|  |
| --- |
| Изображение выглядит как рисунок  Автоматически созданное описание |
| МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт кибербезопасности и цифровых технологий

Кафедра КБ-14 «Цифровые технологии обработки данных»

**ПРОЕКТ (РАБОТА)**

**по дисциплине**

«Интерпретируемый язык программирования высокого уровня»

наименование дисциплины

**Тема курсового проекта (работы**) «Игра “Бродилка-стрелялка”»

Студент группы БСБО-06-21 Ефимов А.Р.

(учебная группа) Фамилия И.О.

Руководитель курсового проекта (работы) Тарланов А.

Доцент кафедры кб – 14, к.т.н. Фамилия И.О.

Работа представлена к защите «\_\_»\_\_\_\_\_\_ 2023г. подпись студента

Допущен к защите «\_\_»\_\_\_\_\_\_ 2023г.

подпись руководителя

Москва 2023 г.

1. Задание проекта

Игровой проект по типу бродилка стрелялка, где игрок( персонаж или какой- нибудь объект) будет сражаться с NPC и например уворачиваться от препятствий. Проект буду делать на 500 и более строк и один.

1. Програмный код с необходимыми комментариями и пояснениями.

Проект содержит необходимые изображения и звуки, 3 JSON файла с таблицами рекордов, а также сам код программы.

Код программы (файл main.py):

import json  
import random  
from os import path  
  
import pygame  
from pygame import freetype  
  
img\_dir = path.join(  
 path.dirname("Необходимые файлы\\изображения\\"),  
 'PNG')  
img\_dir\_1 = path.join(  
 path.dirname("Необходимые файлы\\изображения\\PNG\\"),  
 'Meteors')  
img\_dir\_2 = path.join(  
 path.dirname("Необходимые файлы\\изображения\\PNG\\"),  
 'Lasers')  
img\_dir\_3 = path.join(  
 path.dirname("Необходимые файлы\\изображения\\"),  
 'Взрывы')  
img\_dir\_4 = path.join(path.dirname("Необходимые файлы\\"),  
 'изображения')  
img\_dir\_5 = path.join(  
 path.dirname("Необходимые файлы\\изображения\\PNG\\"),  
 'Power-ups')  
  
snd\_dir = path.join(path.dirname("Необходимые файлы\\"),  
 'звук')  
  
WIDTH = 800  
HEIGHT = 800  
FPS = 60  
POWERUP\_TIME = 5000  
  
# Задаем цвета  
WHITE = (255, 255, 255)  
BLACK = (0, 0, 0)  
RED = (255, 0, 0)  
GREEN = (0, 255, 0)  
BLUE = (0, 0, 255)  
YELLOW = (255, 255, 0)  
  
# Создаем игру и окно  
pygame.init()  
pygame.mixer.init()  
screen = pygame.display.set\_mode((WIDTH, HEIGHT))  
pygame.display.set\_caption("My Project")  
clock = pygame.time.Clock()  
  
font\_name = pygame.font.match\_font('arial')  
  
BG\_COLOR = pygame.Color('gray12')  
BLUE = pygame.Color('dodgerblue')  
FONT = freetype.Font(None, 24)  
  
  
def save(highscores):  
 with open('highscores.json', 'w') as file:  
 json.dump(highscores, file) # Write the list to the json file.  
  
  
def save\_1(highscores\_1):  
 with open('highscores\_1.json', 'w') as file:  
 json.dump(highscores\_1, file) # Write the list to the json file.  
  
  
def save\_2(highscores\_2):  
 with open('highscores\_2.json', 'w') as file:  
 json.dump(highscores\_2, file) # Write the list to the json file.  
  
  
def save\_3(highscores\_3):  
 with open('highscores\_3.json', 'w') as file:  
 json.dump(highscores\_3, file) # Write the list to the json file.  
  
  
def load\_1():  
 try:  
 with open('highscores\_1.json', 'r') as file:  
 highscores\_1 = json.load(file) # Read the json file.  
 except FileNotFoundError:  
 return [] # Return an empty list if the file doesn't exist.  
 # Sorted by the score.  
 return sorted(highscores\_1, reverse=True)  
  
  
def load\_2():  
 try:  
 with open('highscores\_2.json', 'r') as file:  
 highscores\_2 = json.load(file) # Read the json file.  
 except FileNotFoundError:  
 return [] # Return an empty list if the file doesn't exist.  
 # Sorted by the score.  
 return sorted(highscores\_2, reverse=True)  
  
  
def load\_3():  
 try:  
 with open('highscores\_3.json', 'r') as file:  
 highscores\_3 = json.load(file) # Read the json file.  
 except FileNotFoundError:  
 return [] # Return an empty list if the file doesn't exist.  
 # Sorted by the score.  
 return sorted(highscores\_3, reverse=True)  
  
  
def tabl\_record\_1():  
 screen.blit(background, background\_rect)  
 # Display the high-scores.  
 k = 1  
 FONT.render\_to(screen, (100, 50), f'Таблица рекордов', BLUE)  
 FONT.render\_to(screen, (100, 100), f'Чтобы выйти нажмите ESCAPE', BLUE)  
 FONT.render\_to(screen, (100, 150), f'Уровень - 1', BLUE)  
 for y, (hi\_score) in enumerate(sorted(highscores\_1, reverse=True)):  
 FONT.render\_to(screen, (100, y \* 30 + 190), f'{k} - {hi\_score}', BLUE)  
 k += 1  
 pygame.display.flip()  
 clock.tick(60)  
 waiting = True  
 while waiting:  
 clock.tick(FPS)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_ESCAPE:  
 waiting = False  
 show\_choose\_game\_screen()  
  
