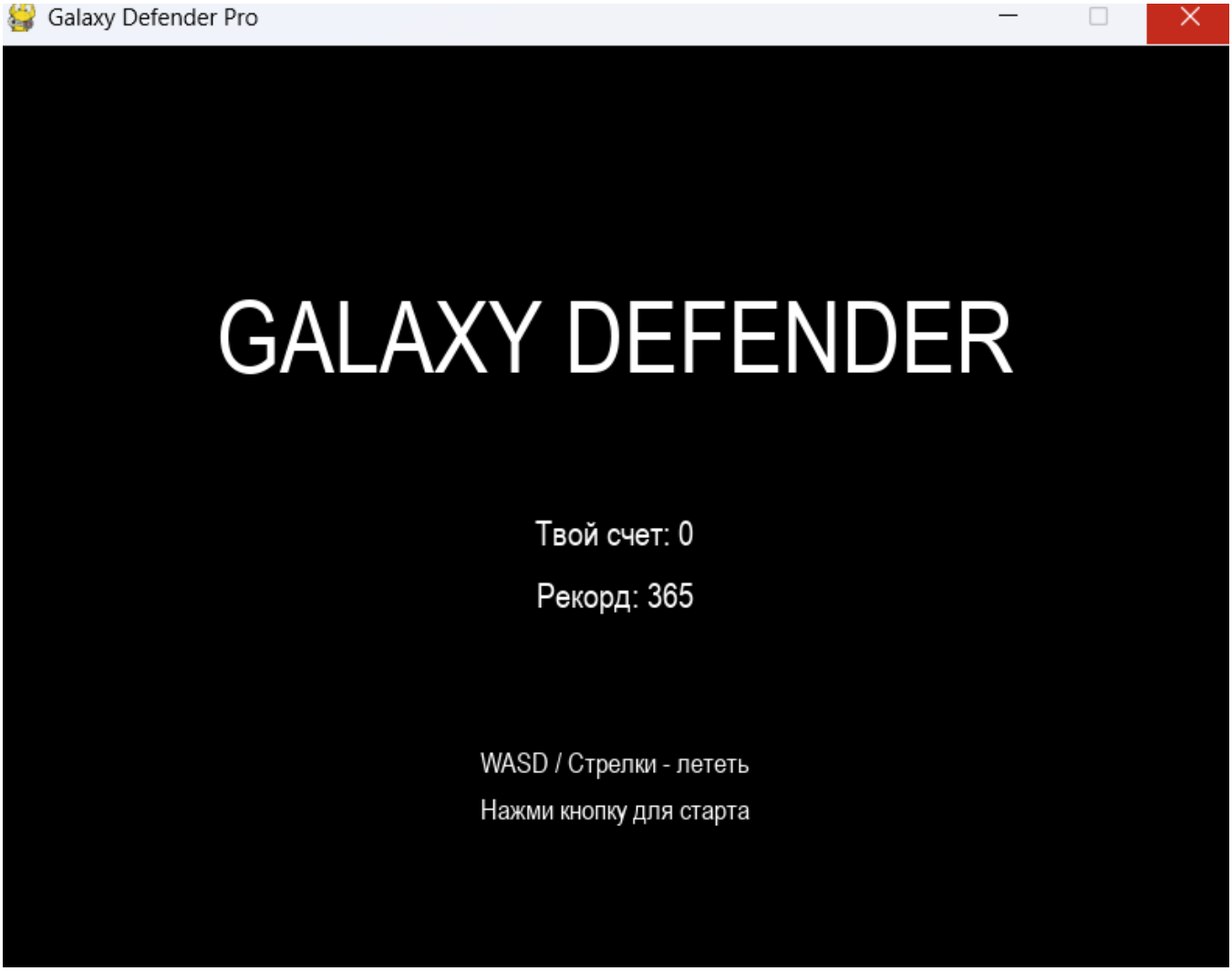
```
highscore = load_high_score()

# --- ГЛАВНЫЙ ЦИКЛ ---
game_over = True
running = True
stars = [Star() for _ in range(100)]
score = 0

while running:
    if game_over:
        # Если побили рекорд - сохраняем
        if score > highscore:
            highscore = score
            save_high_score(highscore)

        show_go_screen(score, highscore)

        game_over = False
        all_sprites = pygame.sprite.Group()
        mobs = pygame.sprite.Group()
        bullets = pygame.sprite.Group()
        enemy_bullets = pygame.sprite.Group()
        enemies = pygame.sprite.Group()
        powerups = pygame.sprite.Group()
        particles = pygame.sprite.Group()

        player = Player()
        all_sprites.add(player)
        for i in range(5):
            m = Mob()
            all_sprites.add(m)
            mobs.add(m)
        for i in range(2):
            e = EnemyShip()
            all_sprites.add(e)
            enemies.add(e)

        score = 0

    clock.tick(FPS)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                player.shoot()

    all_sprites.update()
    for star in stars: star.update()

    # --- СТОЛКНОВЕНИЯ ---

    # 1. Попадание в астероид
    hits = pygame.sprite.groupcollide(mobs, bullets, True, True)
    for hit in hits:
        score += 50 - hit.radius
        for _ in range(15):
            p = Particle(hit.rect.center, RED)
