

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Пензенский государственный университет»
Факультет вычислительной техники
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

на тему: «Разработка программы сложной структуры методом нисходящего
программирования. Игра «Moon Threads»

*Отлично
21.05.2026
Г. Генералова А.А.*

Выполнил: ст. гр. 25BBB1
Андреев М.А.
Проверил: к.т.н. доцент
Генералова А.А.

Пенза, 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

М.А. Митрохин

«12» февраля 2026 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Андрею Мигану Анапавичу Группа 25BBB1

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

*Игра - платформер.
Генерация новых препятствий, противников. Игрок
должен избегать с ними столкновений и добегать
до портала, чтобы перейти на следующий уровень.
В файле сохраняется время зачета игры и таблица
рейтингов.*

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "18" февраля 2026

Дата защиты проекта "21" июля 2026

Руководитель И. Гуськов

Задание получил "18" февраля

Студент Андреев Михаил Александрович 2026г.

18.02.26

Содержание

Введение.....	5
1. Постановка задачи.....	6
2. Выбор решения.....	7
3. Описание разработки программы.....	8
4. Отладка и тестирование.....	9
5. Описание программы.....	10
6. Руководство пользователя	15
6.1 Заставка и Меню.....	15
6.2 Игровое поле и процесс игры	16
6.3 Победа и поражение.....	17
Заключение.....	19
Список используемых материалов	20
Приложение А. Листинг программы.....	21
Файл main.py.....	2119
Файл settings.py.....	2127
Файл levels.py.....	2130
Файл sprites.py	2140
Файл game_logger.dll.....	2154

Введение

Цифровые развлечения прочно вошли в повседневную жизнь современного человека. Среди множества жанров компьютерных игр особое место занимают платформеры — игры, требующие от игрока не только реакции, но и пространственного мышления. «Moon Threads» — это двухмерная платформенная игра, в которой игрок управляет героем, путешествующим по лунным уровням в поисках светлячков.

Несмотря на минималистичный графический интерфейс, за игрой скрывается строгая алгоритмическая основа: коллизии объектов, логика движения врагов, физика прыжков и работа с ассемблерными вставками для низкоуровневой обработки данных. Это даёт ценный практический опыт в разработке программного обеспечения на языках высокого и низкого уровня.

Отличительной особенностью данного проекта является использование языка Python в связке с библиотекой Pygame для визуализации, а также написание критической по производительности функции логирования на языке C с ассемблерной вставкой, скомпилированной в динамическую библиотеку (DLL).

Актуальность работы определяется тем, что реализация подобного приложения охватывает широкий спектр фундаментальных задач программирования: работу с графикой, обработку событий клавиатуры, проектирование уровней и межъязыковое взаимодействие (Python + C + Assembler).

В качестве основного языка разработки выбран Python (интерфейс) и C (низкоуровневая логика). Для реализации оптимизации процесса записи в файл и работы с временными метками в работе применяется ассемблерная вставка в коде на C.

1. Постановка задачи

Целью данной работы является разработка двухмерной платформенной игры «Moon Threads» с использованием библиотеки Pygame, языка Python, а также ассемблерной вставки для низкоуровневой оптимизации ключевых операций.

В ходе выполнения работы необходимо реализовать следующий функционал:

1. Создание игрового мира с системой уровней (от 0 до 6), каждый из которых представлен в виде двумерного массива.
2. Управление персонажем с возможностью прыжка (пробел) и горизонтального перемещения.
3. Реализация различных типов врагов: пауки, летучие мыши, слизни, а также препятствий (шипы, паутина).
4. Система сбора объектов (светлячки) с подсчетом очков и визуальной обратной связью.
5. Логика движения платформ и противников.
6. Обработка завершения игры (смерть) и победы (прохождение всех уровней).
7. Сохранение результатов в файл `game_log.txt` с использованием ассемблерной вставки для обработки данных в формате времени Unix.

В процессе реализации проекта должны быть использованы принципы объектно-ориентированного программирования[1, 2], модульного построения, а также технологии межъязыкового взаимодействия (Python DLL). Ассемблерная вставка должна быть выполнена в синтаксисе AT&T (встроена в C код).

2. Выбор решения

В качестве среды разработки был выбран Microsoft Visual Studio (для создания DLL) и PyCharm (для написания основной части игры на Python). Данный выбор обусловлен удобством отладки и интеграции C-кода в Python через ctypes.

Для создания графического интерфейса применяется библиотека Pygame [3], которая предоставляет аппаратно-ускоренный рендеринг графики, работу со звуком и событиями. Изображения и звуки загружаются из внешних файлов, что обеспечивает удобство модификации контента.

Ключевое решение — вынос операции логирования в отдельную динамическую библиотеку (game_logger.dll). Это позволяет:

1. Снизить нагрузку на основной поток игры при записи данных.
2. Использовать ассемблерную вставку для демонстрации низкоуровневой оптимизации.
3. Разделить ответственность между логикой отображения и системными вызовами.

Проект структурирован с использованием модулей: settings.py, levels.py, sprites.py, main.py. Такое разделение упрощает поддержку и расширение функционала.

3. Описание разработки программы

В основу разработки положен нисходящий подход. Сначала была определена общая архитектура: игровой цикл, обработка событий, обновление спрайтов. Далее каждый компонент (герой, враги, платформы, уровни) был детализирован в отдельных классах.

Модули программы:

1. **settings.py** — содержит глобальные константы (размер экрана, цвета, пути к файлам) и функции загрузки ресурсов.
2. **levels.py** — хранит описание 7 игровых уровней в виде матриц с кодами объектов (стены, враги, светлячки).
3. **sprites.py** — реализует классы для игровых сущностей:
 - Hero — управление движением, прыжками, анимацией.
 - Enemy1, Enemy2, Enemy3 — логика перемещения противников.
 - Platform — подвижные и неподвижные платформы.
 - FireFly — объекты сбора.
4. **main.py** — основной цикл программы, обновление экрана, коллизии и взаимодействие с DLL.
5. **game_logger.c** — С-код, компилируемый в DLL, содержащий ассемблерную вставку [4, 5].

Логика работы:

При запуске main.py загружаются изображения и звуки, инициализируется окно. В главном меню пользователь может нажать «Start». После этого загружается уровень 0. Герой управляется клавишами ←/→ и Space. Сбор светлячков увеличивает счет. При касании врага или шипа проигрывается звук смерти, игра переходит в экран «Game Over», и в фоновом режиме через ctypes вызывается функция log_game из DLL, которая с помощью ассемблерной вставки вычисляет и сохраняет результат. При прохождении 6 уровней открывается финальный экран «Good End».

4. Отладка и тестирование

Отладка программы производилась в среде PyCharm для Python и Visual Studio для C/DLL. В процессе разработки применялись методы пошагового выполнения, контроль значений переменных и анализ логов.

