**Министерство науки и высшего образования Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**(Университет ИТМО)**

Факультет **Прикладной информатики**

Направление подготовки **45.03.04 Интеллектуальные системы в гуманитарной сфере**

Образовательная программа[**Языковые модели и искусственный интеллект**](https://isu.ifmo.ru/pls/apex/f?p=2143:13:116963507483892::NO::EP_UCHEB_YEAR,EP_MEGAFACULTY,EP_FACULTY,EP_CATHEDRA,EP_GROUP,EP_FO,EP_KVAL,EP_UCHEB_PLAN:2024/2025,-1,725,-1,K3160,-1,-1,117277)

**К У Р С О В О Й   П Р О Е К Т**

Тема: «Разработка 2D игры на Python с использованием Pygame»

Обучающийся: Чебукова Александра Александровна, группа K3161

Санкт-Петербург 2024

**СОДЕРЖАНИЕ**

[СОДЕРЖАНИЕ 2](#_heading=h.gjdgxs)

[ВВЕДЕНИЕ 3](#_heading=h.30j0zll)

[1 Суть проекта 4](#_heading=h.1fob9te)

[2 Общий ход работы 6](#_heading=h.3znysh7)

[2.1 Подготовительный этап работы 6](#_heading=h.2et92p0)

[2.2 Разработка игры 6](#_heading=h.tyjcwt)

[2.2.1 Подготовка к работе с кодом 6](#_heading=h.3dy6vkm)

[2.2.2 Начало работы 7](#_heading=h.1t3h5sf)

[2.2.3 Основные классы объектов 7](#_heading=h.4d34og8)

[2.2.4 Разработка уровней 7](#_heading=h.2s8eyo1)

[2.2.5 Основные игровые переменные 8](#_heading=h.17dp8vu)

[2.2.6 Разработка интерфейса 8](#_heading=h.3rdcrjn)

[2.2.7 Игровой цикл 8](#_heading=h.26in1rg)

[2.2.8 Финальный этап разработки 9](#_heading=h.lnxbz9)

[3 Поставленные передо мной задачи и их решение 10](#_heading=h.35nkun2)

[3.1 Поставленные задачи 10](#_heading=h.1ksv4uv)

[3.2 Ход работы 10](#_heading=h.44sinio)

[3.2.1 Начало работы 10](#_heading=h.2jxsxqh)

[3.2.2. Методы класса Soldier 10](#_heading=h.z337ya)

[3.2.3. Методы классов Bullet, Grenade и ItemBox 11](#_heading=h.3j2qqm3)

[3.2.5 Класс World и редактор уровней 12](#_heading=h.1y810tw)

[3.2.6 Игровой цикл 12](#_heading=h.4i7ojhp)

[4 Анализ работы 13](#_heading=h.2xcytpi)

[5 Взаимодействие с командой 14](#_heading=h.1ci93xb)

[6 Взаимодействие с руководителем проекта и оценка его работы 15](#_heading=h.3whwml4)

[ЗАКЛЮЧЕНИЕ 17](#_heading=h.2bn6wsx)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_heading=h.qsh70q)

[ПРИЛОЖЕНИЕ 19](#_heading=h.3as4poj)

**ВВЕДЕНИЕ**

Индустрия развлечений была и остаётся востребованной, а компьютерные игры, как часть её, являются одним из самых популярных видов развлекательного контента и досуга. К тому же это отличный инструмент для развития когнитивных и социальных навыков, зрительно-моторной координации и быстрой реакции, который давно играет немаловажную роль в современном обществе. Наш проект является универсальным и демократичным способом реализовать потребность в отдыхе и эмоциональной разгрузке широкому кругу пользователей разных возрастов, которые интересуются тематикой фэнтези.

Цель проекта: разработать прототип 2D игры на языке программирования Python с использованием фреймворка PyGame.

Задачи:

1. выбрать направление и стилистику игры,
2. определить базовую архитектуру игры,
3. создать дизайн-документ,
4. реализовать графическую составляющую игры в Figma,
5. определить звуковое сопровождение,
6. подготовить и настроить проект в IDE,
7. создать репозиторий проекта на GitHub,
8. разработать графику и анимации,
9. реализовать физику и коллизии,
10. реализовать основные игровые механики,
11. реализовать обработку пользовательского ввода,
12. реализовать интерфейс управления,
13. добавить в игру звуковое сопровождение,
14. провести пользовательское тестирование.

