Лысьвенский городской округ

Муниципальное автономное общеобразовательное учреждение

«Лицей «ВЕКТОРиЯ»

Индивидуальный проект

**Тема «Создание компьютерной игры на языке программирования python»**

|  |  |
| --- | --- |
| Автор проекта (ФИО) | Колыванов Семён Иванович |
| Класс | 10 «А» |
| Руководитель проекта (ФИО) | Белова Наталья Владимировна |
| Оценка проекта руководителем (количество баллов, подпись) |  |
| Эксперты (ФИО) |  |
| Оценка защиты проекта (количество баллов, подписи экспертов) |  |
| Баллы за уникальность |  |
| Всего баллов |  |
| Итоговая оценка за проект |  |
| С итоговой оценкой ознакомлен (подпись учащегося) |  |

Лысьва, 2021-2022

**Оглавление**

[Введение 2](#_Toc103520323)

[Глава 1 - Обзор Литературы 3](#_Toc103520324)

[Глава 2 - Практическая часть 4](#_Toc103520325)

[2.1 – Выбор языка программирования 4](#_Toc103520326)

[2.2 – Выбор GUI библиотеки 4](#_Toc103520327)

[2.3 – Разработка 4](#_Toc103520328)

[2.3.1 – Создание шаблона Pygame 4](#_Toc103520329)

[2.3.2 – Создание игрока 5](#_Toc103520330)

[2.3.3 – Создание карты 5](#_Toc103520331)

[2.3.4 – Столкновение со стенами 5](#_Toc103520332)

[2.3.5 – Камера игрока 6](#_Toc103520333)

[2.3.6 – Создание другого измерения 6](#_Toc103520334)

[2.3.7 – Текстурирование уровня и персонажа 6](#_Toc103520335)

[2.3.8 – 2.5D Top-Down представление 6](#_Toc103520336)

[2.3.9 – 3-уровневая карта уровня 7](#_Toc103520337)

[2.3.10 – Двери 7](#_Toc103520338)

[2.3.11 – Переносной куб 7](#_Toc103520339)

[2.3.12 – Кнопки 7](#_Toc103520340)

[2.3.13 – Связь объектов 7](#_Toc103520341)

[2.3.14 – Меню игры 8](#_Toc103520342)

[2.3.15 – Анимация персонажа 8](#_Toc103520343)

[2.3.16 – Лазер 8](#_Toc103520344)

[2.3.17 – Приёмник лучей 8](#_Toc103520345)

[2.3.18 – Виды кубов 9](#_Toc103520346)

[2.3.19 – Создание уровней 9](#_Toc103520347)

[Заключение 9](#_Toc103520348)

[Приложение 10](#_Toc103520349)

[Ссылки 10](#_Toc103520350)

[Рисунки 10](#_Toc103520351)

[Используемая литература 15](#_Toc103520352)

# Введение

Компьютерные игры стали неотъемлемой частью нашего мира. Создание компьютерных игр является одним из самых мощных и интересных направлений современной компьютерной индустрии. Компьютерные игры наших дней необыкновенно сложны. Сейчас над играми работают огромные студии и корпорации. Но можно делать игры и самому, в среднем, каждый месяц в Google Play выпускается более 100 000 новых приложений. Это означает то, что множество маленьких «инди» студий каждый день создают множество игр. Научиться создавать игры, изучить процесс и создать собственную видеоигру решил и я.

**Актуальность:** игровая и ИТ индустрия одни из самых востребованных отраслей в мире, но знание о них очень ограничены

**Объект:** разработка компьютерной игры

**Предмет:** технологии разработки компьютерной игрыжанра головоломка

**Цель:** разработать прототип компьютерной игры жанра головоломка по средствам среды программирования python

**Задачи:**

1. Анализ литературы
2. Создать игру
3. Собрать отзыва об игре

# Глава 1 - Обзор Литературы

Компьютерная игра — компьютерная программа, служащая для организации игрового процесса. Компьютерные игры часто создаются на основе сторонних источников, таких как фильмы или книги, но в основном для игры придумывают свой мир.

Компьютерные игры оказали столь существенное влияние на общество, что в последнее время в информационных технологиях появилась устойчивая тенденция к геймификации для неигрового прикладного программного обеспечения, например технические симуляторы.