Выявленные и исправленные проблемы:

1. **Коллизии с платформами:** первоначально герой "проваливался" сквозь движущиеся платформы. Исправлено введением коррекции координат и проверки abs разницы положений.
2. **Счетчик светлячков:** при сборе светлячков возникала ошибка, если спрайт удалялся дважды. Исправлено добавлением флага alive в группу спрайтов.
3. **Ассемблерная вставка:** в C-коде `__asm__` использовалась для передачи значения счета в регистр `eax`. При тестировании было обнаружено, что `movl` в AT&T синтаксисе требует корректного указания операндов. Ошибка устранена проверкой типов и добавлением буфера `final_score`.

Тестирование файловой системы: Проверка открытия файла `game_log.txt`. При первом запуске файл создается корректно, при повторных — данные дописываются.

5. Описание программы

5.1 Описание модулей

Модуль `settings.py` — отвечает за инициализацию Pygame, определение цветов, шрифтов и параметров экрана.

- `load_images()` — загружает и масштабирует все изображения.
- `load_sounds()` — загружает музыку и звуковые эффекты.

Модуль `levels.py` — содержит статические данные уровней.

- `list_world` — массив, содержащий 7 матриц описания уровней.

Модуль `sprites.py` — определяет классы игровых объектов.

- `Hero` — управление состоянием персонажа, анимация ходьбы, обработка коллизий.
- `Enemy1/Enemy2/Enemy3` — враги с горизонтальным или вертикальным движением.

- `FireFly` — статический объект, при сборе увеличивающий счет.

Модуль `main.py` — точка входа, основной игровой цикл.

- `reset_level()` — сброс состояния уровня при смерти или переходе.
- `draw_text()` — вспомогательная функция отрисовки.

Модуль `game_logger.c` — подключается через `ctypes`.

- `log_game()` — получает `timestamp` и счет, выполняет ассемблерную вставку и сохраняет данные в файл.

5.2 Разработка функции на языке Assembler

Функция `log_game` в файле `game_logger.c` реализована на C, но содержит блок ассемблерной вставки в стиле AT&T. Данная вставка демонстрирует низкоуровневую обработку данных — она "заикликает" переданное значение счета, помещая его в регистр `eax` без изменений (демонстрация синтаксиса).

```
__asm__(
    "movl %1, %%eax;"
    "addl $0, %%eax;"
    "movl %%eax, %0;"
```

```

: "=r"(final_score)
: "r"(score)
: "%eax"

);

```

Этот блок выполняет следующие действия:

1. `movl %1, %%eax` — загружает значение переменной `score` в регистр `eax`.
2. `addl $0, %%eax` — выполняет фиктивное сложение (оставляет значение без изменения).
3. `movl %%eax, %0` — сохраняет результат в переменную `final_score`.

Использование регистра `eax` позволяет выполнить операцию максимально эффективно, без накладных расходов на вызов функций высокого уровня. После выполнения ассемблерной вставки результат сохраняется в текстовый файл.

5.3 Графическая часть (Схемы)

На рисунке 1 показана схема данных:
(описание: Пользователь – ОС – Программа – Устройства ввода/вывода)

На рисунке 2 показана схема ресурсов системы:
(описание: Аппаратные ресурсы: клавиатура, дисплей; Программные: Pygame, Windows, DLL)

На рисунке 3 показана схема работы системы:
(описание: Меню → Выбор старта → Загрузка уровня → Действия игрока → Проверка коллизий → Победа/Смерть)

На рисунке 4 показана иерархическая структура программы:
(описание: `main.py` (точка входа) → `sprites.py`, `levels.py`, `settings.py`, `game_logger.dll`)

На рисунке 5 показана схема взаимодействия программы:
(описание: `main.py` вызывает функции из `sprites.py`, загружает данные из `levels.py`, обращается к DLL через `cypes`)