            all_sprites.add(p)
        if random.random() > 0.9:
            pow = PowerUp(hit.rect.center)
            all_sprites.add(pow)
            powerups.add(pow)
```

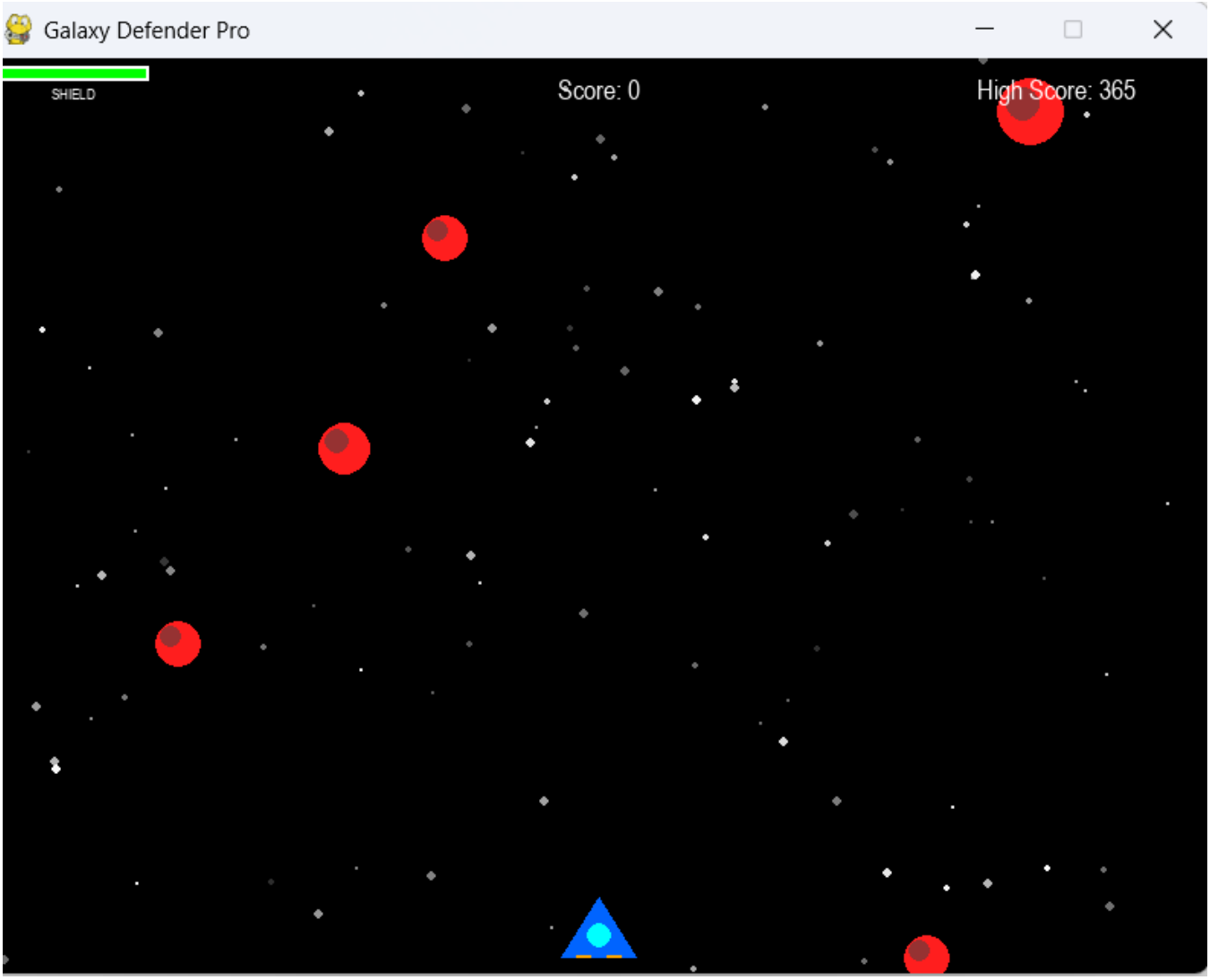
```
            m = Mob()
            all_sprites.add(m)
            mobs.add(m)

        # 2. Попадание в врага
        hits = pygame.sprite.groupcollide(enemies, bullets, True, True)
        for hit in hits:
            score += 100
            for _ in range(25):
                p = Particle(hit.rect.center, PURPLE, 10)
                all_sprites.add(p)
            if random.random() > 0.8:
                pow = PowerUp(hit.rect.center)
                all_sprites.add(pow)
                powerups.add(pow)
            e = EnemyShip()
            all_sprites.add(e)
            enemies.add(e)

        # 3. Игрок врезался
        hits = pygame.sprite.spritecollide(player, mobs, True,
pygame.sprite.collide_circle)
        for hit in hits:
            player.hit(hit.radius * 2)
            for _ in range(20):
                p = Particle(hit.rect.center, ORANGE)
                all_sprites.add(p)
            m = Mob()
            all_sprites.add(m)
            mobs.add(m)
            if player.shield <= 0: game_over = True

        # 4. Игрока подстрелили
        hits = pygame.sprite.spritecollide(player, enemy_bullets, True)
        for hit in hits:
            player.hit(25)
            p = Particle(player.rect.center, CYAN, 5)
            all_sprites.add(p)
            if player.shield <= 0: game_over = True

        # 5. Таран
        hits = pygame.sprite.spritecollide(player, enemies, True)
        for hit in hits:
            player.hit(50)
            e = EnemyShip()
            all_sprites.add(e)
            enemies.add(e)
            if player.shield <= 0: game_over = True

        hits = pygame.sprite.spritecollide(player, powerups, True)
        for hit in hits:
            if hit.type == 'shield':
                player.shield += 20
                if player.shield > 100: player.shield = 100
            if hit.type == 'gun':
                player.powerup()

        # --- ОТРИСОВКА ---
        screen.fill(BLACK)
        for star in stars: star.draw(screen)
        all_sprites.draw(screen)

        # Счет слева
        draw_text(screen, f"Score: {score}", 18, WIDTH / 2, 10)
```

```
    # Рекорд справа
    draw_text(screen, f"High Score: {highscore}", 18, WIDTH - 100, 10)

    draw_shield_bar(screen, 5, 5, player.shield)
    pygame.display.flip()

# Финальное сохранение при выходе
if score > highscore:
    save_high_score(score)

pygame.quit()
```

Скрины Работы:




Galaxy Defender Pro
GALAXY DEFENDER
Твой счет: 0
Рекорд: 365
WASD / Стрелки - лететь
Нажми кнопку для старта


Galaxy Defender Pro
SHIELD
Score: 0
High Score: 365