Более того специально разработанные игры могут выступать в качестве обучающего материала или позволяют использовать игроков в научно-исследовательских целях. Такие игры редко выпускаются в широкие массы.

Так же компьютерные игры с 2011 года официально признаны правительством США и американским Национальным фондом отдельным видом искусства, наряду с театром и кино, а в России киберспорт официально был причислен к видам спорта (Смотри ссылку 1).

Из всего этого следует, что компьютерные игры плотно влились в нашу нынешнюю жизнь. Более того, сфера их использования за последние 10 лет сильно выросла, теперь игры используют не только для развлечения и отдыха, но и для обучения людей и проведения научных исследований.

Компьютерные игры классифицируют по нескольким основным признакам:

• жанр;

• количество игроков;

• визуальное представление;

• платформа.

Жанр игры определяется целью и основной механикой игры.

Игровая механика, или механизм — способ взаимодействия с игровыми объектами в рамках установленных ограничений игры. Взаимодействие изменяет состояние игры, что влияет на последующие решения игрока.

# Глава 2 - Практическая часть

## 2.1 – Выбор языка программирования

С самого начала я выбрал язык программирования. Существует множество языков программирования: Java, C, Pascal. Но я остановился на языке Python. Плюсы Python: простой минималистичный синтаксис (код легко писать, читать и поддерживать), большая стандартная библиотека и много дополнительных библиотек, кроссплатформенность и поддержка почти всех современных систем. Минусы Python: низкая производительность, плохо подходит для разработки мобильных приложений.

## 2.2 – Выбор GUI библиотеки

Python самостоятельно не может отображать графику по этому необходимо подключать сторонние библиотеки. Ещё GUI библиотеки могут считывать ввод данных с клавиатуры и мыши. Библиотек множество, потому что их пишут пользователи языка. Я выбрал библиотеку «pygame», потому что она сделана специально под создание игр, а не приложений, как другие GUI библиотеки.

## 2.3 – Разработка

Для создания игры надо иметь базовые навыки владения языком программирования.

### 2.3.1 – Создание шаблона Pygame

Главным элементом для работы игр является игровой цикл. Он запускается снова и снова, делая все, чтобы работала игра. Каждый цикл в игре называется кадром.

В каждом кадре происходит множество действий, которые делиться на три категории:

1. Обработка ввода (события)

Речь идет обо всем, что происходит вне игры — тех событиях, на которые она должна реагировать. Это могут быть нажатия клавиш на клавиатуре, клики мышью и так далее.

1. Обновление игры

Изменение всего, что должно измениться в течение одного кадра. Если персонаж движется к стене, то его должны оттолкнуть.

1. Рендеринг (прорисовка)

В этом шаге все выводится на экран: фоны, персонаж, меню. Все, что игрок должен видеть, появляется на экране в нужном месте.

Теперь можно создавать шаблон pygame который будет открывать окно игры. Начнём с запуска окна (Смотри рисунок 1). Так же необходимо создать файл настроек приложения и ввести туда необходимые данные (Смотри рисунок 2).

Теперь переходим к основному циклу. Пишу цикл while который будет работать пока running будет True и чтобы выключить игру надо просто сделать running False. Создаю цикл for для обработки событий (Смотри рисунок 3)

Сейчас я пропущу обновление игры, потому что приложению пока нечего обновлять и приступим к рендерингу. Пишу screen.fill(BLACK) это значит заполнения окна приложения чёрным и pygame.display.flip() чтобы экран обновлялся.

Всё! Я создал просто открывающиеся окно.

### 2.3.2 – Создание игрока

Чтобы создать персонажа, я решил использовать метод class. В функции инициализация я указываю позицию игрока, размер и скорость. Также создаю функцию движения, которая будет менять координаты x и y в зависимости от нажатой клавиши (Смотри рисунок 4). При создании этой функции надо иметь ввиду, что начало координат в pygame начинается в верхнем левом углу. Функцию надо добавить в область обновления игры (Смотри рисунок 5).

