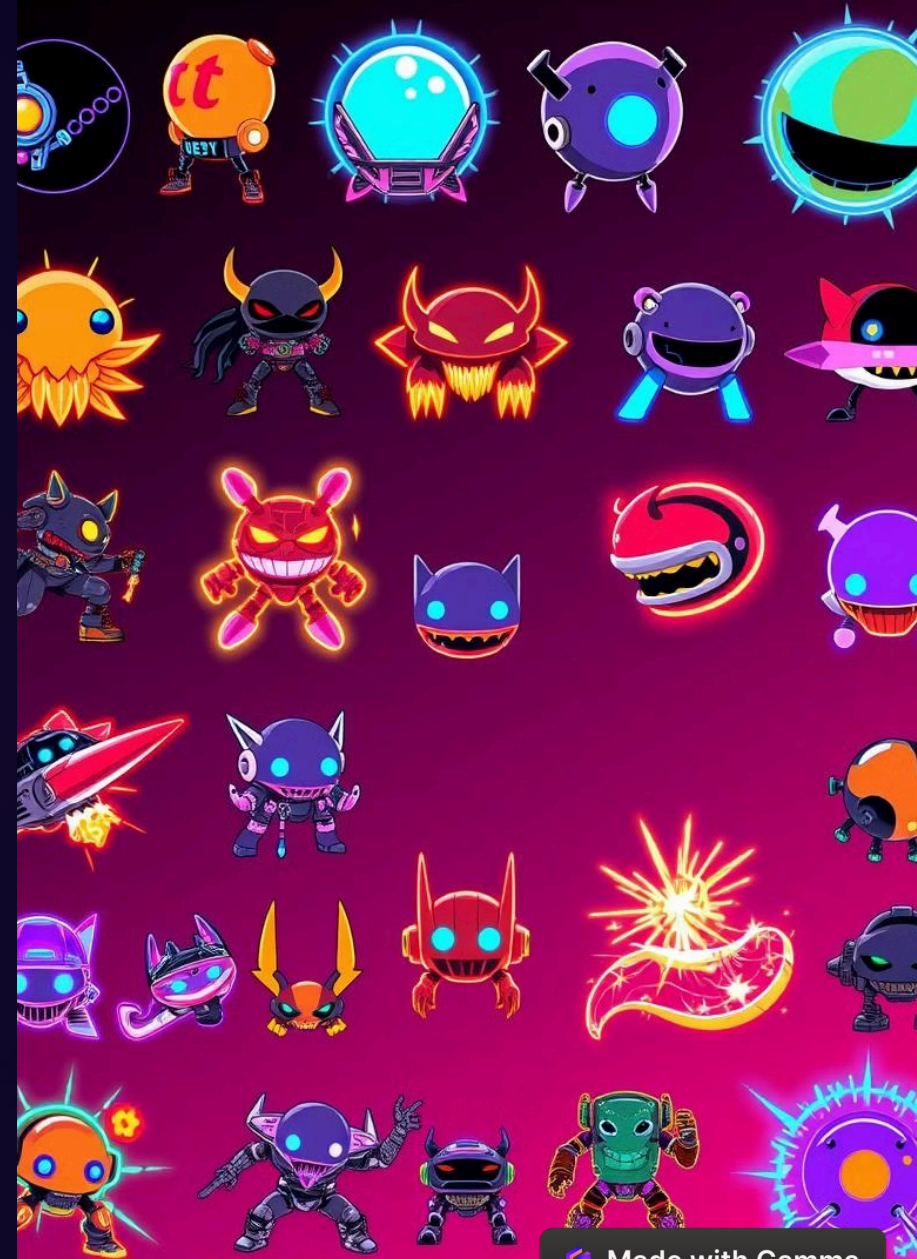


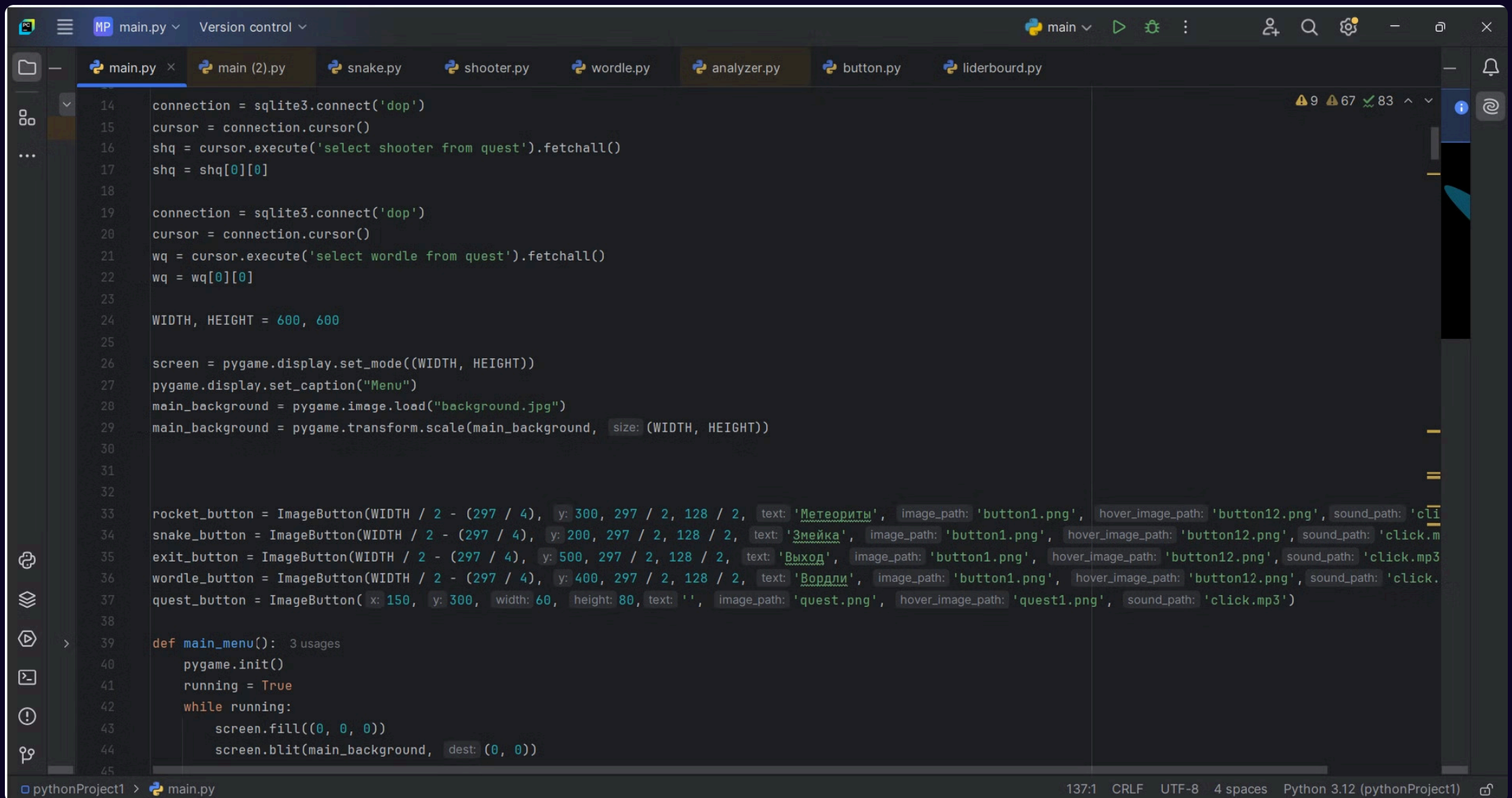
Сборник Игр на Python

Этот сборник игр на Python включает в себя несколько увлекательных игр, таких как "Змейка", "Метеориты" и "Ворддли". Каждая игра реализована как отдельная функция, вызываемая из главного меню. После завершения игры управление возвращается в главное меню, обеспечивая бесшовный игровой процесс.