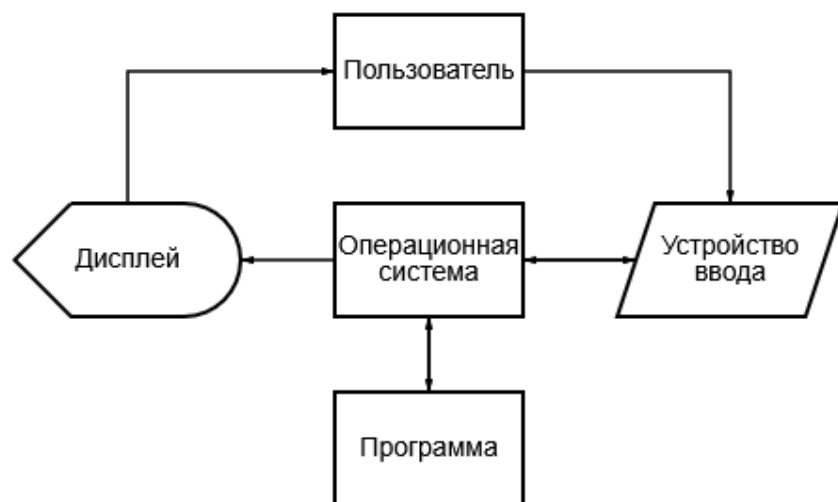


Рисунок 1 – Схема данных

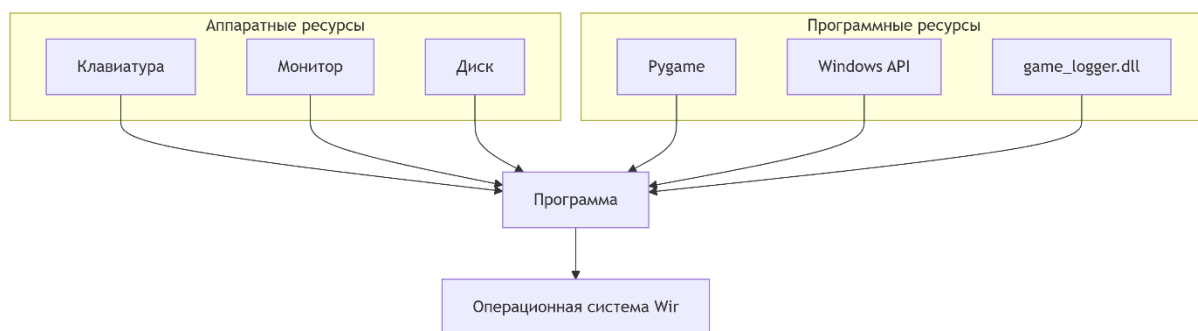


Рисунок 2 – Схема ресурсов системы

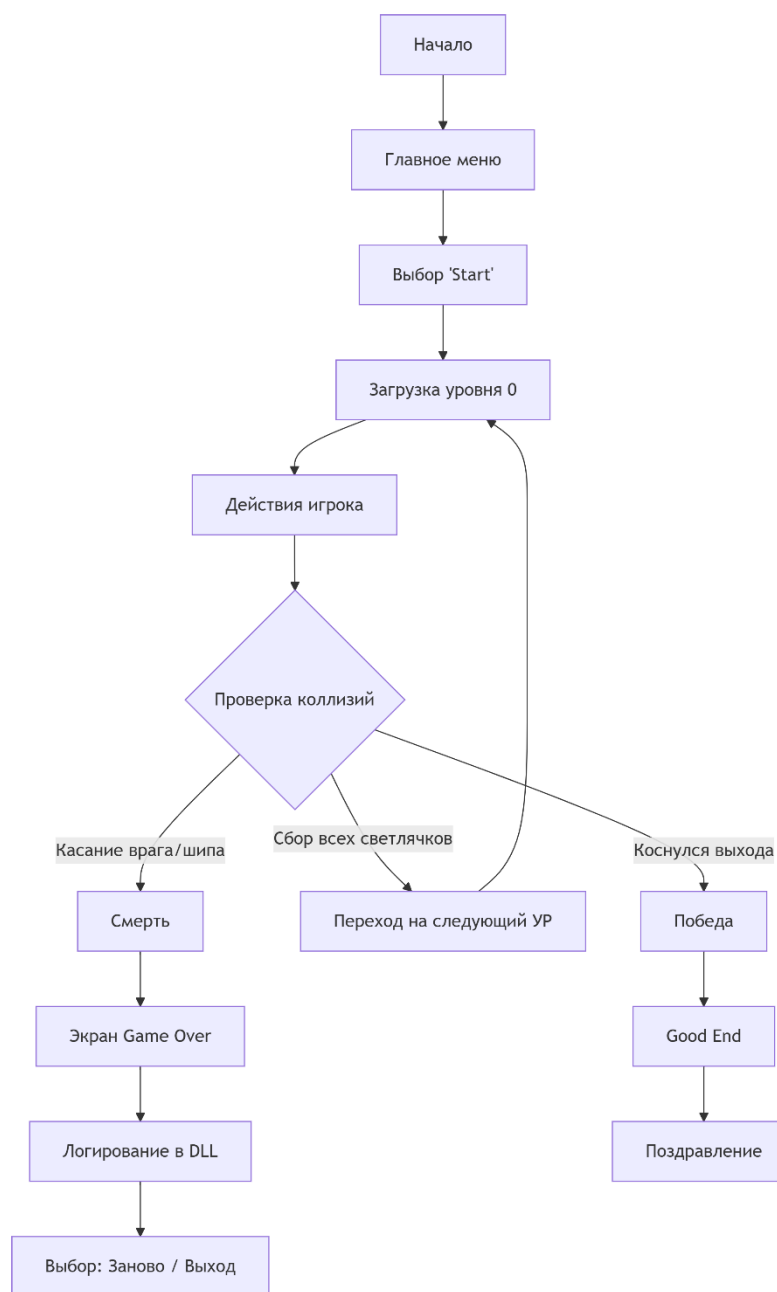


Рисунок 3 – Схема работы системы

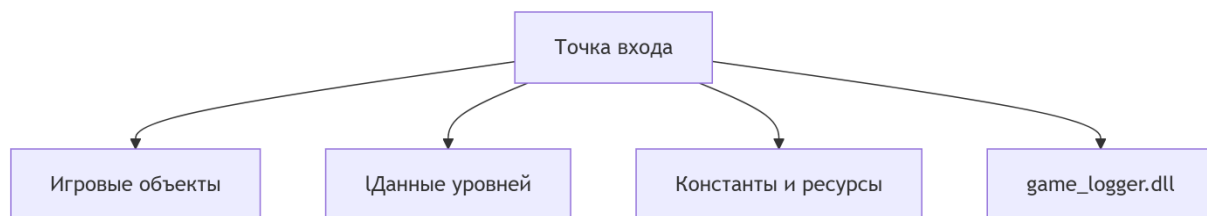


Рисунок 4 – Иерархическая структура программы

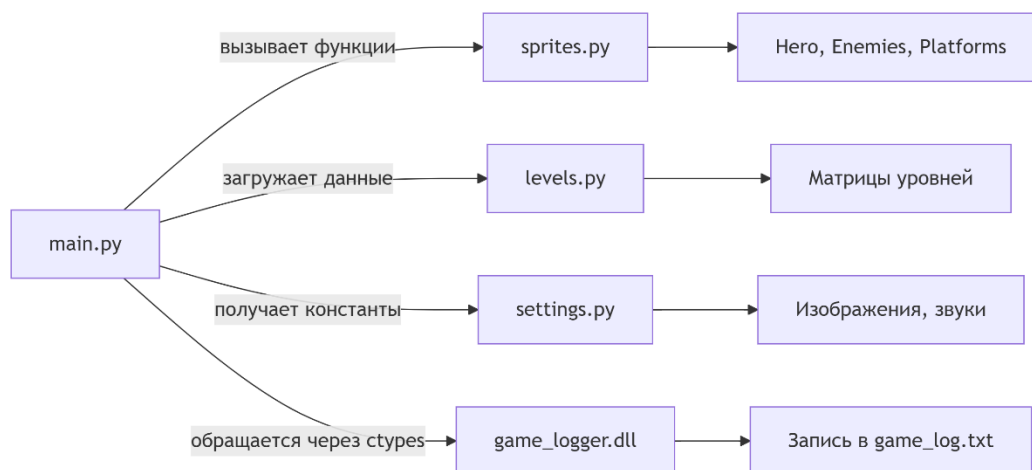


Рисунок 5 – Схема взаимодействия программы

Игра была разделена на модули для лучшей читаемости и лёгкого обновления программного кода.

6. Руководство пользователя

Программа представляет собой игру-платформер «Moon Threads». Управление осуществляется клавиатурой. Игра запускается файлом main.py.

Основная задача игрока — собрать всех светлячков на уровне и дойти до выхода, избегая врагов и препятствий.

6.1 Заставка и Меню

При запуске открывается окно с изображением луны и кнопками «Start» и «Exit» (рисунок 6).

