Автономная некоммерческая организация дополнительного профессионального образования «Центр обучения «Махаон»

Cyber Ravage

**Выполнил:**

Басов Илья Анатольевич

Ученик 9 класса

Школы №2070

**Руководитель:**

Морозов Антон Дмитриевич

Москва 2025

Оглавление

[Введение 2](#_Toc193016193)

[Цель: 2](#_Toc193016194)

[Задачи: 2](#_Toc193016195)

[Описание 3](#_Toc193016196)

[Сюжет 3](#_Toc193016197)

[Запуск 3](#_Toc193016198)

[Игровой процесс 4](#_Toc193016199)

[Техническая реализация 4](#_Toc193016200)

[Выводы 5](#_Toc193016201)

[Источники 6](#_Toc193016202)

# Введение

Создание собственных проектов – серьёзная работа. По мере создания разработчик способен научиться чему-то новому, освоить сложные для него моменты и научиться пользоваться новыми технологиями. Но именно это и позволяет получить ему ценную практику, становиться лучше и лучше, чтобы не проиграть гонку жизни.

Так и я, чтобы улучшить свои навыки программирования, лучше разобраться с ООП, попробовать использовать GIT, и в итоге набраться опыта решил, что проектом, который удовлетворит все мои потребности, будет *игра на python*. Это и улучшит мои навыки в разработки на данном языке, и поможет лучше понять взаимодействия между объектами.

Цель: Познакомиться с созданием игр, для лучшего освоения языка программирования python и работы с объектами.

Задачи:

* Определиться с библиотекой
* Сделать концепцию (модель) игры
* Написать сюжет
* Найти картинки (спрайты), нарисовать анимации
* Начать делать проект, изучив библиотеку
* Тестирование и отладка проекта

# Описание

Существует множество способов для создания игр: писать код самому с нуля, использование игровых движков, использование конструкторов, использование заготовок и шаблонов.

**Полное самостоятельное написание программного продукта с нуля**. Этот способ самый сложный и требует много времени, однако позволяет лучше разобраться в среде, в которой пишется код.

**Использование игровых движков**. Они автоматизируют многие процессы разработки и сводят к минимуму требования к разработчику в части знания языков программирования. Некоторые популярные движки: Unity, Unreal Engine, GameMaker.

**Применение конструкторов видеоигр**. Они не требуют специальных навыков программирования. Конструкторы предоставляют графический интерфейс и простые инструменты для создания игровых объектов, настройки правил и уровней.

**Использование готовых шаблонов или заготовок игры**. Этот способ самый простой и быстрый. Некоторые компоненты шаблона можно заменить на свои.

Для разработки я решил выбрать библиотеку «pygame», так как этот способ лучше других удовлетворял мои цели, а сама библиотека по статистике самая лучшая и используемая библиотека для разработки игр на python.

Моя игра является представителем жанра rouglike. При её создании я вдохновлялся следующими играми: soulsknight, dead cells, isaac. Я захотел взять разнообразия атак из soulknight, уровни из dead cells и крафтовую систему из the binding of Isaac.

# Сюжет

В мире случился апокалипсис: мир был захвачен пришельцами. Все люди были уничтожены, а остальные формы жизни роботизированы.

Но остался один выживший! Он собирает себе тело из конечностей своих врагов, тем самым улучшая себя. Его главная цель – отомстить за весь свой вид, убив центральный мозг, управляющий всеми пришельцами.

# Запуск

При запуске игры вы оказываетесь в меню с 3 кнопками: играть, управление и выход.

При нажатии кнопки «Играть» на выбор будут предоставлены уровни (пока что только 1). После нажатия соответственный уровень будет запущен с начала.

При нажатии кнопки «Управление», открывается меню, в котором рассказывается внутренние управление.

При нажатии кнопки «Выход» происходит выход из игры

# Игровой процесс

На каждом уровне есть локации, которые нужно проходить. В конце каждого уровня есть босс, с которого главный герой и будет получать улучшения.