by Руслан Жулимов



Главное Меню



```
14 connection = sqlite3.connect('dop')
15 cursor = connection.cursor()
16 shq = cursor.execute('select shooter from quest').fetchall()
17 shq = shq[0][0]
18
19 connection = sqlite3.connect('dop')
20 cursor = connection.cursor()
21 wq = cursor.execute('select wordle from quest').fetchall()
22 wq = wq[0][0]
23
24 WIDTH, HEIGHT = 600, 600
25
26 screen = pygame.display.set_mode((WIDTH, HEIGHT))
27 pygame.display.set_caption("Menu")
28 main_background = pygame.image.load("background.jpg")
29 main_background = pygame.transform.scale(main_background, size: (WIDTH, HEIGHT))
30
31
32
33 rocket_button = ImageButton(WIDTH / 2 - (297 / 4), y: 300, 297 / 2, 128 / 2, text: 'Метеориты', image_path: 'button1.png', hover_image_path: 'button12.png', sound_path: 'click.mp3')
34 snake_button = ImageButton(WIDTH / 2 - (297 / 4), y: 200, 297 / 2, 128 / 2, text: 'Змейка', image_path: 'button1.png', hover_image_path: 'button12.png', sound_path: 'click.mp3')
35 exit_button = ImageButton(WIDTH / 2 - (297 / 4), y: 500, 297 / 2, 128 / 2, text: 'Выход', image_path: 'button1.png', hover_image_path: 'button12.png', sound_path: 'click.mp3')
36 wordle_button = ImageButton(WIDTH / 2 - (297 / 4), y: 400, 297 / 2, 128 / 2, text: 'Вордли', image_path: 'button1.png', hover_image_path: 'button12.png', sound_path: 'click.mp3')
37 quest_button = ImageButton(x: 150, y: 300, width: 60, height: 80, text: '', image_path: 'quest.png', hover_image_path: 'quest1.png', sound_path: 'click.mp3')
38
39 def main_menu(): 3 usages
40     pygame.init()
41     running = True
42     while running:
43         screen.fill((0, 0, 0))
44         screen.blit(main_background, dest: (0, 0))
45
```

Графический интерфейс

Фоновое изображение и кнопки для выбора игр и выхода из программы. Отображение текста "СБОРНИК ИГР" в верхней части экрана.

Логика работы

Инициализация Pygame. Обработка событий: нажатие кнопок мыши для выбора игры или выхода. Использование pygame.USEREVENT для обработки нажатий на кнопки.



Игра "Змейка"

```
def snake_game(): 1 usage
import pygame
import time
import random

pygame.init()

width = 600
height = 600
block = 50

font_large = pygame.font.SysFont( name: 'Arial', size: 24)
font_medium = pygame.font.SysFont( name: 'Arial', size: 18)
font_small = pygame.font.SysFont( name: 'Arial', size: 12)

green = (0, 177, 45)
light_green = (34, 191, 17)
red = (199, 5, 0)
white = (255, 255, 255)
black = (0, 0, 0)
gold = (255, 215, 0)

window = pygame.display.set_mode((width, height))
pygame.display.set_caption('Snake')

font = pygame.font.SysFont( name: 'Arial', size: 50)

def message(text, color, x, y, font):
    mess = font.render(text, True, color)
    text_rect = mess.get_rect(center=(x, y))
    window.blit(mess, text_rect)
```

```
foodx = round(random.randrange( start: 0, width - block) / block) * block
foody = round(random.randrange( start: 0, height - block) / block) * block
len_of_snake += 1
score += 1
global sq
if sq < 100:
    sq += 1
    print(sq)
    a = cursor.execute("UPDATE quest SET snake = snake + 1")
    connection.commit()

draw_grid()
pygame.draw.rect(window, red, rect: [foodx, foody, block, block], width: 0)
draw_snake(snake_list)

pygame.draw.rect(window, black, rect: [0, 0, width, 40])
message( text: f"Score: {score}", white, width // 10, y: 20, font_small)

pygame.display.update()
clock.tick(10)

game_over_screen(score)

game()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Menu")
main_background = pygame.image.load("background.jpg")
```

```
def draw_grid():
    for row in range(0, width, block):
        for col in range(0, height, block):
            rect_color = green if (row // block + col // block) % 2 == 0 else light_green
            pygame.draw.rect(window, rect_color, rect: [row, col, block, block])

def draw_snake(snake_list):
    for coord in snake_list:
        pygame.draw.rect(window, color: (0, 24, 156), rect: [coord[0], coord[1], block, block], width: 0)

def game_over_screen(score):
    window.fill(black)
    message( text: "Sw ngame!", red, width // 2, height // 3, font_large)
    message( text: f"Score: {score}", gold, width // 2, height // 2, font_medium)
    message( text: "Богадат е меню...", white, width // 2, height // 1.5, font_small)
    pygame.display.flip()
    time.sleep(2)

def game():
    x1 = width // 2
    y1 = height // 2
    x1_change = 0
    y1_change = 0
    foodx = round(random.randrange( start: 0, width - block) / block) * block
    foody = round(random.randrange( start: 0, height - block) / block) * block
    clock = pygame.time.Clock()
```

