**Doodle Jump**

**План разработки ПО**

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
| 1 | Описание продукта | 2 |
| 2. | Описание команды | 3 |
| 2.1. | Требуемые знания/опыт | 3 |
| 2.2. | Члены команды и их роли | 4 |
| 3 | Описание процесса модели разработки | 5 |
| 3.1. | Модель разработки | 5 |
| 3.2. | Фазы разработки | 5 |
| 3.3. | Введение разработки | 5 |
| 4 | Суть проекта | 6 |
| 5 | План создания проекта | 7 |
| 6 | Контроль версий | 8 |
| 6.1. | Правила введения репозитория | 8 |
| 6.2 | Оформления коммитов | 8 |
| 6.3 | Примеры коммитов | 8 |
| 7 | Используемые инструменты | 9 |

**1. Описание продукта**

**Тип игры**: Web-PC

**Жанр:** аркада

**Стиль:** 2D

**Целевая аудитория:** любые возраста

**Типовой сеанс игры:** При входе в игру появляется главное меню, где пользователь нажимает начать игру. При запуске игры, его персонаж прыгает по платформам, задача пользователя(игрока) прыгать как можно выше, поднимая свой рекорд. Если игрок упадёт с платформы вниз, то он проиграл.

**2. Описание команды**

2.1. Требуемые знания/опыт

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **ПРОЦЕСС РАЗРАБОТКИ** | | |  | | --- | | ТРЕБОВАНИЯ | |
| |  | | --- | | План разработки ПО | | 1. Знание моделей разработки ПО 2. Опыт в разработке подобных   продуктов |
| |  | | --- | | Написание требований к разработке | | |  | | --- | | 1. Знание этапов разработки ПО 2. Полное понимание   разрабатываемого продукта   1. Понимание технических   спецификаций операционных  систем | |
| |  | | --- | | Пользовательский интерфейс(GUI) | | |  | | --- | | Опыт проектирования GUI | |
| |  | | --- | | Реализация | | Знание языка Python, в рамках структурного проектирования и ООП |
| Тестирование | 1)Понимание базовых принципов тестирования (Модульное тестирование)   1. Ручное тестирование 2. Опыт составления тест-планов и написания модульных-тестов |
| Развёртывание | 1. Настройка сборки 2. Сборка проекта |

**2.2. Члены команды, их роли**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | Член команды | | |  | | --- | | Светлов Павел | | |  | | --- | | Рак Дмитрий | |
| |  | | --- | | Основная обязанность | | |  | | --- | | Разработчик основной  игровой логики | | Главный разработчик |
| |  | | --- | | Знания Python | | |  | | --- | | Intern/Junior | | Senior |

**3. Описание модели процесса разработки**

**3.1. Модель разработки**

**Модель разработки**: Waterfall (каскадная) - подразумевает

последовательное прохождение фаз, каждая из которых должна

начаться после завершения предыдущей.

**3.2. Фазы разработки**

**Анализ и сбор требований - 1 неделя**

**Проектирование программносй системы - 2 недели**

**Реализация - 2 недели**

**Тестирование - 1 неделя**

**3.3. Ведение разработки**

Члены команды будут взаимодействовать друг с другом через Telegram или при очных встречах. Встречи организуются обоими разработчиками при необходимости.

**4. Суть проекта**

Doodle Jump - это 2D игра, в которой игрок должен подняться как можно выше.