**1 Суть проекта**

Игра представляет собой классический экшн-платформер в тематике средневекового фэнтези. Главный герой - легендарный вор, а по совместительству смелый искатель приключений и сорвиголова, владеющий ключом от всех дверей - решает отправиться в руины древнего храма давным-давно забытого бога смерти в поисках артефакта, дарующего бессмертие. Его путь преграждают полчища неупокоенной нежити, смертельные ловушки и прочие опасности, отчего нашему вору придется с умом использовать свои навыки и окружение для того, чтобы суметь дойти до конца и выбраться оттуда живым.

Передвижение героя осуществляется стрелками влево-вправо или клавишами «A» и «D», прыжок – стрелкой вверх или клавишей «W», стрельба – пробелом. Также есть возможность бросить взрывающуюся колбу, нажав клавишу «Q» или «G».

В игре есть враги, которые ходят по заданной траектории и, если игрок попадает в зону их видимости, начинают атаковать стрельбой из лука. Также есть враги, которые не атакуют, но приносят урон, если подойти к ним вплотную.

В игре есть система очков и жизней. Очки можно получить за уничтожение врага, их подсчёт ведётся в реальном времени и отражается в статистике по завершении игры. Урон от врагов отнимает определённое количество жизней, попадание в лаву или падение в пропасть отнимает все жизни сразу. Если жизни закончились, то герой погибает, игра заканчивается и начинается заново при перезапуске.

Игра содержит 5 уровней с нарастающей сложностью. Переход на следующий уровень происходит, когда игрок добирается до таблички «выход». Если все уровни пройдены, пользователь видит поздравление с победой и статистику игры, и может попробовать пройти игру ещё раз. Изменить расположение предметов, препятствий и врагов на уровне, а также стартовую точку игрока можно при помощи редактора уровней.

В игре используется 2D-графика, что позволяет сохранить стиль и оптимально использовать ресурсы компьютера, к тому же она проста и удобна в реализации. Графика включает в себя спрайты, анимацию, фон, физику и коллизии. Фоновая музыка представляет собой нейтральную мелодию, задающую нужную атмосферу и настраивающую на боевой лад. Каждое действие в игре сопровождается звуками: например, прыжок, стрельба, урон, ходьба, взрыв колбы и т.д.

**2 Общий ход работы**

**2.1 Подготовительный этап работы**

В первую очередь мы распределили роли в команде и утвердили техническое задание, а также определились с жанром, тематикой, основной идеей игры. Кабиров Марат взял на себя роль геймдизайнера, в его задачи входило определение жанра и тематики игры, игровых механик и создание дизайн-документа – основного документа игры, в котором содержится подробное описание уровней, персонажей и интерфейса игры. Величко Артём занимался дизайном изображений для персонажей, анимаций, фонов, объектов и кнопок, проектированием интерфейса. Также он создал музыкальное сопровождение к игре и подобрал необходимые звуки.

Непосредственно разработкой занимались Суханова Мария, Даутова Арина и я, Чебукова Александра. Мария выполняла роль тимлида: следила за сроками и качеством разработки, уточняла требования и была ответственна за разработку в целом. Также она организовала работу в удаленном репозитории на GitHub, добавляла спрайты персонажей, фонов и объектов, создавала эффекты и анимации. Арина отвечала за пользовательский ввод (управление) и реализацию интерфейса игры: добавляла счётчик очков, подсказки, меню, звуки и фоновую музыку. Я реализовывала физику и игровые механики, подробнее о моих задачах будет рассказано в следующем разделе.

После утверждения технического задания и создания дизайн-документа мы приступили к созданию игры.

**2.2 Разработка игры**

**2.2.1 Подготовка к работе с кодом**

Код проекта хранился в удаленном репозитории на GitHub. Каждый участник команды вел разработку в собственной ветке, и по завершении работы над отдельной частью проекта сливал результат в основную ветку, таким образом разработка отдельных функциональных частей игры велась параллельно. Идеи для реализации мы черпали из видео-туториалов [1][2][3][4], а также изучали код проектов, похожих на наш, находящихся в открытом доступе на GitHub. Это помогло нам составить архитектуру игры и реализовать отдельные функции. Также помогала официальная документация Python и электронные ресурсы по PyGame [5].

**2.2.2 Начало работы**

В первую очередь были установлены, импортированы и инициализированы необходимые библиотеки и модули: pygame, mixer, os, random, csv, button. Далее создано окно игры, определён счетчик времени и заданы параметры FPS.