Теперь надо нарисовать его в окне приложения. С помощью функции рисования программа будет рисовать круг на позиции игрока

Теперь у мeня получилась игра с двигающимся кругом в окне

### 2.3.3 – Создание карты

Карта будет представлять из себя двухмерный массив (Смотри рисунок 6). Массив удобно просматривать с помощью цикла for и отрисовать его в окне. Так же надо иметь в виду что в настройках я обозначил размер одной клетки. Затем я пишу функцию отображения карты, она проходит по очереди весь массив и выводит каждую клетку функцией отображения квадрата (Смотри рисунок 7).

### 2.3.4 – Столкновение со стенами

В класс игрока надо записать функцию столкновения. Она проходит массив и если игрок вступает в границу с клеткой, то сначала проверяет со стороны x или стороны y, потом с какой именно стороны и только потом отталкивает игрока в другом направлении. Так же она учитывает размер и скорость персонажа (Смотри рисунок 8). Вношу функцию в область обновления игры.

### 2.3.5 – Камера игрока

Теперь я сделаю так чтобы игрок был посередине, а все объекты перемещались относительно перемещения игрока. Чтобы сделать так, надо прибавить ко всем объектом в отрисовке дополнительные координаты относительные игрока.

### 2.3.6 – Создание другого измерения

Я перешёл к сознанию одной из основных механик(действий) игры: смещение измерения. Чтобы создать такую механику, нужно для начала создать это измерение, поэтому я создал на карте уровня идентичную комнату на расстояние 11 клеток от основной комнаты, чтобы другого измерения не было видно из комнаты.

Теперь, чтобы сместить измерение, надо просто поменять координату игрока на ширину комнаты плюс 11 клеток. Создаю функцию телепортации в классе игрока и добавляю её в область обновления игры. Также добавляю проверку нажатие кнопки в область ввода данных.

### 2.3.7 – Текстурирование уровня и персонажа

Сейчас игра выглядит как сборка зелёных квадратов и одного круга, поэтому, чтобы выглядело красиво, надо рисовать отдельные картинки(текстуры) на каждой клетке уровня. Pygame умеет загружать картинки, редактировать их и отображать в окне приложения. Чтобы отобразить их, нужно просто заменить функцию рисования квадрата на рисование текстуры. Также чтобы функция рисования находила нужную текстуру надо обозначить каждой клетке названия файла картинки (Смотри рисунок 9). И теперь, создавая новый вид клетки надо просто добавить в этот словарь.

С персонажем тоже самое: только его не надо добавлять в словарь текстур, потому что у него отдельные текстуры.

### 2.3.8 – 2.5D Top-Down представление

2.5D Top-Down перспектива – это представление объекта, у которого видно только вверх и передняя стенка (Смотри рисунок 10). 2.5D графика – это 2D графика с иллюзией 3D. Чтобы создать такую графику надо отображать клетки сверху до низа, потому что клетка должна перекрывать стену клетки. Так же стена должна перекрывать игрока, если игрок стоит за ней, поэтому отрисовку персонажа надо переставить в функцию отрисовки карты, чтобы просчитать, когда игрок стоит за стеной и отобразить её позже.

### 2.3.9 – 3-уровневая карта уровня

Чтобы проще рисовать карту, я буду отделять уровень пола, уровень стен и уровня потолка. Уровень пола будет рисоваться первым. Уровень стен будет рисоваться вторым и с учётом игрока. Уровень потолка будет рисоваться позже всех.

### 2.3.10 – Двери

Двери будут представлять из себя класс со значениями: координат, направления и открыта ли дверь (Смотри рисунок 11). В класс добавляю функцию проверки приближения игрока к двери и последующего его открытия и добавляю её в область обновления игры. Потом добавляю в функцию столкновения проверку столкновения с дверью. В функцию сканирования карты добавляю обозначение двери и функцию выявления координат и направления двери(если рядом с дверью стены сверху и снизу, то дверь вертикальная, если рядом с дверью стены справа и слева - дверь горизонтальная).

### 2.3.11 – Переносной куб

Куб будет представлять из себя класс со значениями: координат и находиться ли он у игрока (Смотри рисунок 12). Чтобы можно было подбирать куб, я записываю дополнительный параметр в класс игрока, в который будет записываться, какой куб он сейчас держит, и добавляю функцию подбора кубика, проверяющую нахождение игрока у куба и держит ли он куб и нажата ли кнопка подбора кубика. После этого функция записывает этот куб в параметр игрока и даёт кубику координаты за пределами карты, чтобы его не было видно.