  
def tabl\_record\_2():  
 screen.blit(background, background\_rect)  
 # Display the high-scores.  
 k = 1  
 FONT.render\_to(screen, (100, 50), f'Таблица рекордов', BLUE)  
 FONT.render\_to(screen, (100, 100), f'Чтобы выйти нажмите ESCAPE', BLUE)  
 FONT.render\_to(screen, (100, 150), f'Уровень - 2', BLUE)  
 for y, (hi\_score) in enumerate(sorted(highscores\_2, reverse=True)):  
 FONT.render\_to(screen, (100, y \* 30 + 190), f'{k} - {hi\_score}', BLUE)  
 k += 1  
 pygame.display.flip()  
 clock.tick(60)  
 waiting = True  
 while waiting:  
 clock.tick(FPS)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_ESCAPE:  
 waiting = False  
 show\_choose\_game\_screen()  
  
  
def tabl\_record\_3():  
 screen.blit(background, background\_rect)  
 # Display the high-scores.  
 k = 1  
 FONT.render\_to(screen, (100, 50), f'Таблица рекордов', BLUE)  
 FONT.render\_to(screen, (100, 100), f'Чтобы выйти нажмите ESCAPE', BLUE)  
 FONT.render\_to(screen, (100, 150), f'Уровень - 3', BLUE)  
 for y, (hi\_score) in enumerate(sorted(highscores\_3, reverse=True)):  
 FONT.render\_to(screen, (100, y \* 30 + 190), f'{k} - {hi\_score}', BLUE)  
 k += 1  
 pygame.display.flip()  
 clock.tick(60)  
 waiting = True  
 while waiting:  
 clock.tick(FPS)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_ESCAPE:  
 waiting = False  
 show\_choose\_game\_screen()  
  
  
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 newmob\_ship\_high():  
 m = Mob\_ship\_high()  
 all\_sprites.add(m)  
 mobs\_ship\_high.add(m)  
  
  
def newmob\_ship\_down():  
 m = Mob\_ship\_down()  
 all\_sprites.add(m)  
 mobs\_ship\_down.add(m)  
  
  
def newmob\_meteor():  
 m = Mob\_meteor()  
 all\_sprites.add(m)  
 mobs\_meteors.add(m)  
  
  
def newmob\_ufo():  
 m = Mob\_ufo()  
 all\_sprites.add(m)  
 mobs\_ufo.add(m)  
  
  
def draw\_shield\_bar(surf, x, y, pct):  
 if pct < 0:  
 pct = 0  
 BAR\_LENGTH = 100  
 BAR\_HEIGHT = 10  
 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)  
  
  
def draw\_lives(surf, x, y, lives, img):  
 for i in range(lives):  
 img\_rect = img.get\_rect()  
 img\_rect.x = x + 30 \* i  
 img\_rect.y = y  
 surf.blit(img, img\_rect)  
  
  
def show\_end\_screen(score):  
 screen.blit(background, background\_rect)  
 draw\_text(screen, "Игра окончена, вы набрали " + str(score) + " очков", 18, WIDTH / 2, HEIGHT \* 1 / 4)  
 draw\_text(screen, "Нажмите ESCAPE, чтобы выйти", 18, WIDTH / 2, HEIGHT \* 2 / 4)  
 draw\_text(screen, "Нажмите 1,2 или 3, чтобы подтвердить, какой уровень был пройден и сохранить счёт", 18, WIDTH / 2,  
 HEIGHT \* 3 / 4)  
 pygame.display.flip()  
 waiting = True  
 while waiting:  
 clock.tick(FPS)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_ESCAPE:  
 waiting = False  
 if event.type == pygame.KEYUP and event.key == pygame.K\_1:  
 highscores\_1.append(score)  
 save\_1(sorted(highscores\_1, reverse=True))  
 waiting = False  
 if event.type == pygame.KEYUP and event.key == pygame.K\_2:  
 highscores\_2.append(score)  
 save\_2(sorted(highscores\_2, reverse=True))  
 waiting = False  
 if event.type == pygame.KEYUP and event.key == pygame.K\_3:  
 highscores\_3.append(score)  
 save\_3(sorted(highscores\_3, reverse=True))  
 waiting = False  
  
  
def show\_choose\_game\_screen():  
 screen = pygame.display.set\_mode((WIDTH, HEIGHT))  
 draw\_text(screen, "Нажмите g, чтобы начать игру", 18, WIDTH / 2, HEIGHT \* 1 / 5)  
 draw\_text(screen, "Нажмите 1,2 или 3, чтобы увидеть таблицу рекордов этого уровня", 18, WIDTH / 2, HEIGHT \* 3 / 5)  
 draw\_text(screen, "Нажмите O сейчас или во время ишры, чтобы выйти", 18, WIDTH / 2, HEIGHT \* 4 / 5)  
 pygame.display.flip()  
 waiting = True  
 while waiting:  
 clock.tick(FPS)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT or (event.type == pygame.KEYUP and event.key == pygame.K\_o):  
 pygame.quit()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_g:  
 waiting = False  
 show\_go\_screen()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_1:  
 waiting = False  
 tabl\_record\_1()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_2:  
 waiting = False  
 tabl\_record\_2()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_3:  
 waiting = False  
 tabl\_record\_3()  
  