**2.2.3 Основные классы объектов**

Так как Python является объектно-ориентированным языком, каждый объект в игре представляет собой класс, имеющий свойства и методы. Например, для всех персонажей игры был создан класс Soldier, так как они имеют схожие свойства, например, изображение, связанный объект типа Rect, его размеры и расположение на экране, количество снарядов и жизней, и схожие методы: перемещение, стрельба. Этот класс является наследником встроенного класса pygame.sprite.Sprite, созданного специально для реализации спрайтов в играх. Для игровых предметов и элементов интерфейса также созданы свои классы: Decoration, Water, Bullet, Exit, ItemBox, HealthBar и другие, определяющие свойства и методы каждого из объектов. Большинство из них тоже наследуют свойства и методы класса Sprite.

**2.2.4 Разработка уровней**

Каждый уровень мы представили набором плиток, каждая из которых соответствует определенной ячейке на экране. Существует 22 вида плиток, распределённых на группы: препятствия, лава, декорации, игрок, 2 группы врагов, 3 группы ящиков с припасами и табличка выхода. Для разработки уровней был создан редактор уровней, позволяющий расположить плитку любого типа в любую ячейку с помощью удобного графического интерфейса.

**2.2.5 Основные игровые переменные**

В процессе написания классов и методов игры вне классов были инициализированы некоторые переменные, определяющие настройки игры, такие как гравитация, количество игровых плиток, размер плитки, количество типов плиток, количество уровней, текущий уровень, прокрутка игрового экрана и прокрутка фона. Также определены булевы переменные, отвечающие за события в игре и действия игрока: начало проигрывания интро, начало игры, движение игрока вправо и влево, прыжок, стрельба, бросок колбы.

**2.2.6 Разработка интерфейса**

Для интерфейса были определены цвета, для текста был выбран шрифт, также были написаны функции, позволяющие выводить на экран текст и фон. Когда был готов дизайн, в игру были загружены и установлены фоны, картинки персонажей и объектов, реализованы анимации. Кнопки и затемнения в игре были реализованы с помощью специальных классов Button и ScreenFade соответственно и инициализированы в файле игры как элементы этих классов.

**2.2.7 Игровой цикл**

Параллельно с разработкой необходимых функций создавался игровой цикл. До начала игры выводится меню и кнопки, на которые нужно нажать, чтобы начать игру или выйти из игры. После начала игры реализуется игровой цикл, в котором на каждой итерации на экран выводятся изображения и анимации фона, игрового мира (загрузка информации о первом уровне и создание игрового мира на её основе происходит до начала игрового цикла), элементов интерфейса, предметов, персонажей и самого игрока. Также обновляются состояния и действия персонажей и объектов, реализованные в специальных функциях. При смерти игрока открывается соответствующий экран с кнопкой перезапуска, при нажатии на которую реализуется функция перезапуска уровня, а список с информацией об игровом мире очищается, и игровой мир создаётся заново. Отдельный блок составляет код, реализующий обновление действий игрока и обработку пользовательского ввода с клавиатуры.

**2.2.8 Финальный этап разработки**

На финальном этапе разработки в игру были добавлены звуки и музыка. Также было проведено ручное тестирование игры, найдены и устранены некоторые баги.

**3 Поставленные передо мной задачи и их решение**

**3.1 Поставленные задачи**

Мне нужно было реализовать физику персонажей: движение, прыжки, гравитация, коллизии, а также игровые механики: подсчет очков, жизней, стрельба, бросание колб, смерть, перезапуск уровня и переход на следующий уровень.

**3.2 Ход работы**

**3.2.1 Начало работы**

Работу я начала с просмотра видео-туториалов [1][2][3][4], рекомендованных нашим руководителем Качайло Валерием. Таким образом я ознакомилась с основными объектами и методами библиотеки PyGame, узнала, какие стандартные приёмы в создании 2D-игр существуют. Далее, изучив несколько примеров игр, подобных нашей, углубилась в реализацию непосредственно моей задачи.