Чтобы опускать куб, надо проверять, ничего ли не мешает его опустить. После даёт кубу позицию игрока и выписывает куб из параметра игрока.

### 2.3.12 – Кнопки

Кнопка будет представлять из себя класс со значениями: координат и активирована ли она. Функция активации будет проверять, находиться ли на кнопке игрок или куб, и активировать кнопку или деактивировать.

### 2.3.13 – Связь объектов

Чтобы кнопка могла открывать дверь, надо добавить связь между ними, поэтому я записываю в класс кнопки и двери параметр связи, который будет варьироваться от 1 до 9, и где 0 – нет связи. Эта связь будет записываться в функции сканирование карты, а для этого нужно добавить новый четвёртый уровень карты, где будет связь объектов. Теперь осталось проверять, активен ли источник, и активировать приёмник, для этого надо написать проверяющую функцию и добавить её в область обновления игры.

### 2.3.14 – Меню игры

Пришло время создавать меню игры. С этим предстоят некоторые сложности, потому что придётся поделить игровой цикл на сцену меню и сцену игры, также придётся отдельно прописывать области цикла.

Чтобы поделить цикл, надо проверять, в какой сцене находиться игрок, и в начале кода прописать переменные сцен (Смотри рисунок 13).

Теперь создаю кнопки меню, которые будут представлять из себя классы со значениями: координат верхнего левого угла, высота, ширина, текст кнопки и что она делает. Добавляю в её класс функцию проверки позиции курсора и функцию нажатия левой кнопки мыши.

В меню я добавлю 3 кнопки: продолжить, уровни, выход. Кнопка «продолжить» будет переносить в игру, где остановился игрок. Кнопка «уровни» - на сцену с выбором уровня.

### 2.3.15 – Анимация персонажа

Один из сложнейших разделов, потому что анимация зависит от времени, поэтому первым делом добавляем счётчик времени в игровой цикл. Чтобы его добавить надо считать каждую итерацию цикла, после чего делить это число на FPS(кадры в секунду), чтобы получить результат в секундах.

Теперь надо нарисовать анимацию. Двухмерная анимация состоит из кадров, которые идут по очереди, поэтому надо нарисовать каждый кадр (Смотри рисунок 14).

Теперь я создаю функцию анимации в класс игрока, которая будет менять текстуру игрока в зависимости от времени.

### 2.3.16 – Лазер

Во-первых, я создал класс лазера со значениями: координат, включён ли он, массив лучей, направления и параметр связи. Теперь пишу функцию в классе лазера, добавляющая луч в массив лучей и записывающая координаты и направление, и которая будет учитывать столкновение с кубами и стенами.

### 2.3.17 – Приёмник лучей

Приёмник лучей будет представлять из себя класс со значениями: координат, активен ли он и параметра связи. Добавляю функцию проверки в класс приёмника, которая будет проверять все лучи на попадание в приёмник.

### 2.3.18 – Виды кубов

Настало время разнообразить механику куба. Я придумал ещё три вида куба: лазерный, квантовый и квантовый лазерный.

Лазерный куб будет перенаправлять лучи в сторону, куда смотрит куб, из-за этого придётся в класс куба добавить параметр направления, и в функцию лучей в классе лазера добавить учёт лазерного куба.

Квантовый куб будет находиться сразу в двух измерениях, и если игрок перетаскивает куб в одном измерении, то он и двигается в другом. Поэтому я добавляю в класс куба параметр связи кубов, в который будет записываться куб, который находиться в другом измерении, и добавляю в класс куба функцию, которая будет создавать куб в другом измерении и удалять его, когда игрок поднимает куб.

Квантовый лазерный куб будет перемещать лазер в другое измерение. Работает почти на тех же функциях, как и предыдущее два вида куба.

