#### 1. ****Общие сведения****

* **Название проекта**: 2D-платформер "Gribok".
* **Цель проекта**: Создание простой 2D-игры с элементами платформера, включающей несколько уровней, управление персонажем, врагов, сбор монет и систему подсчета очков.
* **Целевая аудитория**: Пользователи, интересующиеся casual-играми и платформерами.

#### 2. ****Функциональные требования****

* **Основной игровой процесс**:

**·** Управление персонажем (движение влево/вправо, прыжки).

**·** Взаимодействие с объектами (сбор монет, столкновение с врагами).

**·** Завершение уровня через достижение двери выхода.

* **Уровни**:

**·** Два уровня с разными картами и расположением врагов.

**·** Возможность выбора уровня из меню.

* **Враги**:

**·** Два типа врагов с разным поведением:

Тип 1: Патрулирует заданную область.

Тип 2: Атакует игрока при приближении.

* **Монеты**:

Сбор монет увеличивает счет игрока.

* **Анимации**:

Анимация персонажа, врагов и обьектов на карте.

* **Звуковые эффекты**:

Звук при поражении, победе, взаимодействии с обьектами на карте и частями интерфейса.

* **Музыка**:

Фоновая музыка, которая играет в цикле.

* **Меню и настройки**:

**·** Главное меню с кнопками:

"Play" — переход к выбору уровня.

"Options" — настройки (полноэкранный режим, отключение музыки).

"Quit" — выход из игры.

**·** Меню выбора уровня.

**·** Меню настроек с возможностью включения/выключения полноэкранного режима и музыки.

* **Система подсчета очков**:

**·** Подсчет очков за сбор монет и победу над врагами.

**·** Сохранение статистики (счет, количество монет, количество побежденных врагов) в CSV-файл.

* **Обработка завершения игры**:

**·** Поражение: при падении в воду или столкновении с врагом.

**·** Победа: при достижении двери выхода.

**·** Возможность продолжить игру после поражения или вернуться в меню.

#### 3. ****Нефункциональные требования****

* **Удобство интерфейса**:

**·** Интуитивно понятное управление.

**·** Четкие подсказки и текстовые сообщения.

* **Кроссплатформенность**:

Игра должна работать на Windows, macOS и Linux.

* **Масштабируемость**:

Возможность добавления новых уровней и механик без значительных изменений в коде.

#### 4. ****Технические требования****

* **Язык программирования**: Python.
* **Библиотеки**:

Pygame — для создания игровой логики и графикии

* **Ресурсы**:

**·** Графика: спрайты для персонажа, врагов, монет, фонов и кнопок.

**·** Звуки: звуковые эффекты и фоновая музыка.

* **Структура проекта**:

**·** Основной файл main.py main.pyma входа в игру.

**·** Папка assets для хранения графики и звуков.

**·** Папка classes classec для хранения классов игры (персонаж, враги, монеты и т.д.).

#### 5. ****Этапы разработки****

1. **Подготовка**:

**·** Установка Python и Pygame.

**·** Создание структуры проекта.

1. **Разработка базовой логики**:

**·** Создание классов для персонажа, врагов, монет и других объектов.

**·** Реализация управления персонажем.

1. **Создание уровней**:

**·** Разработка карт для уровней.

**·** Добавление врагов и монет.

1. **Реализация меню и настроек**:

**·** Создание главного меню.

**·** Добавление меню выбора уровня и настроек.

1. **Добавление звуков и музыки**:

Интеграция звуковых эффектов и фоновой музыки.

1. **Тестирование:**

**·** Проверка работы всех механик и интерфейса.

**·** Исправление багов

1. **Финальная доработка**:

**·** Оптимизация кода.

**·** Добавление комментариев и документации.

#### 7. ****Дополнительные требования****

**·** Возможность добавления новых уровней через редактирование текстовых файлов карт.

**·** Поддержка полноэкранного режима и изменения громкости музыки.