  
def show\_go\_screen():  
 screen.blit(background, background\_rect)  
 draw\_text(screen, "My Project!", 64, WIDTH / 2, HEIGHT / 7)  
 draw\_text(screen, "Стрелки - перемещают корабль, "  
 "пробел - позволяет стрелять", 22,  
 WIDTH / 2, HEIGHT \* 2 / 7)  
 draw\_text(screen, "Нажмите 1, чтобы выбрать первый уровень и начать", 18, WIDTH / 2, HEIGHT \* 3 / 7)  
 draw\_text(screen, "Нажмите 2, чтобы выбрать второй уровень и начать", 18, WIDTH / 2, HEIGHT \* 4 / 7)  
 draw\_text(screen, "Нажмите 3, чтобы выбрать третий уровень и начать", 18, WIDTH / 2, HEIGHT \* 5 / 7)  
 draw\_text(screen, "Нажмить ESCAPE, чтобы вернуться назад", 18, WIDTH / 2, HEIGHT \* 6 / 7)  
 pygame.display.flip()  
 waiting = True  
 while waiting:  
 clock.tick(FPS)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_ESCAPE:  
 waiting = False  
 show\_choose\_game\_screen()  
 if event.type == pygame.KEYUP and event.key == pygame.K\_1:  
 for i in range(8):  
 newmob\_meteor()  
 waiting = False  
  
 if event.type == pygame.KEYUP and event.key == pygame.K\_2:  
 for i in range(4):  
 newmob\_ufo()  
 waiting = False  
  
 if event.type == pygame.KEYUP and event.key == pygame.K\_3:  
 for i in range(3):  
 newmob\_ship\_high()  
 newmob\_ship\_down()  
 waiting = False  
  
  
class Player(pygame.sprite.Sprite):  
 def \_\_init\_\_(self):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = pygame.transform.scale(player\_img, (50, 38))  
 self.image1 = pygame.transform.scale(player\_img, (50, 38))  
 self.image2 = pygame.transform.rotate(self.image1, 180)  
 self.naprav = 1  
 self.image.set\_colorkey(BLACK)  
 self.rect = self.image.get\_rect()  
 self.radius = 20  
 # pygame.draw.circle(self.image, RED, self.rect.center, self.radius)  
 self.rect.centerx = WIDTH / 2  
 self.rect.bottom = HEIGHT / 2  
 self.speedx = 0  
 self.speedy = 0  
 self.shield = 100  
 self.shoot\_delay = 250  
 self.last\_shot = pygame.time.get\_ticks()  
 self.lives = 3  
 self.hidden = False  
 self.hide\_timer = pygame.time.get\_ticks()  
 self.power = 1  
 self.power\_time = pygame.time.get\_ticks()  
  
 def update(self):  
 # timeout for powerups  
 if self.power >= 2 and pygame.time.get\_ticks() - self.power\_time > POWERUP\_TIME:  
 self.power -= 1  
 self.power\_time = pygame.time.get\_ticks()  
 # показать, если скрыто  
 if self.hidden and pygame.time.get\_ticks() - self.hide\_timer > 1000:  
 self.hidden = False  
 self.rect.centerx = WIDTH / 2  
 self.rect.bottom = HEIGHT / 2  
  
 self.speedx = 0  
 self.speedy = 0  
 keystate = pygame.key.get\_pressed()  
 if keystate[pygame.K\_LEFT]:  
 self.speedx = -8  
 if keystate[pygame.K\_RIGHT]:  
 self.speedx = 8  
 if keystate[pygame.K\_UP]:  
 self.speedy = -8  
 if keystate[pygame.K\_DOWN]:  
 self.speedy = 8  
 if keystate[pygame.K\_SPACE]:  
 self.shoot()  
 if self.rect.right > WIDTH:  
 self.rect.right = WIDTH  
 if self.rect.left < 0:  
 self.rect.left = 0  
 self.rect.x += self.speedx  
 self.rect.y += self.speedy  
 if self.rect.top < 0:  
 self.rect.top = 0  
 if self.rect.bottom > HEIGHT and self.hidden == False:  
 self.rect.bottom = HEIGHT  
 if self.speedy < 0:  
 self.image = self.image1  
 self.image.set\_colorkey(BLACK)  
 self.naprav = 1  
 elif self.speedy > 0:  
 self.image = self.image2  
 self.image.set\_colorkey(BLACK)  
 self.naprav = 2  
  
 def powerup(self):  
 self.power += 1  
 self.power\_time = pygame.time.get\_ticks()  
  
 def shoot(self):  
 now = pygame.time.get\_ticks()  
 if now - self.last\_shot > self.shoot\_delay:  
 self.last\_shot = now  
 if self.power == 1:  
 if self.speedy > 0:  
 if self.naprav == 2:  
 bullet = Bullet(self.rect.centerx, self.rect.top, 3)  
 all\_sprites.add(bullet)  
 bullets\_3.add(bullet)  
 else:  
 bullet = Bullet(self.rect.centerx, self.rect.top, 1)  
 all\_sprites.add(bullet)  
 bullets\_1.add(bullet)  
 shoot\_sound.play()  
  
 elif self.speedy == 0:  
 if self.naprav == 2:  
 bullet = Bullet(self.rect.centerx, self.rect.top, 3)  
 all\_sprites.add(bullet)  
 bullets\_3.add(bullet)  
 else:  
 bullet = Bullet(self.rect.centerx, self.rect.top, 1)  
 all\_sprites.add(bullet)  
 bullets\_1.add(bullet)  
 shoot\_sound.play()  
  