### 2.3.19 – Создание уровней

Самое длительным этапом в этом проекте было создание уровней, так как придумывать оригинальные неповторяющиеся уровни очень сложно! Но я поделил уровень на 3 части: вход, сам уровень, выход. Вход и выход почти на всех уровнях выглядит одинаково, поэтому начинать было проще. Вот пример уровня в файле: (Смотри рисунок 15), и в игре: (Смотри рисунок 16).

# Заключение

Создание игры - долгое и сложное занятие, но я получил бесценный опыт в программировании, и разрабатывать игры мне очень понравилось. Придумывать каждую механику игры и прорабатывать код механики было очень интересно и даже захватывающе, так как я сразу мог видеть результат своей работы.

Так же я провёл тестирование своей игры на учащихся своего класса и собрал отзывы. На основе этих отзывов я сделал вывод что игра всем понравилась.

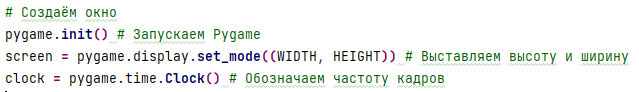
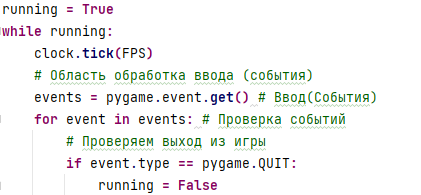
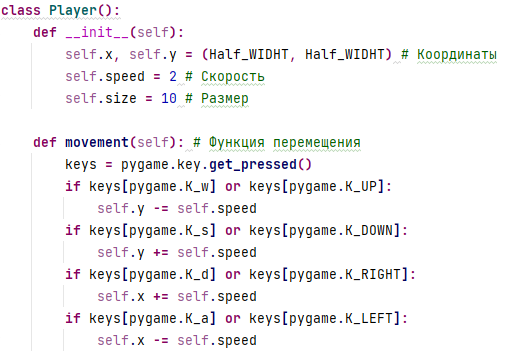
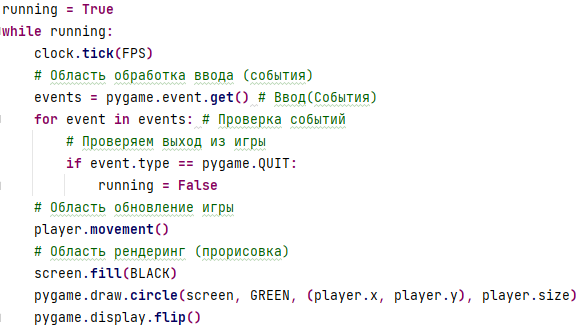
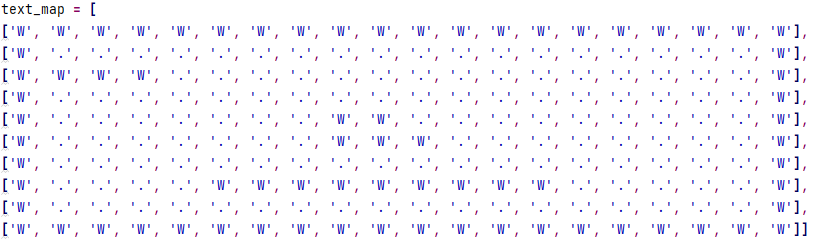
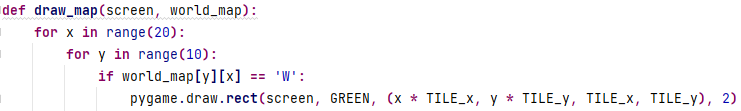
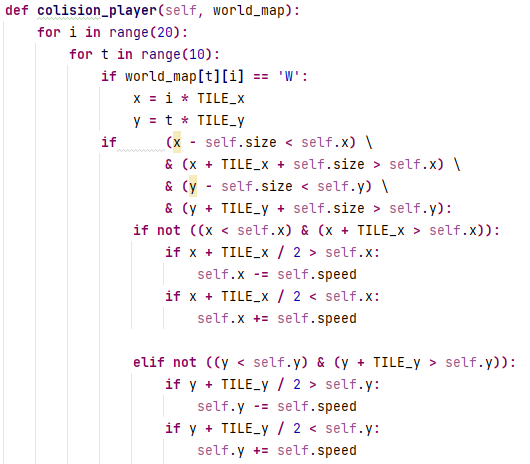
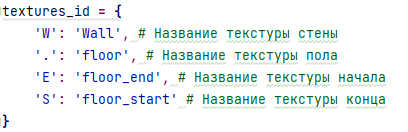
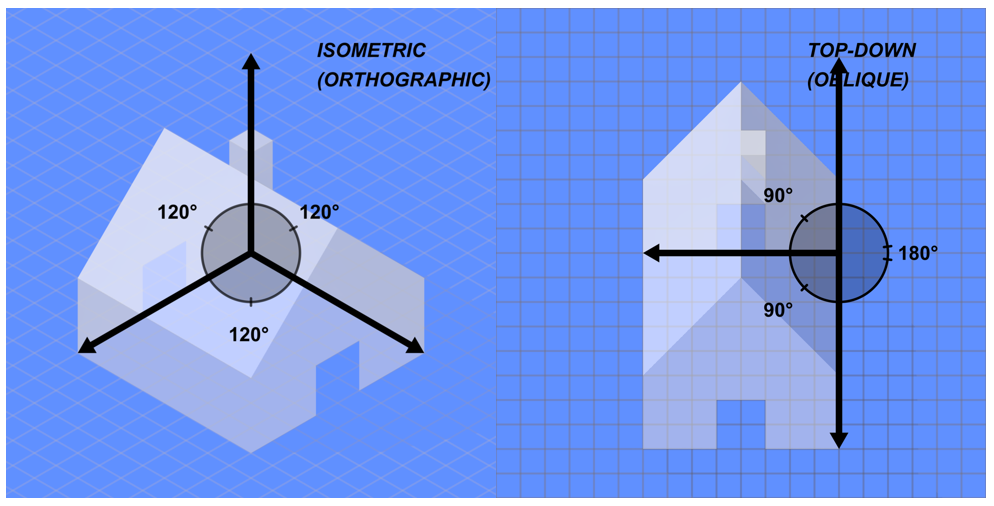
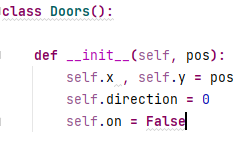
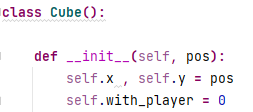
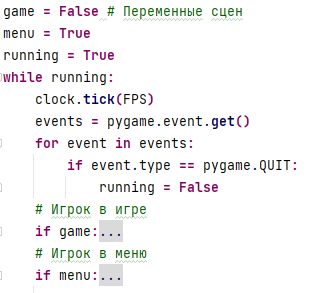