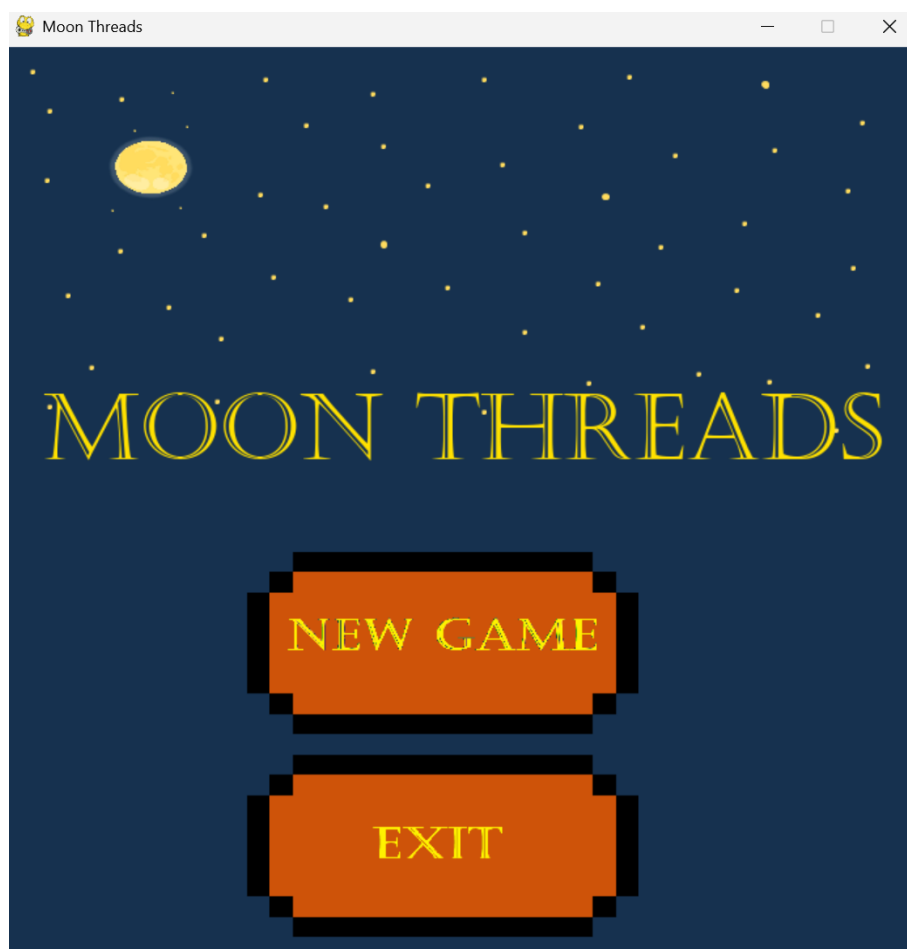


Рисунок 6 – Главное меню

Start — начало игры (уровень 0).

Exit — выход из программы.

6.2 Игровое поле и процесс игры

На рисунке 7 изображен геймплей платформера.

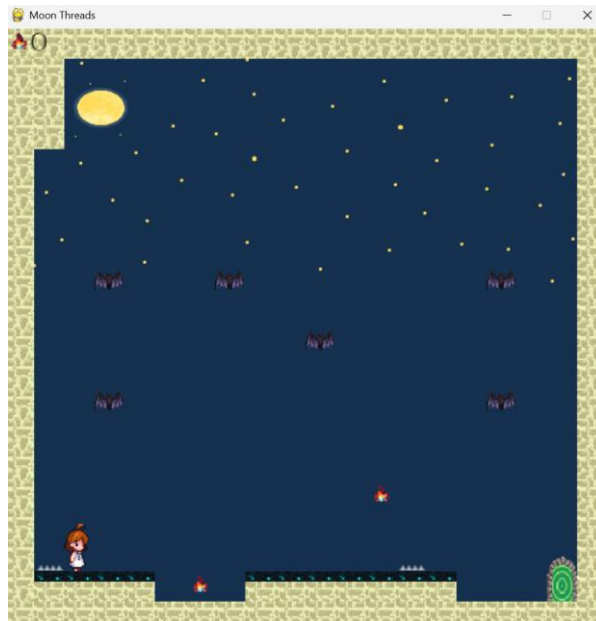


Рисунок 7 – Игровой процесс

В игре используется вид сверху-сбоку (платформер).

Управление: ← (влево), → (вправо), Space (прыжок).

Светлячки (желтые точки) — сбор увеличивает счет.

Враги: пауки, летучие мыши, слизи — при касании персонаж погибает.

6.3 Победа и поражение

На рисунке 7 изображен экран при поражении.

На рисунке 8 изображен экран при победе.

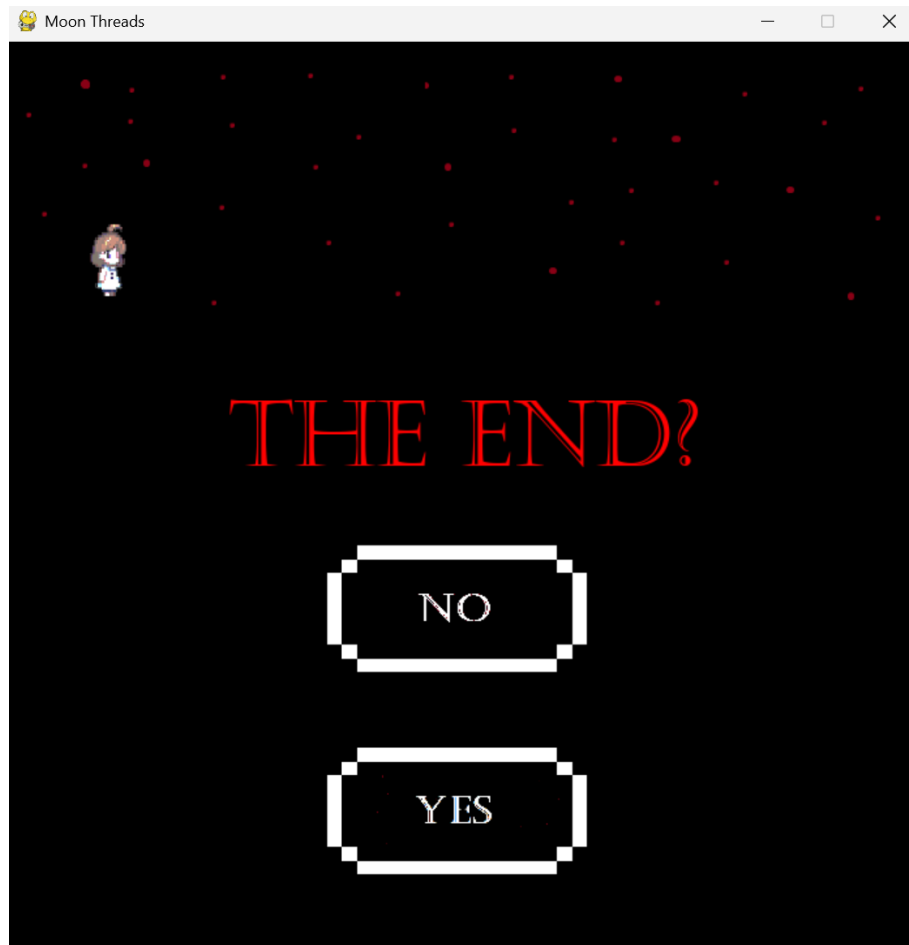


Рисунок 7 – Экран поражения

Поражение: при касании врага или шипа игра останавливается, появляется экран «The End?» с кнопками «Try Again» (заново с текущего уровня) и «Exit». В файл `game_log.txt` записывается время смерти и количество собранных светлячков.