Боеприпасов у игрока нет, но есть перезарядка на атаки. В будущем планируется создать разные виды оружия(ближние, дальние: стреляющие очередями, дробью, лазерами)

Также планируется увеличить количество локаций и систему прокачки героя. Можно будет комбинировать конечности, получая уникальные эффекты атак и подобного

# Техническая реализация

Для создания игры использовались язык программирования python, библиотека pygame. Спрайты, локации и анимации создавались в программе Aseprite.

Задний фон был задан картинкой повторяющий себя по координате x слева и справа для создания имитации движения по локации.

У каждого объекта в игре есть своя картинка(набор картинок) и коллизия(прямоугольники «rect», которые и фиксируют взаимодействия между объектами).

Код разделён на несколько частей. Первая – реализация объектов, их методов и аргументов. Второй – главный цикл, в котором прописываются все действия совершаемые объектами(методы). И третий – обработчик событий, в котором происходит проверка на нажатия клавиш, истечения таймеров и т.д.

Для реализации движения было создано 2 списка и 1 переменная. Списки хранят в себе картинки ходьбы в левую и правую сторону, а переменная картинку неподвижной модели. Во время нажатия клавиши запускается анимация, а координаты игрока изменяются в зависимости от скорости, хранящейся как аргумент self.speed

Прыжок реализован с помощью функции, проверяющий состояние игрока (находится ли персонаж в состоянии прыжка) и постепенно изменяя его координаты по y.

Пули реализованы с помощью класса-Vector2 в pygame. Определяются координаты курсора, ищется угол с помощью встроенных функций и библиотеки math а далее изменяются координаты по вектору единичного отрезка (вектор (1, 0)) под углом, найденного ранее. Пули хранятся в массиве, являющимся аргументом класса bullet.all. В каждом объекте класса bullet хранится информация о его коллизии в аргументе self.rect.

Враг (пока что один) реализован как отдельный класс, перемещается подобно игроку, следуя за ним. Если враг дотронется игрока, то последнему наносится урон. Если враг касается пули, урон наносится призраку.

Звуковое сопровождение отсутствует, но в скорее будет записано.

Испытать игру можно по моей ссылку на git: https://github.com/Arktoth/Cyber-Ravage

# Выводы

У меня получилось опробовать себя в создании игры, погрузиться в процесс разработки, отладки. Я получил большее понимание о работе с ООП, улучшил навыки программирования на python и попробовал работать с системой контроля версий GIT.

# Источники

1. Полезный канал, помогший разобраться с основами pygame: <https://www.youtube.com/@itproger>
2. Сайт, показывающий реализацию огромного количества функций, и объясняющий логику большинства внутренних игровых механик:  
   <https://pythonru.com/primery/>
3. Сайт со всеми функциями и методами pygame:  
   <https://pg1.readthedocs.io/en/latest/index.html>
4. Сайт, позволяющий задать вопрос более опытным людям, предоставить свой код. Много вопросов, интересующих меня, уже были заданы другими пользователями, что позволило увеличить скорость работы над игрой:   
   <https://pg1.readthedocs.io/en/latest/index.html>
5. Сайт, с описанием реализации передвижения объектов с помощью векторов в pygame:  
   <https://myrusakov.ru/python-pygame_asteroids_part4.html>
6. Сайт, дающий понимание по двухмерным векторам в pygame:  
   <https://thepythoncode.com/article/make-a-planet-simulator-using-pygame-in-python>
7. Нейросеть YandexGPT – в разы ускоряла поиски нужных мне ресурсов, а после длительной работы с ней, специально фильтровала все запросы именно на библиотеку pygame
8. Официальный сайт dead cells: <https://deadcells.com>  
   страница в steam: <https://store.steampowered.com/app/588650/Dead_Cells/>
9. Страница the binding of Isaac в steam: <https://store.steampowered.com/app/113200/The_Binding_of_Isaac/>