**3.2.2. Методы класса Soldier**

Я реализовывала функции, отвечающие за перемещение персонажей, обработку коллизий персонажей между собой, а также с различными препятствиями и предметами. Для класса персонажей Soldier это был метод move(), где прописаны и реализованы сдвиги спрайта по горизонтальной и вертикальной осям, физика прыжка, сымитирована гравитация, обработаны коллизии с препятствиями для игрока и врагов: в случае столкновения игрока со стеной он не может пройти дальше в сторону стены, а враг будет разворачиваться и идти в другую сторону на заданное количество плиток. Также обработаны следующие коллизии:

- при столкновении с лавой или падении в пропасть герой теряет все жизни,

- при столкновении с выходом герой переходит на следующий уровень,

- игрок не может выйти за рамки экрана, при этом прокрутка экрана меняется в зависимости от положения игрока.

Для двух типов врагов в классе Soldier отдельно созданы методы ai() и ai2(), определяющие их поведение. Оба типа двигаются по заданной траектории, то есть проходят некоторое количество плиток, разворачиваются и идут обратно. Кроме того, в случайный момент они могут остановиться, а затем продолжить перемещение. Первый тип врагов разворачивается к игроку и производит атаку, когда игрок находится в зоне видимости. Второй же тип не производит атаку, но при столкновении с игроком наносит урон. Расположение обоих типов персонажей учитывает экранный сдвиг в соответствии с расположением игрока.

Метод check\_alive() проверяет количество здоровья у персонажа. Если оно становится меньше или равно нулю, то оно автоматически становится равным нулю, и персонаж умирает.

**3.2.3. Методы классов Bullet, Grenade и ItemBox**

В методе update() класса Bullet было реализовано движение снаряда (пули или стрелы), проверка попадания за границы экрана, проверка коллизий:

- при коллизии с препятствием снаряд исчезает,

- при коллизии с игроком снаряд тоже исчезает, при этом у игрока отнимается некоторое количество жизней,

- при коллизии с врагом снаряд также исчезает, при этом у врага отнимается некоторое количество жизней, а игроку начисляются очки.

В методе update() класса Grenade реализована физика броска колбы, а также проверка коллизий с препятствиями и персонажами. При столкновении с препятствием колба взрывается (объект класса Grenade исчезает, появляется объект класса Explosion), полученный взрыв наносит урон всем персонажам, находящимся рядом.

В методе update() класса ItemBox проверяется коллизия игрока с ящиками в игре. Если происходит коллизия, то в зависимости от типа ящика пополняется либо здоровье, либо боеприпасы, либо колбы.

**3.2.5 Класс World и редактор уровней**

Информация о типе каждой плитки на экране содержится в специальном файле, представляющем собой список чисел от 0 до 21. Каждое число соответствует одной плитке и обозначает её тип. Класс World необходим для обработки информации о каждом уровне, это осуществляется с помощью метода process\_data(). В зависимости от типа плитки для каждой ячейки создаётся свой объект определенного класса и помещается в соответствующую группу. Для перезагрузки уровня была создана функция отдельная функция reset\_level(), удаляющая все объекты из всех групп.

Редактор уровней запускается при необходимости в отдельном файле. Информация о созданном уровне преобразовывается в нужный формат и попадает в соответствующий файл.

**3.2.6 Игровой цикл**

В игровом цикле мной реализован функционал обновления действий персонажей и состояния предметов с помощью функций update(), ai() и ai2(). Также мной реализован функционал загрузки уровней, перехода на следующий уровень и перезагрузки в случае, если игрок потерял все жизни.

**4 Анализ работы**

В процессе работы возникало довольно много трудностей, особенно в самом начале. Например, не сразу получилось корректно обрабатывать коллизии, так как в PyGame для этого есть несколько способов, у каждого из которых есть свои нюансы. Также возникли сложности на этапе применения гравитации.

Не хватало систематических знаний и опыта, приходилось усваивать большой объем новой информации в короткие сроки. Однако по мере изучения теоретических материалов и получения практического опыта я осваивалась, становилось всё проще и проще вникать в новые понятия и решать поставленные задачи.

Не всегда получалось работать планомерно, так как было сложно распределить время и внимание на несколько учебных предметов и задач. Порой работа над проектом откладывалась, но в итоге получилось закончить все поставленные передо мной задачи в срок.

За время выполнения курсового проекта я освоила новую для меня библиотеку PyGame, глубже изучила концепции объектно-ориентированного программирования, научилась работать с Git и GitHub, получила навыки работы в команде и опыт презентации проектов онлайн.