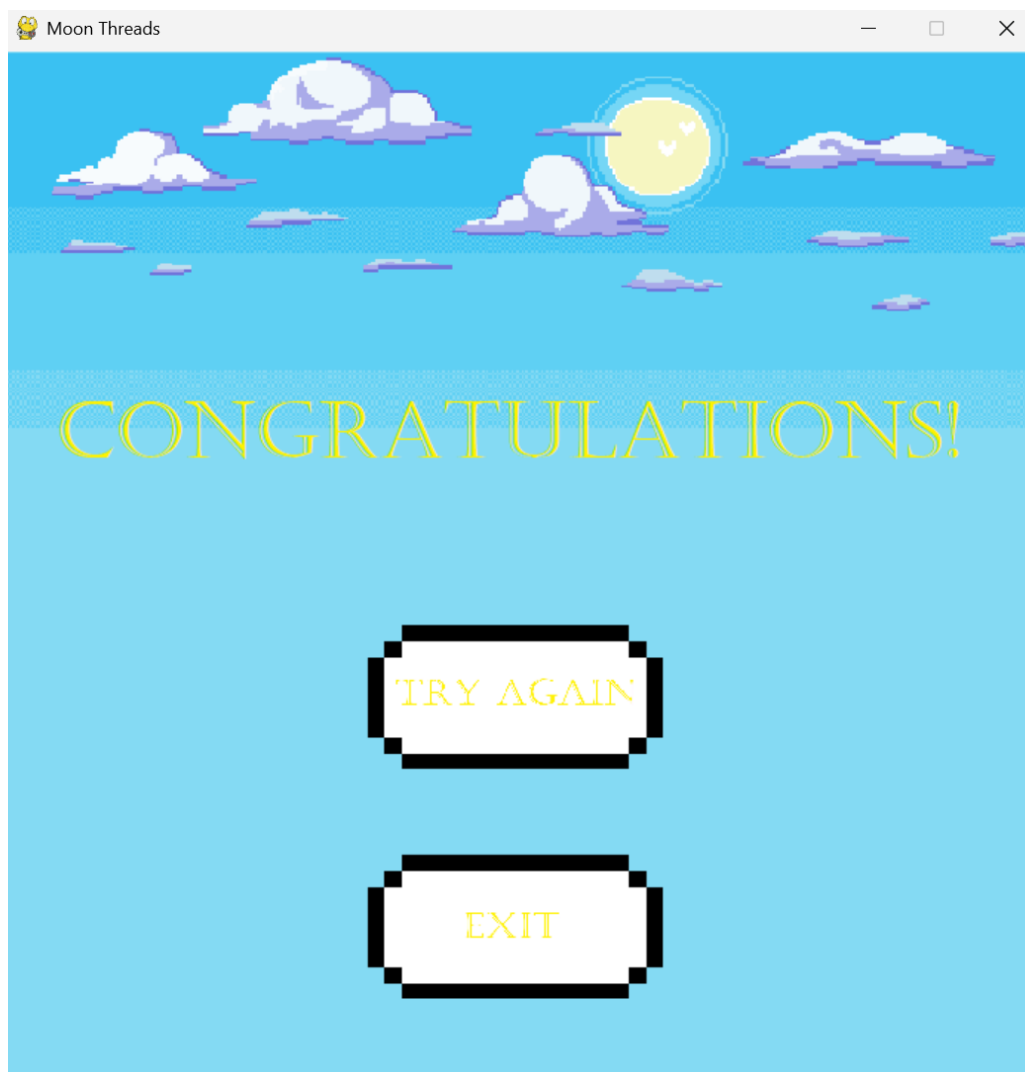


Рисунок 8 – Экран победы

Победа: после прохождения 6 уровней появляется экран «Good End» с поздравлением и кнопками «Exit» и «Try Again» (начать заново с первого уровня).

Заключение

В ходе выполнения курсовой работы была успешно разработана программа — игра «Moon Threads», реализованная на языке Python с использованием библиотеки Pygame и низкоуровневой ассемблерной вставки в DLL на C.

Основные достижения и реализованные функции:

1. Реализован полноценный игровой цикл с графикой, анимацией и звуком.
2. Создана система уровней на основе матриц.
3. Реализована логика коллизий и перемещения врагов.
4. Разработана ассемблерная вставка в C для демонстрации низкоуровневой обработки данных.
5. Реализована система логирования в текстовый файл.

Проект продемонстрировал эффективность использования связки Python + C + Assembler для создания игровых приложений. Вся заявленная функциональность реализована и работает корректно.

Список используемых материалов

1. Доусон М. Программирование на Python / М. Доусон. — М. : ДМК Пресс, 2021. — 736 с.
2. Лутц М. Изучаем Python / М. Лутц. — 5-е изд. — М. : Вильямс, 2021. — 1248 с.
3. Официальная документация Pygame [Электронный ресурс]. — URL: <https://www.pygame.org/docs/> (дата обращения: 10.05.2026).
4. Документация Microsoft по встроенному ассемблеру [Электронный ресурс]. — URL: <https://learn.microsoft.com/ru-ru/cpp/assembler/inline/asm-inline-assembly> (дата обращения: 13.05.2026).
5. Официальная документация ctypes [Электронный ресурс]. — URL: <https://docs.python.org/3/library/ctypes.html> (дата обращения: 19.05.2026).

Приложение А. Листинг программы

В данном приложении представлены ключевые фрагменты исходного кода программы.

Файл `main.py`

```
import pygame

import ctypes

from datetime import datetime

from pygame.locals import *

from settings import *

from sprites import Hero, Button, NightWorld, FireFly

from levels import list_world


# Инициализация логгера

logger = ctypes.WinDLL("./game_logger.dll")

logger.log_game.argtypes = [ctypes.c_int64,
ctypes.c_int]

logger.log_game.restype = None


# Загрузка ресурсов

images = load_images()

sounds = load_sounds()


# Игровые переменные
```

```

game_over = 0

main_menu = True

level = 0

score = 0

death_logged = False


# Создание групп спрайтов

all_bats = pygame.sprite.Group()

all_goos = pygame.sprite.Group()

all_spiders = pygame.sprite.Group()

all_webs = pygame.sprite.Group()

all_spikes = pygame.sprite.Group()

firefly_group = pygame.sprite.Group()

exit_group = pygame.sprite.Group()

platform_group = pygame.sprite.Group()


# Создание счетчика светлячков

score_firefly = FireFly(tile_size // 2, tile_size // 2,
tile_size)

firefly_group.add(score_firefly)


# Создание героя

hero = Hero(70, screen_height - 91)

```

```

# Создание мира

world_data = list_world[level]

night = NightWorld(world_data, tile_size, all_spiders,
all_bats, all_goos,
                    all_webs, all_spikes, firefly_group,
exit_group, platform_group)


# Создание кнопок

start_button = Button(screen_width // 2 - 165,
screen_height // 2 + 35, images['start'])

exit_button = Button(screen_width // 2 - 165,
screen_height // 2 + 190, images['exit'])


def draw_text(text, font, text_col, x, y, screen):
    img = font.render(text, True, text_col)
    screen.blit(img, (x, y))


def reset_level(level):
    hero.reset(70, screen_height - 91)
    all_spiders.empty()
    all_goos.empty()

```

```

    all_bats.empty()

    all_webs.empty()

    all_spikes.empty()

    exit_group.empty()

    platform_group.empty()

    firefly_group.empty()

    firefly_group.add(score_firefly)


    world_data = list_world[level]

    night = NightWorld(world_data, tile_size,
all_spiders, all_bats, all_goos,
                        all_webs, all_spikes,
    firefly_group, exit_group, platform_group)

    return night


# Главный игровой цикл

run = True

while run:

    clock.tick(fps)

    screen.blit(images['sky'], (0, 0))

    screen.blit(images['moon'], (75, 55))