 if self.power >= 2:  
 if self.speedy > 0:  
 if self.naprav == 2:  
 bullet1 = Bullet(self.rect.left, self.rect.centery, 3)  
 bullet2 = Bullet(self.rect.right, self.rect.centery, 3)  
 all\_sprites.add(bullet1)  
 all\_sprites.add(bullet2)  
 bullets\_3.add(bullet1)  
 bullets\_3.add(bullet2)  
 else:  
 bullet1 = Bullet(self.rect.left, self.rect.centery, 1)  
 bullet2 = Bullet(self.rect.right, self.rect.centery, 1)  
 all\_sprites.add(bullet1)  
 all\_sprites.add(bullet2)  
 bullets\_1.add(bullet1)  
 bullets\_1.add(bullet2)  
 shoot\_sound.play()  
  
 elif self.speedy == 0:  
 if self.naprav == 2:  
 bullet1 = Bullet(self.rect.left, self.rect.centery, 3)  
 bullet2 = Bullet(self.rect.right, self.rect.centery, 3)  
 all\_sprites.add(bullet1)  
 all\_sprites.add(bullet2)  
 bullets\_3.add(bullet1)  
 bullets\_3.add(bullet2)  
 else:  
 bullet1 = Bullet(self.rect.left, self.rect.centery, 1)  
 bullet2 = Bullet(self.rect.right, self.rect.centery, 1)  
 all\_sprites.add(bullet1)  
 all\_sprites.add(bullet2)  
 bullets\_1.add(bullet1)  
 bullets\_1.add(bullet2)  
 shoot\_sound.play()  
  
 def hide(self):  
 # временно скрыть игрока  
 self.hidden = True  
 self.hide\_timer = pygame.time.get\_ticks()  
 self.rect.center = (WIDTH / 2, HEIGHT + 200)  
  
  
class Mob\_meteor(pygame.sprite.Sprite):  
 def \_\_init\_\_(self):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image\_orig = random.choice(meteor\_images)  
 self.image\_orig.set\_colorkey(BLACK)  
 self.image = self.image\_orig.copy()  
  
 self.rect = self.image.get\_rect()  
 self.speedy = random.randrange(1, 4)  
 self.speedx = random.randrange(-3, 3)  
 self.radius = int(self.rect.width \* .85 / 2)  
 # pygame.draw.circle(self.image, RED, self.rect.center, self.radius)  
 self.speedy = random.randrange(1, 8)  
 self.speedx = random.randrange(-3, 3)  
 self.rect.x = random.randrange(WIDTH - self.rect.width)  
 self.rect.y = random.randrange(-150, -100)  
  
 self.rot = 0  
 self.rot\_speed = random.randrange(-8, 8)  
 self.last\_update = pygame.time.get\_ticks()  
  
 def rotate(self):  
 now = pygame.time.get\_ticks()  
 if now - self.last\_update > 50:  
 self.last\_update = now  
 self.rot = (self.rot + self.rot\_speed) % 360  
 new\_image = pygame.transform.rotate(self.image\_orig, self.rot)  
 old\_center = self.rect.center  
 self.image = new\_image  
 self.rect = self.image.get\_rect()  
 self.rect.center = old\_center  
  
 def change\_speed(self):  
 self.speedy = -self.speedy  
 self.speedx = -self.speedx  
  
 def update(self):  
 self.rotate()  
 self.rect.x += self.speedx  
 self.rect.y += self.speedy  
 if self.rect.top > HEIGHT + 10 or self.rect.left < -25 or self.rect.right > WIDTH + 20:  
 self.rect.x = random.randrange(WIDTH - self.rect.width)  
 self.rect.y = random.randrange(-100, -40)  
 self.speedy = random.randrange(1, 8)  
  
 def shoot(self):  
 pass  
  
  
class Mob\_ufo(pygame.sprite.Sprite):  
 def \_\_init\_\_(self):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image\_orig = random.choice(ufo\_images)  
 self.image\_orig.set\_colorkey(BLACK)  
 self.image = pygame.transform.scale(self.image\_orig, (40, 40))  
  
 self.rect = self.image.get\_rect()  
 self.speedy = random.randrange(1, 4)  
 self.speedx = random.randrange(-3, 3)  
 self.radius = int(self.rect.width \* .85 / 2)  
 # pygame.draw.circle(self.image, RED, self.rect.center, self.radius)  
 self.speedy = random.randrange(1, 4)  
 self.speedx = random.randrange(-3, 3)  
 self.rect.x = random.randrange(WIDTH - self.rect.width)  
 self.rect.y = random.randrange(-150, -100)  
 self.predx = random.randrange(WIDTH - self.rect.width)  
 self.predy = random.randrange(-150, -100)  
  
 self.rot = 0  
 self.rot\_speed = random.randrange(-8, 8)  
 self.last\_update = pygame.time.get\_ticks()  
  
 def rotate(self):  
 pass  
  
 def change\_speed(self):  
 self.speedy = -self.speedy  
 self.speedx = -self.speedx  
  
 def update(self):  
 self.rect.x += self.speedx  
 self.rect.y += self.speedy  
 if self.rect.bottom > 300:  
 self.speedy = -self.speedy  
 if self.rect.left < -25 or self.rect.right > WIDTH + 20:  
 self.speedx = -self.speedx  
 if self.rect.top < 0 and self.speedy < 0:  
 self.speedy = -self.speedy  
 if abs(self.rect.x - self.predx) + abs(self.rect.y - self.predy) > 200:  
 self.shoot()  
 self.predx = self.rect.x  
 self.predy = self.rect.y  
  
 def shoot(self):  
 bullet = Bullet(self.rect.centerx, self.rect.bottom, 2)  
 all\_sprites.add(bullet)  
 bullets\_2.add(bullet)  
 shoot\_sound.play()  
  