**5 Взаимодействие с командой**

Взаимодействие с командой происходило в общем чате в Telegram, где состояли все участники проекта и руководитель. В нём происходило распределение ролей в команде, обсуждение и утверждение технического задания, общей концепции игры, там же участники обсуждали промежуточные итоги и отчитывались о проделанной работе.

Также существовал отдельный чат разработчиков вместе с руководителем проекта. Это было удобно, так как вопросы, касающиеся разработки, не терялись в общем обсуждении и не мешали остальным участникам.

Взаимодействие в команде было комфортным, все вопросы мы старались решать оперативно, конфликтов не возникало. Каждый участник ответственно подходил к выполнению своих задач, но если возникали трудности, старались помогать друг другу.

**6 Взаимодействие с руководителем проекта и оценка его работы**

Основное взаимодействие с руководителем проекта происходило в общих чатах, а также в личных сообщениях в Telegram, было несколько встреч в Zoom. Также на этапе выбора проектов и распределения по командам мы взаимодействовали через платформу “ITMO Learning Projects”, там же было размещено техническое задание.

В первую очередь необходимо было распределить роли в команде и утвердить техническое задание. При распределении ролей пожелания и резюме участников руководителем были учтены, был предложен вариант, с которым участники проекта согласились. Перед утверждением технического задания руководитель уточнил, все ли формулировки понятны и устраивает ли нас распределение задач. После уточнения деталей ТЗ было утверждено.

После утверждения технического задания Валерий наметил поэтапный ход действий и предоставил каждому участнику ресурсы, к которым можно обратиться при выполнении поставленных задач. В дальнейшем Валерий отвечал на возникающие вопросы, помогал с выполнением задач и следил за сроками выполнения, при необходимости их корректировал.

Перед началом работы был проведена онлайн-встреча с командой разработчиков, где ещё раз были обговорены задачи и сроки, а также была возможность задать вопросы. После завершения работы над проектом была проведена общая встреча команды в Zoom для репетиции защиты.

Работу руководителя я оцениваю высоко, так как было дано четкое техническое задание, предоставлены качественные информационные ресурсы, на протяжении всего периода работы над проектом Валерий был открыт к общению, оперативно отвечал на вопросы, давал ценные советы и оказывал необходимую помощь. Отношение к участникам проекта было лояльным и внимательным, управление гибким и эффективным. Возможно, стоило более жёстко контролировать сроки выполнения, чтобы работа шла более планомерно, но это не помешало выполнить все задачи и достичь всех поставленных целей.

**ЗАКЛЮЧЕНИЕ**

В процессе работы над проектом некоторые идеи были пересмотрены, упрощены, от какого-то намеченного функционала пришлось отказаться в силу ограниченного времени и опыта участников проекта. Однако минимальные выдвинутые требования соблюдены, цель проекта достигнута, поставленные задачи выполнены: мы разработали функционирующий прототип 2D игры на языке программирования Python с использованием фреймворка PyGame. Мой вклад в проект соизмерим с вкладом остальных участников, я полноценно участвовала в разработке, выполняла все необходимые задачи и по мере сил участвовала в организации процесса.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Holzer, R. Pygame tutorial Documentation Release 2019 [Электронный ресурс]. — 2021. — URL: <https://pygame.readthedocs.io/_/downloads/en/latest/pdf/> (дата обращения: 20.11.2024)
2. PyGame - Platformer [Электронный ресурс]. — 2020. — URL: <https://www.youtube.com/playlist?list=PLjcN1EyupaQnHM1I9SmiXfbT6aG4ezUvu> (дата обращения: 19.11.2024)
3. PyGame - Level Editor [Электронный ресурс]. — 2021. — URL: <https://www.youtube.com/playlist?list=PLjcN1EyupaQn-x_9QuGIwIESxfV9kNUZf> (дата обращения: 25.11.2024)
4. Уроки по PyGame [Электронный ресурс]. — 2020. — URL: <https://www.youtube.com/playlist?list=PLA0M1Bcd0w8xg_hyqpPpHdbZnPubSyIQ_> (дата обращения: 20.11.2024)
5. PyGame Tutorials [Электронный ресурс]. — 2020. — URL: <https://www.youtube.com/playlist?list=PL30AETbxgR-feEfqwQxZ-_8s0fcvMQgqJ> (дата обращения: 18.11.2024)

**ПРИЛОЖЕНИЕ**

**Список сокращений и условных обозначений**

Спрайт – элемент компьютерной графики, представляющий объект на экране, который может двигаться. В двухмерной игре все, что вы видите на экране, является спрайтами. Спрайты можно анимировать, заставлять их взаимодействовать между собой или передавать управление ими игроку.