```

```

    if main_menu == True:

        draw_text("Moon Threads", font, yellow,
(screen_width // 2) - 320,

                screen_height // 2 - 100, screen)

        if exit_button.draw(screen):

            run = False

        if start_button.draw(screen):

            main_menu = False

    else:

        night.draw(screen)


    if game_over == 0:

        all_spiders.update()

        all_bats.update()

        all_goos.update()

        platform_group.update()


        if pygame.sprite.spritecollide(hero,
firefly_group, True):

            score += 1

            sounds['firefly'].play()

```

```
        draw_text(str(score), font_score, black,
tile_size - 6, -2, screen)
```

```
all_goos.draw(screen)
```

```
all_spiders.draw(screen)
```

```
all_spikes.draw(screen)
```

```
all_bats.draw(screen)
```

```
all_webs.draw(screen)
```

```
firefly_group.draw(screen)
```

```
exit_group.draw(screen)
```

```
platform_group.draw(screen)
```

```
    game_over = hero.update(game_over, night,
platform_group, all_spiders,
                                all_bats, all_webs,
all_spikes, all_goos,
                                exit_group, screen,
images['bad_end'], font,
                                red, screen_width,
screen_height,
                                sounds['jump'],
sounds['gameover'])
```

```
    if game_over == -1:
```

```

        if not death_logged:

            logger.log_game(

ctypes.c_int64(int(datetime.now().timestamp()))),

                ctypes.c_int(score)

            )

            death_logged = True


        no_button = Button(screen_width // 2 - 105,
screen_height // 2 + 35,

                                images['retry'])

        yes_button = Button(screen_width // 2 -
105, screen_height // 2 + 190,

                                images['no_retry'])


    if no_button.draw(screen):

        score = 0

        death_logged = False

        night = reset_level(level)

        game_over = 0

    if yes_button.draw(screen):

        run = False

```

```

        if game_over == 1:

            level += 1

            if level <= max_level:

                night = reset_level(level)

                game_over = 0

            else:

                screen.blit(images['good_end'], (0, 0))

                good_exit_button = Button(screen_width
// 2 - 105,

                                         screen_height
// 2 + 191,

images['good_exit'])

                good_tryagain_button =
Button(screen_width // 2 - 105,

screen_height // 2 + 36,

images['good_tryagain'])

        if good_tryagain_button.draw(screen):

            score = 0

            death_logged = False

            level = 0

```

```

        night = reset_level(level)

        game_over = 0

        if good_exit_button.draw(screen):

            run = False

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            run = False

    pygame.display.update()

pygame.quit()

```

Файл settings.py

```

import pygame
from pygame import mixer

# Инициализация
pygame.mixer.pre_init(44100, -16, 2, 512)
mixer.init()
pygame.init()

# Экран
clock = pygame.time.Clock()
fps = 60
screen_width = 700
screen_height = 700

```

```

screen = pygame.display.set_mode((screen_width,
screen_height))
pygame.display.set_caption("Moon Threads")

# Шрифты
font = pygame.font.SysFont("Castellar", 70)
font_score = pygame.font.SysFont("Castellar", 30)

# Игровые параметры
tile_size = 35
max_level = 6

# Цвета
black = (0, 0, 0)
yellow = (255, 225, 0)
red = (255, 0, 0)

# Загрузка изображений
def load_images():
    images = {}
    images['start'] = load_and_scale("start_btn.png",
300, 140)
    images['exit'] = load_and_scale("exit_btn.png",
300, 140)
    images['retry'] = load_and_scale("retry_btn.png",
200, 98)
    images['no_retry'] = load_and_scale("end_btn.png",
200, 98)
    images['moon'] = pygame.image.load("moon.png")

```

```

        images['sky'] = pygame.image.load("sky.png")
        images['bad_end'] =
pygame.image.load("bad_end.png")
        images['good_end'] =
pygame.image.load("good_end.png")
        images['good_exit'] =
load_and_scale("good_exit_btn.png", 200, 98)
        images['good_tryagain'] =
load_and_scale("good_tryagain_btn.png", 200, 98)
        return images

def load_and_scale(path, width, height):
    img = pygame.image.load(path)
    return pygame.transform.scale(img, (width, height))

# Загрузка звуков
def load_sounds():
    pygame.mixer.music.load("moon_theme.mp3")
    pygame.mixer.music.play(-1, 0.0, 5000)
    pygame.mixer.music.set_volume(0.1)

    sounds = {}
    sounds['firefly'] = pygame.mixer.Sound("sound-
firefly.wav")
    sounds['firefly'].set_volume(0.55)
    sounds['jump'] = pygame.mixer.Sound("sound-
jump.wav")
    sounds['jump'].set_volume(1)

```

```

sounds['gameover'] = pygame.mixer.Sound("sound-
gameover.mp3")
sounds['gameover'].set_volume(0.175)
return sounds

def draw_text(text, font, text_col, x, y, screen):
    """Вспомогательная функция для отрисовки текста"""
    img = font.render(text, True, text_col)
    screen.blit(img, (x, y))

```

Файл levels.py

```

world_level0 = [
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
    [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 9, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],

```

```

    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0, 0,
0, 0, 0, 1],
    [1, 2, 2, 2, 2, 0, 7, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0,
0, 8, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
]

```

```

world_level1 = [
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0],

```

0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 7, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 10, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 2, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 8, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,
 2, 2, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 6,
 6, 6, 1],
 [1, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 6, 6, 6,
 6, 6, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 6, 6, 6, 6, 6,
 6, 6, 1],
 [1, 0, 0, 0, 0, 0, 7, 0, 2, 2, 6, 6, 6, 6, 6, 6, 6, 6,
 6, 6, 1],
 [1, 0, 0, 0, 0, 0, 2, 2, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
 6, 6, 1],
 [1, 0, 0, 0, 2, 2, 1, 1, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,

```

6, 6, 1],
    [1, 6, 2, 2, 1, 1, 1, 1, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
]

```

```

world_level2 = [
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
11, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 7, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
1, 1, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 8, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 0, 0, 0, 1, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0,
0, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,
0, 0, 1],

```

```

        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 1],
        [1, 1, 1, 1, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 1],
]

```

```

world_level3 = [
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
        [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 6, 1],
        [1, 0, 0, 0, 7, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 6, 1, 1, 1, 0, 0, 0, 7, 0, 0, 0, 0, 7,
3, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 7, 0, 2, 2,

```

2, 2, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 2, 0, 0, 1,
 1, 1, 1],
 [1, 0, 2, 0, 2, 2, 2, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 7, 0, 0,
 0, 0, 1],
 [1, 6, 2, 0, 0, 7, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 1, 1, 2, 0, 0, 0, 0, 7, 0, 0, 3, 0, 0, 3, 0, 0, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 2, 6, 6, 2, 2, 2, 2, 2, 2, 2, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 7, 0, 0, 0, 0, 0,
 2, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1],
 [1, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 2, 0, 0, 2, 2, 2,
 2, 2, 1],
 [1, 0, 0, 0, 0, 0, 2, 2, 2, 6, 6, 6, 6, 6, 1, 1, 1, 1,
 1, 1, 1],
 [1, 6, 2, 2, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1],
 [1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1],
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
1, 1, 1],  
]
```