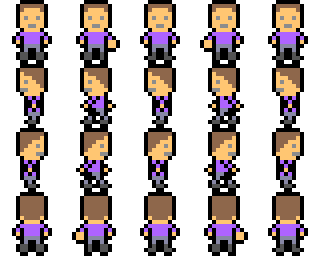
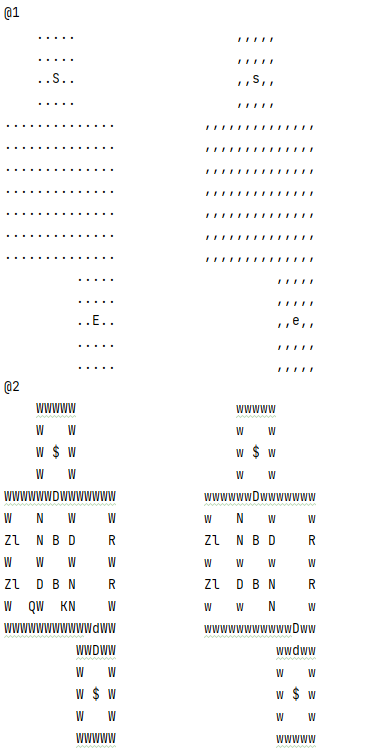
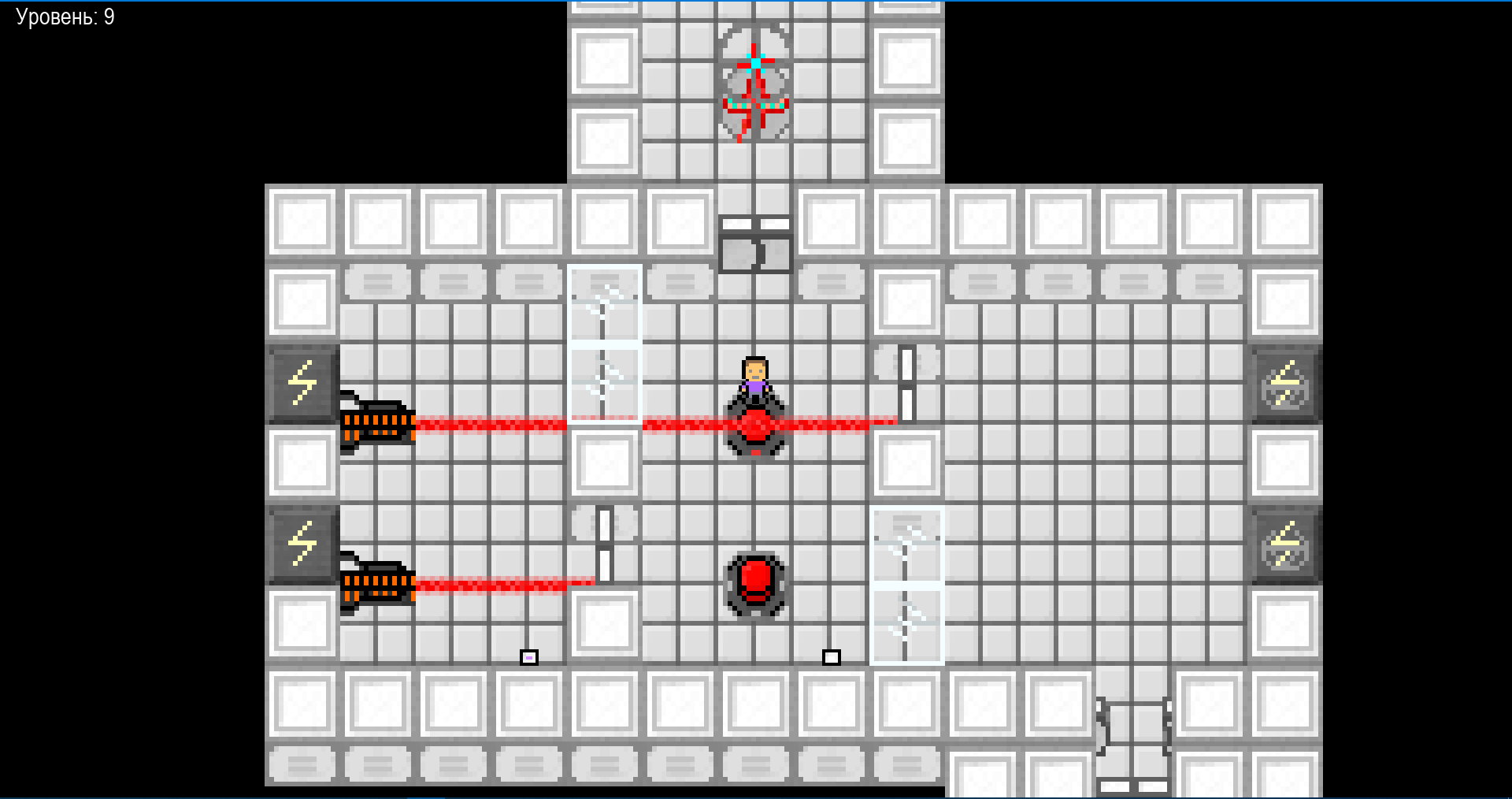
Так же весь код я выкладывал на свою страницу на Github (Смотри ссылку 2). Там можно посмотреть весь код и историю его изменения.

# **Приложение**

## Ссылки

1. <http://publication.pravo.gov.ru/Document/View/0001201704140005?index=1&rangeSize=1>
2. <https://github.com/semakol/game1.1>

## Рисунки

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 
10. 
11. 
12. 
13. 
14. 
15. 
16. 

# Используемая литература

1. <https://pythonru.com/uroki/biblioteka-pygame-chast-1-vvedenie>
2. <https://en.wikipedia.org/wiki/Graphical_user_interface>