  
class Mob\_ship\_high(pygame.sprite.Sprite):  
 def \_\_init\_\_(self):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image\_orig = random.choice(ship2\_images)  
 self.image\_orig.set\_colorkey(BLACK)  
 self.image\_orig = pygame.transform.rotate(self.image\_orig, 180)  
 self.image = pygame.transform.scale(self.image\_orig, (40, 40))  
  
 self.rect = self.image.get\_rect()  
 self.speedy = random.randrange(1, 4)  
 self.speedx = random.randrange(-3, 3)  
 self.radius = int(self.rect.width \* .85 / 2)  
 # pygame.draw.circle(self.image, RED, self.rect.center, self.radius)  
 self.speedy = random.randrange(1, 4)  
 self.speedx = random.randrange(-3, 3)  
 self.rect.x = random.randrange(WIDTH - self.rect.width)  
 self.rect.y = random.randrange(-150, -100)  
 self.predx = random.randrange(WIDTH - self.rect.width)  
 self.predy = random.randrange(-150, -100)  
  
 self.rot = 0  
 self.rot\_speed = random.randrange(-8, 8)  
 self.last\_update = pygame.time.get\_ticks()  
  
 def rotate(self):  
 pass  
  
 def change\_speed(self):  
 self.speedy = -self.speedy  
 self.speedx = -self.speedx  
  
 def update(self):  
 self.rect.x += self.speedx  
 self.rect.y += self.speedy  
 if self.rect.bottom > 200:  
 self.speedy = -self.speedy  
 if self.rect.left < -25 or self.rect.right > WIDTH + 20:  
 self.speedx = -self.speedx  
 if self.rect.top < 0 and self.speedy < 0:  
 self.speedy = -self.speedy  
 if abs(self.rect.x - self.predx) + abs(self.rect.y - self.predy) > 200:  
 self.shoot()  
 self.predx = self.rect.x  
 self.predy = self.rect.y  
  
 def shoot(self):  
 bullet = Bullet(self.rect.centerx, self.rect.bottom, 2)  
 all\_sprites.add(bullet)  
 bullets\_2.add(bullet)  
 shoot\_sound.play()  
  
  
class Mob\_ship\_down(pygame.sprite.Sprite):  
 def \_\_init\_\_(self):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image\_orig = random.choice(ship3\_images)  
 self.image\_orig.set\_colorkey(BLACK)  
 self.image = pygame.transform.scale(self.image\_orig, (40, 40))  
  
 self.rect = self.image.get\_rect()  
 self.speedy = random.randrange(1, 4)  
 self.speedx = random.randrange(-3, 3)  
 self.radius = int(self.rect.width \* .85 / 2)  
 # pygame.draw.circle(self.image, RED, self.rect.center, self.radius)  
 self.speedy = random.randrange(-4, -1)  
 self.speedx = random.randrange(-3, 3)  
 self.rect.x = random.randrange(WIDTH - self.rect.width)  
 self.rect.y = random.randrange(HEIGHT + 100, HEIGHT + 150)  
 self.predx = random.randrange(WIDTH - self.rect.width)  
 self.predy = random.randrange(HEIGHT + 100, HEIGHT + 150)  
  
 self.rot = 0  
 self.rot\_speed = random.randrange(-8, 8)  
 self.last\_update = pygame.time.get\_ticks()  
  
 def rotate(self):  
 pass  
  
 def change\_speed(self):  
 self.speedy = -self.speedy  
 self.speedx = -self.speedx  
  
 def update(self):  
 self.rect.x += self.speedx  
 self.rect.y += self.speedy  
 if self.rect.top < 600:  
 self.speedy = -self.speedy  
 if self.rect.left < -25 or self.rect.right > WIDTH + 20:  
 self.speedx = -self.speedx  
 if self.rect.bottom > HEIGHT and self.speedy > 0:  
 self.speedy = -self.speedy  
 if abs(self.rect.x - self.predx) + abs(self.rect.y - self.predy) > 200:  
 self.shoot()  
 self.predx = self.rect.x  
 self.predy = self.rect.y  
  
 def shoot(self):  
 bullet = Bullet(self.rect.centerx, self.rect.top, 4)  
 all\_sprites.add(bullet)  
 bullets\_4.add(bullet)  
 shoot\_sound.play()  
  
  
class Bullet(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, x, y, z):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 if z == 1 or z == 3:  
 self.image = bullet\_img  
 elif z == 2:  
 self.image = bullet\_img\_1  
 else:  
 self.image = bullet\_img\_2  
 self.image.set\_colorkey(BLACK)  
 self.rect = self.image.get\_rect()  
 if z == 3 or z == 2:  
 self.rect.top = y  
 else:  
 self.rect.bottom = y  
 self.rect.centerx = x  
 if z == 1 or z == 4:  
 self.speedy = -10  
 else:  
 self.speedy = 10  
 self.image = pygame.transform.rotate(self.image, 180)  
 self.naprav = z  
  
 def update(self):  
 if self.naprav == 1 or self.naprav == 4:  
 self.rect.y += self.speedy  
 else:  
 self.rect.y += self.speedy  
 # убить, если он заходит за верхнюю часть экрана  
 if self.rect.bottom < 0 or self.rect.top > HEIGHT:  
 self.kill()  
  
  
class Pow(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, center):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.type = random.choice(['shield', 'gun'])  
 self.image = powerup\_images[self.type]  
 self.image.set\_colorkey(BLACK)  
 self.rect = self.image.get\_rect()  
 self.rect.center = center  
 self.speedy = 2  
  
 def update(self):  
 self.rect.y += self.speedy  
 # убить, если он сдвинется с нижней части экрана  
 if self.rect.top > HEIGHT:  
 self.kill()  
  
  
class Explosion(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, center, size):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.size = size  
 self.image = explosion\_anim[self.size][0]  
 self.rect = self.image.get\_rect()  
 self.rect.center = center  
 self.frame = 0  
 self.last\_update = pygame.time.get\_ticks()  
 self.frame\_rate = 50  
  
 def update(self):  
 now = pygame.time.get\_ticks()  
 if now - self.last\_update > self.frame\_rate:  
 self.last\_update = now  
 self.frame += 1  
 if self.frame == len(explosion\_anim[self.size]):  
 self.kill()  
 else:  
 center = self.rect.center  
 self.image = explosion\_anim[self.size][self.frame]  
 self.rect = self.image.get\_rect()  
 self.rect.center = center  
  