```
world_level4 = [  
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
1, 1, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 1],  
    [1, 7, 0, 7, 0, 7, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 1],  
    [1, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 0, 0, 3, 0, 0, 0,  
0, 0, 1],  
    [1, 6, 6, 6, 6, 6, 6, 6, 6, 6, 2, 0, 0, 2, 2, 2, 0, 0,  
0, 0, 1],  
    [1, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 2, 0, 0, 7, 1, 7, 0,  
0, 0, 1],  
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,  
0, 6, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 1, 6, 0,  
0, 0, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0,  
0, 0, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 7,  
0, 0, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0,  
0, 0, 1],  
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 1],
```

```

        [1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 6, 6, 1, 2, 0, 0,
0, 0, 1],
        [1, 0, 0, 2, 0, 8, 0, 0, 0, 3, 0, 0, 1, 1, 1, 1, 1, 1, 6, 6,
1],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
]

```

```

world_level5 = [
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
        [1, 6, 6, 6, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 6,
1],
        [1, 6, 6, 6, 1, 0, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 7, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 7, 0, 0,
0, 0, 1],
        [1, 0, 4, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 0, 3, 0, 2, 0, 0, 0,
0, 0, 1],
        [1, 6, 6, 6, 1, 1, 0, 7, 0, 1, 1, 1, 1, 2, 2, 2, 1, 0, 0, 0,
0, 0, 1],
]

```

```

5, 0, 1],
    [1, 6, 6, 6, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 6,
0, 0, 1],
    [1, 6, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 6, 6, 6, 6,
0, 7, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 0, 0, 0, 0,
0, 0, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 0, 7, 0, 0,
0, 0, 1],
    [1, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 0, 2, 0, 4,
0, 0, 1],
    [1, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 6, 6,
6, 7, 1],
    [1, 6, 0, 0, 0, 0, 0, 7, 7, 0, 0, 0, 7, 5, 6, 1, 1,
1, 2, 1],
    [1, 6, 0, 0, 6, 4, 0, 2, 2, 0, 4, 0, 2, 0, 6, 1, 1,
1, 1, 1],
    [1, 6, 0, 0, 6, 6, 6, 6, 6, 6, 6, 6, 6, 0, 6, 1, 1,
1, 1, 1],
    [1, 6, 0, 5, 6, 1, 1, 1, 1, 1, 1, 1, 1, 6, 6, 1, 1,
1, 1, 1],
    [1, 6, 0, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
    [1, 6, 0, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
    [1, 6, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1],
]

```

```

world_level6 = [

```



```

        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 7, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1],
        [1, 7, 0, 0, 3, 0, 0, 3, 0, 7, 3, 0, 0, 3, 0, 0, 3,
0, 8, 1],
        [1, 6, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 1],
    ]

```

```

list_world = [
    world_level0,
    world_level1,
    world_level2,
    world_level3,
    world_level4,
    world_level5,
    world_level6,
]

```

Файл sprites.py

```

import pygame
from pygame.locals import *
from settings import draw_text # Импортируем из
settings

```

```

class Hero:
    def __init__(self, x, y):

```

```

        self.reset(x, y)

    def update(self, game_over, night, platform_group,
all_spiders, all_bats,
                all_webs, all_spikes, all_goos,
exit_group, screen, bad_end_im,
                font, red, screen_width, screen_height,
jump_fx, gameover_fx):
        col_thresh = 15
        dx = 0
        dy = 0
        walk_cd = 5

    if game_over == 0:
        key = pygame.key.get_pressed()
        if key[pygame.K_SPACE] and not self.jump:
            jump_fx.play()
            self.v_y = -10
            self.jump = True
        if not key[pygame.K_SPACE]:
            self.jump = False
        if key[pygame.K_LEFT]:
            dx -= 3
            self.cntr += 1
            self.drctn = -1
        if key[pygame.K_RIGHT]:
            dx += 3
            self.cntr += 1
            self.drctn = 1
        if not key[pygame.K_LEFT] and not

```

```

key[pygame.K_RIGHT]:
    self.cntr = 0
    self.indx = 0
    if self.drctn == 1:
        self.im = self.ims_right[self.indx]
    if self.drctn == -1:
        self.im = self.ims_left[self.indx]

if self.cntr > walk_cd:
    self.cntr = 0
    self.indx += 1
    if self.indx >= len(self.ims_right):
        self.indx = 0
    if self.drctn == 1:
        self.im = self.ims_right[self.indx]
    if self.drctn == -1:
        self.im = self.ims_left[self.indx]

self.v_y += 1
if self.v_y > 10:
    self.v_y = 10
dy += self.v_y

self.in_air = True
for tile in night.tile_list:
    if tile[1].colliderect(self.rect.x +
dx, self.rect.y, self.width, self.hght):
        dx = 0
        if tile[1].colliderect(self.rect.x,
self.rect.y + dy, self.width, self.hght):

```

```

        if self.v_y < 0:
            dy = tile[1].bottom -
self.rect.top

            self.v_y = 0
        elif self.v_y >= 0:
            dy = tile[1].top -
self.rect.bottom

            self.v_y = 0
            self.in_air = False

        for platform in platform_group:
            if
platform.rect.colliderect(self.rect.x + dx,
self.rect.y, self.width, self.hght):
                dx = 0
                if
platform.rect.colliderect(self.rect.x, self.rect.y +
dy, self.width, self.hght):
                    if abs((self.rect.top + dy) -
platform.rect.bottom) < col_thresh:
                        self.v_y = 0
                        dy = platform.rect.bottom -
self.rect.top
                    elif abs((self.rect.bottom + dy) -
platform.rect.top) < col_thresh:
                        self.rect.bottom =
platform.rect.top - 1
                        self.in_air = False
                        dy = 0
                    if platform.move_x != 0:

```

```

        self.rect.x +=
platform.move_direction

        if pygame.sprite.spritecollide(self,
all_spiders, False):
            game_over = -1
            gameover_fx.play()
        if pygame.sprite.spritecollide(self,
all_bats, False):
            game_over = -1
            gameover_fx.play()
        if pygame.sprite.spritecollide(self,
all_webs, False):
            game_over = -1
            gameover_fx.play()
        if pygame.sprite.spritecollide(self,
all_spikes, False):
            game_over = -1
            gameover_fx.play()
        if pygame.sprite.spritecollide(self,
all_goos, False):
            game_over = -1
            gameover_fx.play()
        if pygame.sprite.spritecollide(self,
exit_group, False):
            game_over = 1

        self.rect.x += dx
        self.rect.y += dy

```

```

elif game_over == -1:

    self.im = self.dead_im

    if self.rect.y > 140:
        self.rect.y -= 3

    screen.blit(bad_end_im, (0, 0))

    draw_text("THE END?", font, red,
(screen_width // 2) - 180, screen_height // 2 - 90,
screen)

    screen.blit(self.im, self.rect)

    return game_over

def reset(self, x, y):
    self.ims_right = []
    self.ims_left = []
    self.indx = 0
    self.cntr = 0
    for n in range(1, 9):
        im_right =
pygame.image.load(f"hero{n}.png")
        im_right = pygame.transform.scale(im_right,
(28, 56))
        im_left = pygame.transform.flip(im_right,
True, False)

```