Логика игры

Игрок управляет змейкой, которая движется по полю и собирает еду. При съедании еды длина змейки увеличивается, а счет растёт. Игра заканчивается, если змейка сталкивается со стеной или сама с собой.

Управление

Стрелки клавиатуры: управление направлением движения змейки.

Игра "Метеориты"

1

Логика игры

Игрок управляет ракетой, уклоняясь от метеоритов и стреляя по ним. За каждое попадание начисляются очки. Игра заканчивается, если ракета столкнется с метеоритом.

2

Классы

Player (ракета), Enemy (метеорит), Bullet (пуля). Каждый класс имеет атрибуты и методы для обновления позиции и выстрелов.

3

Управление

Стрелки клавиатуры: управление движением ракеты. Пробел: выстрел.

```
WIDTH = 600
HEIGHT = 400
SPEED = 5

window = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Shooter')
back = pygame.image.load('starfield.png').convert()
back_rect = back.get_rect()
player_img = pygame.image.load('rocket.png').convert()
meteor_img = pygame.image.load('meteor.png').convert()
lazer_img = pygame.image.load('laser.png').convert()
clock = pygame.time.Clock()

class Player(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.transform.scale(player_img, (50, 60))
        self.image.set_colorkey((0, 0, 0))
        self.rect = self.image.get_rect()
        self.rect.centerx = WIDTH / 2
        self.rect.bottom = HEIGHT - 5
        self.speedx = 0
        self.hp = 100

    def update(self):
        self.speedx = 0
        keystate = pygame.key.get_pressed()
        if keystate[pygame.K_LEFT]:
            self.speedx = -5
        if keystate[pygame.K_RIGHT]:
            self.speedx = 5
```

```
# проверка попадания пуль по врагам
hits = pygame.sprite.groupcollide(enemies, bullets, dokilla=True, dokillb=True)
for hit in hits:
    score += 10
    glo.score = int = 0
    if shq += 10
    print(shq)
    a = cursor.execute('UPDATE quest SET shooter = shooter + 10')
    connection.commit()
    enemy = Enemy()
    all_sprites.add(enemy)
    enemies.add(enemy)

# Рендер
window.blit(back, back_rect)
all_sprites.draw(window)
draw_text(window, text=f"Score: {score}", size=25, WIDTH // 2, y=10)
draw_text(window, text=f"HP: {player.hp}", size=25, WIDTH // 2, y=40)
pygame.display.flip()

# Экран поражения
if game_over:
    window.fill((0, 0, 0))
    draw_text(window, text="Вы проиграли!", size=50, WIDTH // 2, HEIGHT // 3, color=(255, 0, 0))
    draw_text(window, text="Возврат в меню...", size=30, WIDTH // 2, HEIGHT // 2, color=(255, 255, 255))
    pygame.display.flip()
    pygame.time.delay(2000) # Ожидание 3 секунды
```

Игра "Вордли"

1

Логика игры

Игрок должен угадать загаданное слово за ограниченное количество попыток. После каждой попытки буквы подсвечиваются: зеленый, желтый, серый.