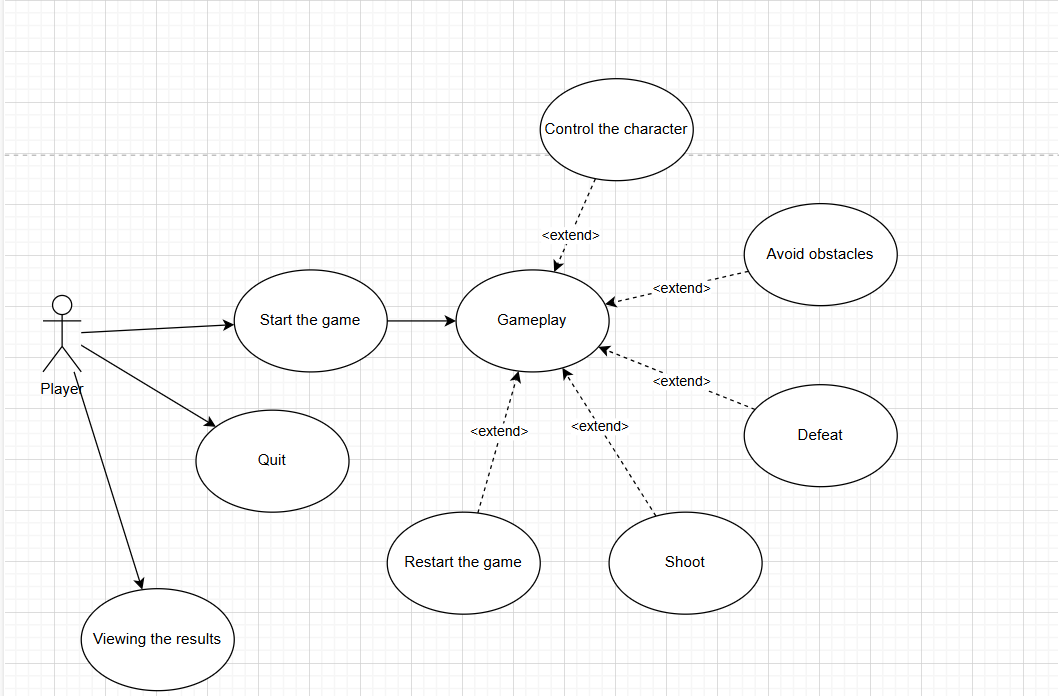
**Все промежутки времени задаются в конфигурации уровня.**

**Платформы:** расположены по всей игре. Они автоматически генерируются во время того, как игрок поднимается всё выше и выше. Есть разные типы платформ, обычная и поломанная. Игрок, прыгая на обычную платформу, подскакивает вверх до следующих платформ. Игрок, прыгая на поломанную платформу, не подпрыгивает вверх, и под ногами она у него ломается.

**Игрок:** Его цель подниматься по платформам как можно выше, и главное не упасть вниз, дабы не проиграть.

**Управление:** Персонаж прыгает автоматически. Игрок должен перемещать персонажа из стороны в стороны с помощью стрелочек Left и Right.

Рисунок 1. Обзор вариантов использования.



1. **План создания проекта**

**Doodle Jump**

**Анализ и сбор требований**

**Тестирование**

**Проектирование**

**Реализация**

Описание продукта

Создание персонажа

Поток задач пользователя

Модульное тестирование

Описание команды

Создание обьектов

Описание модели разработки

Создание Use-Case диаграммы

Ручное тестирование

Разработка UI

Изучение правил игры

Выделение основных обьектов

Суть проекта

Оптимизация

Контроль версий

Используемые инструменты

Функциональные требования

Нефункциональные требования

К концу должно быть реализовано:

● SDP

● SRS

● Дизайн программной системы (Use-Case диаграммы)

● MVP; core-gameplay; управление

**6. Контроль версий**

Для эффективного, удобного и простого управления разработкой проекта используется система контроля версий Git.

**Ссылка на репозиторий github**: <https://github.com/Inferno-kN/DoodleJump2.git>

**6.1. Правила ведения репозитория**

В репозитории имеется одна главная ветка main:

main - в нее вливается ветка dev, когда в ней содержится готовая версия

продукта или обновления для него без каких либо багов, ошибок,

недоработок и т.п.

**6.2. Оформление коммитов**

**Коммиты пишутся следующим образом:**

Тип коммита: описание

**Типы коммитов**:

● **feat:** добавление нового функционала

● **fix:** исправление ошибок

● **refactor:** Правки кода без исправления ошибок или

добавления новых функций

● **delete:** удаление ненужного/лишнего функционала

● **test:** добавление/изменение тестового функционала

● **comment:** комментирование кода

**6.3.**

**Примеры коммитов**

● feat(project): add key[RIGHT] movement

● delete(project): delete key[LEFT] movement

**7. Используемые инструменты**

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **Инструменты** | | |  | | --- | | Назначения | |
| |  | | --- | | PyCharm Community | | Программа, используемая для написания кода на языке Python |
| |  | | --- | | Draw.io | | |  | | --- | | Сайт для создания диаграмм | |
| |  | | --- | | Git | | |  | | --- | | Система контроля версий | |
| |  | | --- | | Github | | Сайт для ведения репозитория |