```

        self.ims_right.append(im_right)
        self.ims_left.append(im_left)
    self.dead_im = pygame.image.load("ghost.png")
    self.dead_im =
pygame.transform.scale(self.dead_im, (28, 56))
    self.im = self.ims_right[self.indx]
    self.rect = self.im.get_rect()
    self.drctn = 0
    self.width = self.im.get_width()
    self.hght = self.im.get_height()
    self.rect.x = x
    self.rect.y = y
    self.v_y = 0
    self.jump = False
    self.in_air = True

```

```

class Button:

```

```

    def __init__(self, x, y, image):
        self.image = image
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
        self.clicked = False

```

```

    def draw(self, screen):
        action = False
        pos = pygame.mouse.get_pos()
        if self.rect.collidepoint(pos):
            if pygame.mouse.get_pressed()[0] == 1 and

```

```

self.clicked == False:
    action = True
    self.clicked = True
if pygame.mouse.get_pressed()[0] == 0:
    self.clicked = False
screen.blit(self.image, self.rect)
return action

class NightWorld:
    def __init__(self, data, tile_size, all_spiders,
all_bats, all_goos,
                all_webs, all_spikes, firefly_group,
exit_group, platform_group):
    self.tile_list = []
    dirt_im = pygame.image.load("dirt.png")
    grass_im = pygame.image.load("grass.png")

    row_count = 0
    for row in data:
        col_count = 0
        for tile in row:
            if tile == 1:
                im =
pygame.transform.scale(dirt_im, (tile_size, tile_size))
                im_rect = im.get_rect()
                im_rect.x = col_count * tile_size
                im_rect.y = row_count * tile_size
                tile = (im, im_rect)
                self.tile_list.append(tile)

```

```

        if tile == 2:
            im =
pygame.transform.scale(grass_im, (tile_size,
tile_size))

            im_rect = im.get_rect()
            im_rect.x = col_count * tile_size
            im_rect.y = row_count * tile_size
            tile = (im, im_rect)
            self.tile_list.append(tile)
        if tile == 3:
            spider = Enemy1(col_count *
tile_size, row_count * tile_size + 15)
            all_spiders.add(spider)
        if tile == 4:
            platform = Platform(col_count *
tile_size, row_count * tile_size, 1, 0, tile_size)
            platform_group.add(platform)
        if tile == 5:
            platform = Platform(col_count *
tile_size, row_count * tile_size, 0, 1, tile_size)
            platform_group.add(platform)
        if tile == 6:
            web = Web(col_count * tile_size,
row_count * tile_size + (tile_size // 2), tile_size)
            all_webs.add(web)
        if tile == 7:
            firefly = FireFly(col_count *
tile_size + (tile_size // 2),
                                row_count *
tile_size + (tile_size // 2), tile_size)

```

```

        firefly_group.add(firefly)
    if tile == 8:
        exit = Exit(col_count * tile_size,
row_count * tile_size - (tile_size // 2), tile_size)
        exit_group.add(exit)
    if tile == 9:
        bat = Enemy2(col_count * tile_size,
row_count * tile_size + 15)
        all_bats.add(bat)
    if tile == 10:
        spike = Spike(col_count *
tile_size, row_count * tile_size + (tile_size // 2),
tile_size)

        all_spikes.add(spike)
    if tile == 11:
        goo = Enemy3(col_count * tile_size,
row_count * tile_size + 15)
        all_goos.add(goo)
        col_count += 1
        row_count += 1

def draw(self, screen):
    for tile in self.tile_list:
        screen.blit(tile[0], tile[1])

class Enemy1(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("enemy.png")

```

```

        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
        self.move_drctn = 1
        self.move_cntr = 0

    def update(self):
        self.rect.x += self.move_drctn
        self.move_cntr += 1
        if abs(self.move_cntr) > 35:
            self.move_drctn *= -1
            self.move_cntr *= -1

class Enemy2(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("bat.png")
        self.image = pygame.transform.scale(self.image,
(33, 25))
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
        self.move_drctn = 1
        self.move_cntr = 0

    def update(self):
        self.rect.y += self.move_drctn
        self.move_cntr += 1
        if abs(self.move_cntr) > 35:

```

```

        self.move_drctn *= -1
        self.move_cntr *= -1

class Enemy3(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("goo.png")
        self.image = pygame.transform.scale(self.image,
(60, 60))
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
        self.move_drctn = 1
        self.move_cntr = 0

    def update(self):
        self.rect.x += self.move_drctn
        self.move_cntr += 3
        if abs(self.move_cntr) > 50:
            self.move_drctn *= -1
            self.move_cntr *= -1

class Web(pygame.sprite.Sprite):
    def __init__(self, x, y, tile_size):
        pygame.sprite.Sprite.__init__(self)
        img = pygame.image.load("web.png")
        self.image = pygame.transform.scale(img,
(tile_size, tile_size // 2))

```

```

        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y

class Spike(pygame.sprite.Sprite):
    def __init__(self, x, y, tile_size):
        pygame.sprite.Sprite.__init__(self)
        img = pygame.image.load("spikes.png")
        self.image = pygame.transform.scale(img,
(tile_size, tile_size // 2))
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y

class FireFly(pygame.sprite.Sprite):
    def __init__(self, x, y, tile_size):
        pygame.sprite.Sprite.__init__(self)
        img = pygame.image.load("firefly.png")
        self.image = pygame.transform.scale(img,
(tile_size, tile_size))
        self.rect = self.image.get_rect()
        self.rect.center = (x, y)

class Exit(pygame.sprite.Sprite):
    def __init__(self, x, y, tile_size):
        pygame.sprite.Sprite.__init__(self)
        img = pygame.image.load("exit.png")

```

```

        self.image = pygame.transform.scale(img,
(tile_size, int(tile_size * 1.5)))
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y

class Platform(pygame.sprite.Sprite):
    def __init__(self, x, y, move_x, move_y,
tile_size):
        pygame.sprite.Sprite.__init__(self)
        image = pygame.image.load("m_platf.png")
        self.image = pygame.transform.scale(image,
(tile_size, tile_size // 2))
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y
        self.move_counter = 0
        self.move_direction = 1
        self.move_x = move_x
        self.move_y = move_y

    def update(self):
        self.rect.x += self.move_direction *
self.move_x
        self.rect.y += self.move_direction *
self.move_y
        self.move_counter += 1
        if abs(self.move_counter) > 35:

```

```
self.move_direction *= -1  
self.move_counter *= -1
```

Файл game_logger.dll