2

Управление

Клавиатура: ввод букв. Backspace: удаление последней буквы. Enter: подтверждение догадки. Пробел: новая игра после окончания.

```
def wordle_game(): 1 usage
    import pygame
    import random
    from pymorphy2 import MorphAnalyzer
    pygame.init()
    WIDTH, HEIGHT = 800, 900
    WHITE = (255, 255, 255)
    BLACK = (0, 0, 0)
    GRAY = (200, 200, 200)
    GREEN = (106, 170, 100)
    YELLOW = (201, 180, 88)
    DARK_GRAY = (120, 124, 126)
    LIGHT_BLUE = (52, 152, 219)
    ROWS = 6
    COLS = 5
    LETTER_SIZE = 70
    SPACING = 10
    FONT_SIZE = 40
    KEYBOARD_HEIGHT = 200
    screen = pygame.display.set_mode((WIDTH, HEIGHT))
    pygame.display.set_caption("Wordle")
    try:
        font = pygame.font.Font(name="arial.ttf", FONT_SIZE)
        score_font = pygame.font.Font(name="arial.ttf", size=30)
        info_font = pygame.font.Font(name="arial.ttf", size=20)
    except:
        font = pygame.font.SysFont(name="Arial", FONT_SIZE)
        score_font = pygame.font.SysFont(name="Arial", size=30)
        info_font = pygame.font.SysFont(name="Arial", size=20)
```

```
try:
    with open("russian.txt", "r", encoding="windows-1251") as file:
        all_words = [word.strip().upper() for word in file if len(word.strip()) == 5]
except UnicodeDecodeError:
    print("Ошибка: Неверная кодировка файла. Убедитесь, что файл сохранен в кодировке Windows-1251.")
    exit()
morph = MorphAnalyzer()

def is_noun_in_nominative(word):
    parsed = morph.parse(word.lower())[0]
    return 'NOUN' in parsed.tag and 'nomn' in parsed.tag

russian_words = [word for word in all_words if is_noun_in_nominative(word)]
current_guess = []
guesses = [[] for _ in range(ROWS)]
current_row = 0
score = 0
game_over = False
correct_word = random.choice(russian_words).upper()
russian_letters = "ЙЦУКЕНГШХЪФЫВАПРОЛДЖЗЯЧСМИТЬБЮЁ"
keyboard_status = {letter: GRAY for letter in russian_letters}

def draw_grid():
    for row in range(ROWS):
        for col in range(COLS):
            x = col * (LETTER_SIZE + SPACING) + (WIDTH - COLS * (LETTER_SIZE + SPACING)) // 2
            y = row * (LETTER_SIZE + SPACING) + 100
            rect = pygame.Rect(x, y, LETTER_SIZE, LETTER_SIZE)
            color = GRAY
            if len(guesses[row]) > col:
```

Общие Замечания



Модульность

Каждая игра реализована как отдельная функция, что делает код более читаемым и удобным для поддержки. Использование классов в игре "Метеориты" позволяет эффективно управлять объектами.



Графика

Для отрисовки используются стандартные функции Pygame. Каждая игра имеет уникальный дизайн, соответствующий её механике.



Обработка ошибок

В игре "Вордли" проверяется кодировка файла со словами, чтобы избежать ошибок при чтении.





Возможные Улучшения

1

Больше игр

Добавить больше игр в сборник, чтобы расширить выбор для пользователя.

2

Сохранение рекордов

Реализовать сохранение рекордов, чтобы добавить соревновательный элемент.

3

Настройки

Добавить настройки, например, выбор уровня сложности, чтобы адаптировать игры под разные предпочтения.



Ключевые Выводы

Этот сборник игр демонстрирует возможности Python и Pygame для создания простых, но увлекательных игр. Модульная структура и использование классов облегчают поддержку и расширение кода. Возврат в главное меню обеспечивает удобный игровой процесс.