  
# Загрузка всей игровой графики  
background = pygame.image.load(path.join(img\_dir, "nebo\_7.jpg")).convert()  
background\_rect = background.get\_rect()  
player\_img = pygame.image.load(path.join(img\_dir, "playerShip1\_orange.png")).convert()  
player\_mini\_img = pygame.transform.scale(player\_img, (25, 19))  
player\_mini\_img.set\_colorkey(BLACK)  
bullet\_img = pygame.image.load(path.join(img\_dir\_2, "laserRed16.png")).convert()  
bullet\_img\_1 = pygame.image.load(path.join(img\_dir\_2, "laserBlue16.png")).convert()  
bullet\_img\_2 = pygame.image.load(path.join(img\_dir\_2, "laserGreen10.png")).convert()  
meteor\_images = []  
ufo\_images = []  
ship1\_images = []  
ship2\_images = []  
ship3\_images = []  
meteor\_list = ['meteorBrown\_big1.png', 'meteorBrown\_med1.png', 'meteorBrown\_med1.png',  
 'meteorBrown\_med3.png', 'meteorBrown\_small1.png', 'meteorBrown\_small2.png',  
 'meteorBrown\_tiny1.png']  
ufo\_list = ['ufoBlue.png', 'ufoGreen.png', 'ufoRed.png', 'ufoYellow.png']  
ship1\_list = ['playerShip2\_orange.png']  
ship2\_list = ['playerShip2\_blue.png', 'playerShip2\_green.png', 'playerShip2\_red.png', 'playerShip2\_orange.png']  
ship3\_list = ['playerShip3\_blue.png', 'playerShip3\_green.png', 'playerShip3\_red.png', 'playerShip3\_orange.png']  
for img in meteor\_list:  
 meteor\_images.append(pygame.image.load(path.join(img\_dir\_1, img)).convert())  
for img in ufo\_list:  
 ufo\_images.append(pygame.image.load(path.join(img\_dir, img)).convert())  
for img in ship1\_list:  
 ship1\_images.append(pygame.image.load(path.join(img\_dir, img)).convert())  
for img in ship2\_list:  
 ship2\_images.append(pygame.image.load(path.join(img\_dir, img)).convert())  
for img in ship3\_list:  
 ship3\_images.append(pygame.image.load(path.join(img\_dir, img)).convert())  
  
explosion\_anim = {}  
explosion\_anim['lg'] = []  
explosion\_anim['sm'] = []  
explosion\_anim['player'] = []  
for i in range(9):  
 filename = 'regularExplosion0{}.png'.format(i)  
 img = pygame.image.load(path.join(img\_dir\_3, filename)).convert()  
 img.set\_colorkey(BLACK)  
 img\_lg = pygame.transform.scale(img, (75, 75))  
 explosion\_anim['lg'].append(img\_lg)  
 img\_sm = pygame.transform.scale(img, (32, 32))  
 explosion\_anim['sm'].append(img\_sm)  
 filename = 'sonicExplosion0{}.png'.format(i)  
 img = pygame.image.load(path.join(img\_dir\_4, filename)).convert()  
 img.set\_colorkey(BLACK)  
 explosion\_anim['player'].append(img)  
powerup\_images = {}  
powerup\_images['shield'] = pygame.image.load(path.join(img\_dir\_5, 'shield\_gold.png')).convert()  
powerup\_images['gun'] = pygame.image.load(path.join(img\_dir\_5, 'bolt\_gold.png')).convert()  
  
# Загрузка мелодий игры  
# Загрузка мелодий игры  
shoot\_sound = pygame.mixer.Sound(path.join(snd\_dir, 'pew.wav'))  
shield\_sound = pygame.mixer.Sound(path.join(snd\_dir, 'db0809038a0e70a.mp3'))  
power\_sound = pygame.mixer.Sound(path.join(snd\_dir, 'Sound\_08420.mp3'))  
expl\_sounds = []  
for snd in ['expl3.wav', 'expl6.wav']:  
 expl\_sounds.append(pygame.mixer.Sound(path.join(snd\_dir, snd)))  
  
all\_sprites = pygame.sprite.Group()  
mobs\_ship\_high = pygame.sprite.Group()  
mobs\_ship\_down = pygame.sprite.Group()  
mobs\_ufo = pygame.sprite.Group()  
mobs\_meteors = pygame.sprite.Group()  
bullets\_1 = pygame.sprite.Group()  
bullets\_2 = pygame.sprite.Group()  
bullets\_3 = pygame.sprite.Group()  
bullets\_4 = pygame.sprite.Group()  
powerups = pygame.sprite.Group()  
player = Player()  
all\_sprites.add(player)  
  
score = 0  
  
# Цикл игры  
highscores\_1 = load\_1()  
highscores\_2 = load\_2()  
highscores\_3 = load\_3()  
  
table\_records = False  
chosing\_game = True  
game\_over = False  
running = True  
  
while running:  
 if chosing\_game:  
 show\_choose\_game\_screen()  
 chosing\_game = False  
  