Игровой цикл – основной механизм управления игрой, обеспечивающий обновление состояния игры, обработку пользовательского ввода и рендеринг графики. Цикл включает этапы обработки событий, обновления логики и отрисовки кадра на экране.

Коллизия – событие, возникающее при пересечении объектов в игровом пространстве. В играх коллизии обрабатываются специальными алгоритмами для корректного отображения взаимодействий между персонажами, объектами и окружением.

Платформер – жанр видеоигр, где основным элементом геймплея является управление персонажем, который перемещается по платформам, преодолевает препятствия и взаимодействует с игровыми элементами (например, прыжки, сбор предметов).

Рендеринг – процесс отрисовки графики игры на экране. Включает отображение объектов, фонов и интерфейса в соответствии с текущим состоянием игрового мира.

Дизайн-документ – документ, в котором содержится вся информация о создаваемой игре. Он нужен, чтобы команда, которая будет работать над проектом, понимала, что в итоге должно получиться.

**Техническое задание**

Требования:

- игра должна отображать 2D-графику с использованием спрайтов,

- анимации персонажей и объектов должны переключаться плавно и корректно в зависимости от действий (ходьба, прыжок, атака),

- должна быть реализована поддержка задних фонов на уровнях игры,

- должна быть реализована базовая система физики, включающая гравитацию, коллизии и прыжки,

- коллизии объектов должны корректно обрабатываться, обеспечивая реалистичное взаимодействие между персонажем и игровым окружением,

- должна поддерживаться обработка ввода с клавиатуры (перемещение персонажа, атаки и взаимодействие с объектами),

- игра должна содержать несколько уровней (от 5) с нарастающей сложностью,

- подсчет очков должен вестись в реальном времени и отображаться на экране,

- игра должна иметь фоновую музыку, которая воспроизводится в каждом уровне,

- звуковые эффекты должны сопровождать действия игрока: атака, прыжки, удары, сбор предметов, смерть персонажа,

- время загрузки уровней не должно превышать 5 секунд,

- игра должна корректно работать на ОС Windows,

- интерфейс должен быть интуитивно понятным: управление соответствует ожиданиям пользователя и не требует глубокого изучения,

- в случае ошибки игра должна корректно завершаться или возвращаться в главное меню, без критических сбоев.

Цель проекта - разработать прототип 2D игры на языке программирования Python с использованием фреймворка PyGame.

Задачи:

– сформулировать и представить требования. Исполнитель: Валерий Качайло, сроки: 01.11.2024-14.11.2024,

– выбрать направление и стилистику игры. Исполнитель: Марат Кабиров, сроки: 11.11.2024-14.11.2024,

– определить базовую архитектуру игры. Исполнитель: Марат Кабиров, сроки: 18.11.2024-24.11.2024,

– создать дизайн-документ. Исполнитель: Марат Кабиров, сроки: 18.11.2024-27.11.2024,

– определить звуковое сопровождение. Исполнитель: Артём Величко, сроки: 18.11.2024-01.12.2024,

– реализовать графическую составляющую игры в Figma. Исполнитель: Артём Величко, сроки: 18.11.2024-01.12.2024,

– подготовить и настроить проект в IDE. Исполнитель: Мария Суханова, сроки: 11.11.2024-19.11.2024,

– создать репозиторий проекта на GitHub. Исполнитель: Мария Суханова, сроки: 18.11.2024-16.12.2024,

– разработать графику и анимации. Исполнитель: Мария Суханова, сроки: 18.11.2024-16.12.2024,

– реализовать обработку пользовательского ввода. Исполнитель: Арина Даутова, сроки: 18.11.2024-16.12.2024,

– реализовать физику и коллизии. Исполнитель: Александра Чебукова, сроки: 18.11.2024-08.12.2024,

– реализовать основные игровые механики. Исполнитель: Александра Чебукова, сроки: 18.11.2024-08.12.2024,

– добавить в игру звуковое сопровождение. Исполнитель: Арина Даутова, сроки: 18.11.2024-08.12.2024,

– реализовать интерфейс управления. Исполнитель: Арина Даутова, сроки: 18.11.2024-08.12.2024,

– провести пользовательское тестирование. Исполнитель: Мария Суханова, сроки: 08.12.2024-17.12.2024.