 # Держим цикл на правильной скорости  
  
 clock.tick(FPS)  
  
 # Ввод процесса (события)  
 for event in pygame.event.get():  
 # проверка для закрытия окна  
 if event.type == pygame.QUIT or (event.type == pygame.KEYUP and event.key == pygame.K\_o):  
 running = False  
  
 # Обновление  
 all\_sprites.update()  
  
 # проверьте, не попала ли пуля в игрока  
 hits = pygame.sprite.spritecollide(player, bullets\_2, True, pygame.sprite.collide\_circle)  
 for hit in hits:  
 player.shield -= random.randrange(15, 25)  
 expl = Explosion(hit.rect.center, 'sm')  
 all\_sprites.add(expl)  
 if player.shield <= 0:  
 death\_explosion = Explosion(player.rect.center, 'player')  
 all\_sprites.add(death\_explosion)  
 player.hide()  
 player.lives -= 1  
 player.shield = 100  
 hits = pygame.sprite.spritecollide(player, bullets\_4, True, pygame.sprite.collide\_circle)  
 for hit in hits:  
 player.shield -= random.randrange(15, 25)  
 expl = Explosion(hit.rect.center, 'sm')  
 all\_sprites.add(expl)  
 if player.shield <= 0:  
 death\_explosion = Explosion(player.rect.center, 'player')  
 all\_sprites.add(death\_explosion)  
 player.hide()  
 player.lives -= 1  
 player.shield = 100  
  
 # проверьте, не попала ли пуля в пулю  
 hits = pygame.sprite.groupcollide(bullets\_1, bullets\_2, True, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
  
 hits = pygame.sprite.groupcollide(bullets\_1, bullets\_4, True, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
  
 hits = pygame.sprite.groupcollide(bullets\_2, bullets\_3, True, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
  
 hits = pygame.sprite.groupcollide(bullets\_2, bullets\_4, True, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
  
 hits = pygame.sprite.groupcollide(bullets\_3, bullets\_4, True, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
  
 # проверьте, не попала ли пуля в моб  
 hits = pygame.sprite.groupcollide(mobs\_meteors, bullets\_1, True, True)  
 for hit in hits:  
 score += random.randrange(30, 50)  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 if random.random() > 0.9:  
 pow = Pow(hit.rect.center)  
 all\_sprites.add(pow)  
 powerups.add(pow)  
 newmob\_meteor()  
  
 hits = pygame.sprite.groupcollide(mobs\_meteors, bullets\_3, True, True)  
 for hit in hits:  
 score += random.randrange(30, 50)  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 if random.random() > 0.9:  
 pow = Pow(hit.rect.center)  
 all\_sprites.add(pow)  
 powerups.add(pow)  
 newmob\_meteor()  
  
 hits = pygame.sprite.groupcollide(mobs\_ufo, bullets\_1, True, True)  
 for hit in hits:  
 score += random.randrange(30, 50) \* 3  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 if random.random() > 0.9:  
 pow = Pow(hit.rect.center)  
 all\_sprites.add(pow)  
 powerups.add(pow)  
 newmob\_ufo()  
  
 hits = pygame.sprite.groupcollide(mobs\_ufo, bullets\_3, True, True)  
 for hit in hits:  
 score += random.randrange(30, 50) \* 3  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 if random.random() > 0.9:  
 pow = Pow(hit.rect.center)  
 all\_sprites.add(pow)  
 powerups.add(pow)  
 newmob\_ufo()  
  
 hits = pygame.sprite.groupcollide(mobs\_ship\_high, bullets\_1, True, True)  
 for hit in hits:  
 score += random.randrange(30, 50) \* 3  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 if random.random() > 0.9:  
 pow = Pow(hit.rect.center)  
 all\_sprites.add(pow)  
 powerups.add(pow)  
 newmob\_ship\_high()  
  
 hits = pygame.sprite.groupcollide(mobs\_ship\_high, bullets\_3, True, True)  
 for hit in hits:  
 score += random.randrange(30, 50) \* 3  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 if random.random() > 0.9:  
 pow = Pow(hit.rect.center)  
 all\_sprites.add(pow)  
 powerups.add(pow)  
 newmob\_ship\_high()  
  
 hits = pygame.sprite.groupcollide(mobs\_ship\_down, bullets\_3, True, True)  
 for hit in hits:  
 score += random.randrange(30, 50) \* 3  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 newmob\_ship\_down()  
  
 hits = pygame.sprite.groupcollide(mobs\_ship\_down, bullets\_1, True, True)  
 for hit in hits:  
 score += random.randrange(30, 50) \* 3  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 newmob\_ship\_down()  
  
 hits = pygame.sprite.groupcollide(mobs\_ship\_high, bullets\_4, True, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 if random.random() > 0.9:  
 pow = Pow(hit.rect.center)  
 all\_sprites.add(pow)  
 powerups.add(pow)  
 newmob\_ship\_high()  
  
 hits = pygame.sprite.groupcollide(mobs\_ship\_down, bullets\_2, True, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
 newmob\_ship\_down()

hits = pygame.sprite.groupcollide(mobs\_ship\_down, bullets\_4, False, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
  
 hits = pygame.sprite.groupcollide(mobs\_ship\_high, bullets\_2, False, True)  
 for hit in hits:  
 random.choice(expl\_sounds).play()  
 expl = Explosion(hit.rect.center, 'lg')  
 all\_sprites.add(expl)  
  
 # Проверка, не ударил ли моб игрока  
 hits = pygame.sprite.spritecollide(player, mobs\_meteors, True, pygame.sprite.collide\_circle)  
 for hit in hits:  
 player.shield -= hit.radius \* 2  
 expl = Explosion(hit.rect.center, 'sm')  
 all\_sprites.add(expl)  
 newmob\_meteor()  
 if player.shield <= 0:  
 death\_explosion = Explosion(player.rect.center, 'player')  
 all\_sprites.add(death\_explosion)  
 player.hide()  
 player.lives -= 1  
 player.shield = 100  
  
 hits = pygame.sprite.spritecollide(player, mobs\_ufo, True, pygame.sprite.collide\_circle)  
 for hit in hits:  
 player.shield -= hit.radius \* 2  
 expl = Explosion(hit.rect.center, 'sm')  
 all\_sprites.add(expl)  
 newmob\_ufo()  
 if player.shield <= 0:  
 death\_explosion = Explosion(player.rect.center, 'player')  
 all\_sprites.add(death\_explosion)  
 player.hide()  
 player.lives -= 1  
 player.shield = 100  
  
 hits = pygame.sprite.spritecollide(player, mobs\_ship\_high, True, pygame.sprite.collide\_circle)  
 for hit in hits:  
 player.shield -= hit.radius \* 2  
 expl = Explosion(hit.rect.center, 'sm')  
 all\_sprites.add(expl)  
 newmob\_ship\_high()  
 if player.shield <= 0:  
 death\_explosion = Explosion(player.rect.center, 'player')  
 all\_sprites.add(death\_explosion)  
 player.hide()  
 player.lives -= 1  
 player.shield = 100  
  
 hits = pygame.sprite.spritecollide(player, mobs\_ship\_down, True, pygame.sprite.collide\_circle)  
 for hit in hits:  
 player.shield -= hit.radius \* 2  
 expl = Explosion(hit.rect.center, 'sm')  
 all\_sprites.add(expl)  
 newmob\_ship\_down()  
 if player.shield <= 0:  
 death\_explosion = Explosion(player.rect.center, 'player')  
 all\_sprites.add(death\_explosion)  
 player.hide()  
 player.lives -= 1  
 player.shield = 100  
  
 # Проверка столкновений игрока и улучшения  
 hits = pygame.sprite.spritecollide(player, powerups, True)  
 for hit in hits:  
 if hit.type == 'shield':  
 player.shield += random.randrange(10, 30)  
 if player.shield >= 100:  
 player.shield = 100  
 shield\_sound.play()  
 if hit.type == 'gun':  
 player.powerup()  
 power\_sound.play()  
  
 # Если игрок умер, игра окончена  
 if player.lives == 0 and not death\_explosion.alive():  
 show\_end\_screen(score)  
  
 all\_sprites = pygame.sprite.Group()  
 mobs\_ship\_high = pygame.sprite.Group()  
 mobs\_ship\_down = pygame.sprite.Group()  
 mobs\_ufo = pygame.sprite.Group()  
 mobs\_meteors = pygame.sprite.Group()  
 bullets\_1 = pygame.sprite.Group()  
 bullets\_2 = pygame.sprite.Group()  
 bullets\_3 = pygame.sprite.Group()  
 bullets\_4 = pygame.sprite.Group()  
 powerups = pygame.sprite.Group()  
 player = Player()  
 all\_sprites.add(player)  
  
 score = 0  
 show\_go\_screen()  
  
 # Рендеринг  
 screen.fill(BLACK)  
 screen.blit(background, background\_rect)  
 all\_sprites.draw(screen)  
 draw\_text(screen, str(score), 18, WIDTH / 2, 10)  
 draw\_shield\_bar(screen, 5, 5, player.shield)  
 draw\_lives(screen, WIDTH - 100, 5, player.lives,  
 player\_mini\_img)  
 # После отрисовки всего, переворачиваем экран  
 pygame.display.flip()  
  
pygame.quit()

1. Пояснительная записка

Название проекта – игра “Бродилка-стрелялка” на языке программирования Python с помощью библиотеки pygame.

Автор проекта – Ефимов Алексей БСБО-06-21

Описание идеи – идея состоит в том, чтобы сделать игру на питоне, похожую на старый космический шутер. В игре будет 2 начальных экрана и экран завершения уровня. Также будет 3 уровня и возможность их выбора. Первый уровень – стрельба по метеоритам. Второй уровень – одна группа отстреливающихся врагов. Третий уровень – две группы отстреливающихся врагов.

Описание реализации – В проекте кроме обязательной части кода для всех игр с Pegame (высота и ширина окна, FPS, clock, есть цвета, начало игры и её конец, цикл running (сама игра)) есть классы: Player – класс, описывающий игрока, в нем описывается его создание, перемещение и стрельба. Классы мобов – объектами этих классов являются метеориты и вражеские космические корабли, они также могут двигаться. Метеориты могут вращаться, а корабли – стрелять. Есть 4 класса мобов для каждого их типа: метеориты, нло, корабли сверху, корабли снизу. Также есть класс Explosion (описание анимации взрыва), класс Pow – в игре есть 2 суперспособности – увелечение огневой мощи и здоровья. Последний класс – класс Pow. Он описывает поведение пуль.

Также есть по 3 метода для загрузки и сохранения таблиц рекордов, так как есть 3 разные таблицы для каждого уровня. И методы создания экранов – загрузочных и экрана окончания игры.

Описание технологий – В проекте задействованы основные технологии языка Python – условия if и elif, списки, циклы и условия выхода из них, функции, классы, работа с файлами, а также множество составляющих библиотеки pygame и следующие библиотеки:







