Учреждение образования
«Гродненский государственный политехнический колледж»

ОТЧЁТ
ПО УЧЕБНОЙ ПРАКТИКЕ ПО ПРОГРАММИРОВАНИЮ

Учащегося 3 курса, группы ПЗТ-40
специальности 2 - 40 01 01 «Программное обеспечение информационных технологий»
Место прохождения практики УО «Гродненский государственный политехнический колледж»

Тема проекта: «Разработка игрового приложения «Путь авантюриста» в жанре GPG»
Ссылка на проект: https://github.com/Ksenija000/KPiJP.git

Выполнил _____ К.В. Савило
(инициалы, фамилия)
Руководитель практики от колледжа _____ В.В. Хомич
(инициалы, фамилия)

2025

Содержание

Введение. Описание структуры предприятия

На данной учебной практике была поставлена задача разработать игровое приложение «Путь авантюриста». Данный программный продукт позволит пользователю получить эстетическое удовольствие, создаст неповторимый игровой опыт, сочетающий приключения, исследование, торговлю и сражения. Игроки получают удовольствие от самого процесса путешествия, открытия новых локаций и достижения успеха.

Создаваемое приложение ориентировано на пользователей старше 12 лет.

Далее приведено краткое описание разделов пояснительной записки.

Первый раздел носит название «Анализ предметной области и формулировка требований к программе». В нем можно ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, функциональные и нефункциональные требования к программному продукту. В подразделе «Инструменты разработки» рассмотрена среда, в которой создается данный программный продукт. Здесь также установлены минимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачей.

В разделе «Проектирование задачи» рассмотрены основные аспекты разработки программного продукта. Здесь можно узнать об организации данных в контексте среды разработки. В данном разделе будут составлены алгоритмы процесса обработки информации.

«Построение программы» – третий раздел отчета по практике, в котором описываются все элементы и объекты, которые использованы при реализации данного приложения. В этом разделе четко описаны компоненты проекта и их структура.

Четвертый раздел – «Тестирование». В нем описано функциональное тестирование данной программы, смоделированы все основные действия пользователя при работе с программой.

В разделе «Применение» будет описано назначение программы, область применения и среда функционирования программного обеспечения.

«Заключение» содержит краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В разделе «Список использованных источников» приведен список используемых при разработке источников.

В приложении А будет приведен листинг программы.

В приложении В будет приведена диаграмма компонентов.

1 Анализ предметной области и формулировка требований к программе

1.1 Исследование предметной области

Наименование задачи – игровое приложение «Путь авантюриста».

Цель разработки данного программного продукта заключается в создании игрового приложения, которое позволит пользователю окунуться в фэнтезийный мир увлекательных приключений, где он в роли мага-авантюриста, сможет выполнять интересные задания и исследовать игровой мир.

Игровое приложение выполнено в жанре RPG в стиле фентези-средневековья.

Назначение: данный программный продукт ориентирован на несколько категорий игроков: любителей аниме и фэнтези, поклонников ролевых игр (RPG), фанатов крафта и торговли, любителей экшена и приключений, креативных игроков. Таким образом, игра будет привлекательна для широкой аудитории, включая как казуальных игроков, так и более опытных любителей RPG, что делает её универсальной и многогранной.

В центре сюжета стоит авантюрист, чей жизненный путь состоит в поиске приключений. Игрок приходит в город, где ему предстоит вступить в гильдию авантюристов «Бесконечный путь», здесь его будут ждать разнообразные задания, от охоты на монстров до расшифровки древних текстов. Игра предлагает игрокам возможность развивать своего персонажа, повышая его ранг в гильдии и открывая новые уровни сложности и уникальные награды. Игрок может взаимодействовать с NPC, покупать и продавать различные предметы, а также украшать свой собственный дом.

Задача игрока - достичь наивысшего ранга, в гильдии, выполнив все задания и получить звание лучшего авантюриста, а также же наслаждаться увлекательными приключениями в фэнтези мире.

Периодичность использования данного программного продукта неограниченна. Игрок может в любой момент сбросить игровой прогресс, а так же имеет возможность даже после выполнения всех заданий продолжать игру.

Существует множество игр про авантюристов, однако абсолютных аналогов данного программного продукта не было найдено.

1.2 Инструменты разработки

Для разработки данного проекта была выбрана среда Unity, которая является межплатформенной средой для создания и разработки видеоигр и приложений, разработанной компанией Unity Technologies.

Разработка будет производиться на языке программирования C# – многофункциональный язык программирования, активно развивающийся на данный момент. Он часто используется как первый язык программирования начинающих программистов, так и для реализации крупных коммерческих проектов.

Иные инструменты, используемые при разработке и написании сопутствующей документации:

– WEB-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;

– Microsoft Office Word 2013 – для написания документации к программному продукту;

– Smart Install Maker – утилита для создания инсталляторов;

– Dr.Explain – инструмент разработки пользовательской документации;

– GitHub – веб-сервис для хостинга IT-проектов;

– Google Drive – сервис хранения, редактирования и синхронизации файлов, разработанный компанией Google.

При разработке данного программного продукта был использован компьютер со следующими характеристиками:

– процессор AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz;

– ОЗУ: 8Gb;

– память: HDD 512Gb;

– ОС: Windows 10.

2 Проектирование

2.1 Диаграмма вариантов использования

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Суть такой диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

В данной диаграмме вариантов использования актером является игрок.

Игроку доступны следующие варианты использования: играть, читать справку, изменить музыку, выти из игры.

У варианта использования «Играть» наблюдаются следующие «extend»-связи: посмотреть инвентарь, перейти на локации, взаимодействовать с персонажем, купить игровые предметы, продать добытые ресурсы, пройти тест на вступление в гильдию, украсит дом, выбрать задание соответствующее рангу.

У варианта использования «Изменить настройки» наблюдаются следующие «extend»-связи: выбрать мелодию, остановить воспроизведение музыки, продолжить воспроизведение музыки.

У варианта использования «Посмотреть инвентарь» наблюдаются следующие «extend»-связи: перейти к использованию некоторых предметов, посмотреть информацию о предметах.

У варианта использования «Взаимодействовать с прсонажем» наблюдаются следующие «extend»-связи: начать диалог.

У варианта использования «Украсить дом» наблюдаются следующие «include»-связи: разместить предмет интерьера.

У варианта использования «Выбрать задание, соответствующее рангу» наблюдаются следующие «extend»-связи: собрать растения, расшифровать сообщение, собрать кристаллы, создать зелье.

У варианта использования «Перейти к использованию некоторых предметов» наблюдаются следующие «extend»-связи: создать зелье, посмотреть информацию в руководстве начинающего травника, посмотреть информацию в руководстве по зельеварению.

Диаграмма вариантов использования представлена на рисунке 1.

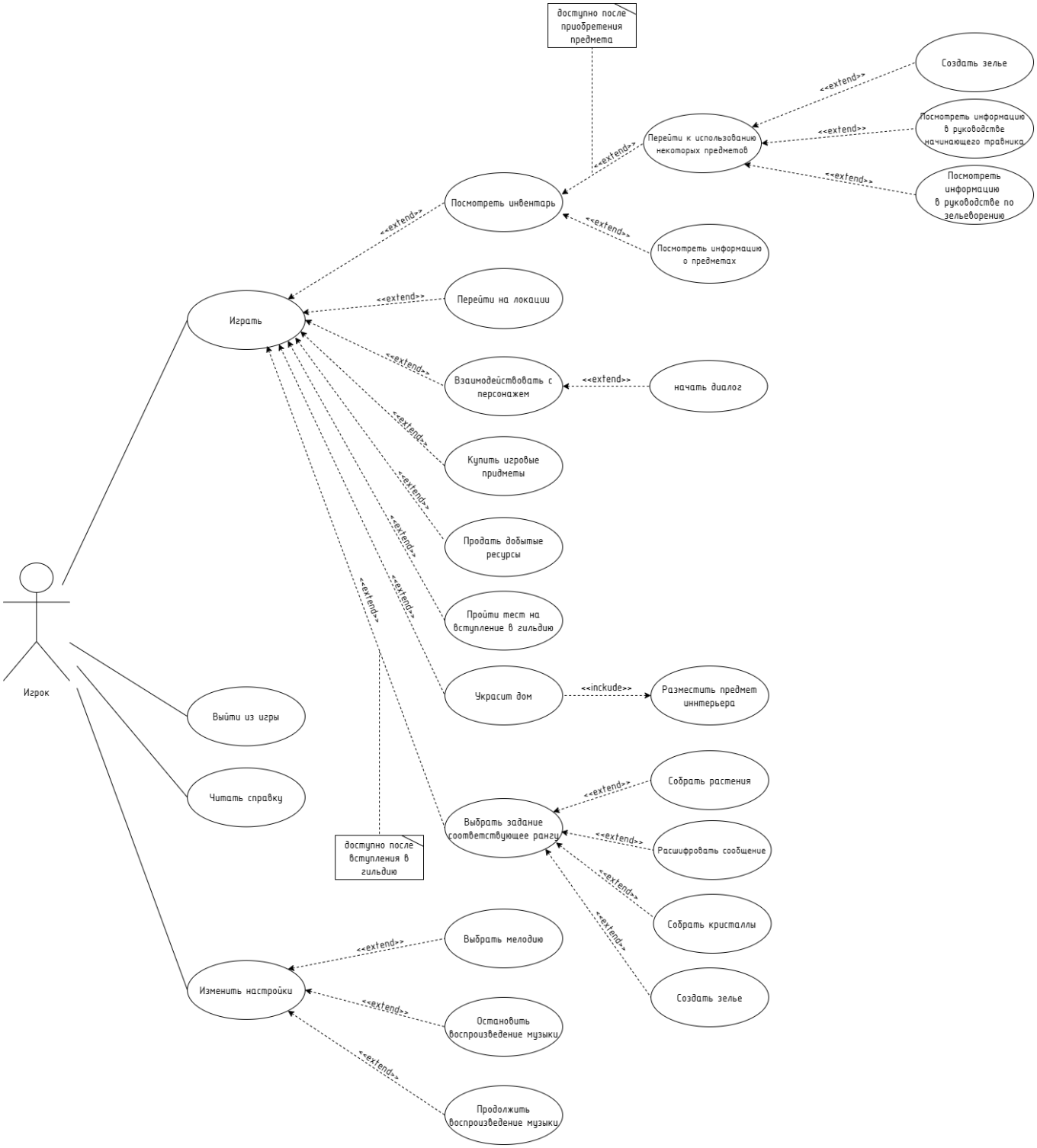

Рисунок 1 – Диаграмма вариантов использования

2.2 Диаграмма деятельности

Диаграмма деятельности – это поведенческая диаграмма, которая показывает поток работы или действий в рамках системы или процесса. Она отображает последовательность шагов и возможные варианты выполнения работы,

включая параллельные процессы и ветвления. Диаграмма деятельности включает в себя такие элементы, как начальные и конечные узлы, узлы действий, узлы решений, вилки и слияния, а также потоки управления, которые связывают эти узлы.

На диаграмме отображен процесс покупки предмета.

Диаграмма деятельности представлена на рисунке 2.

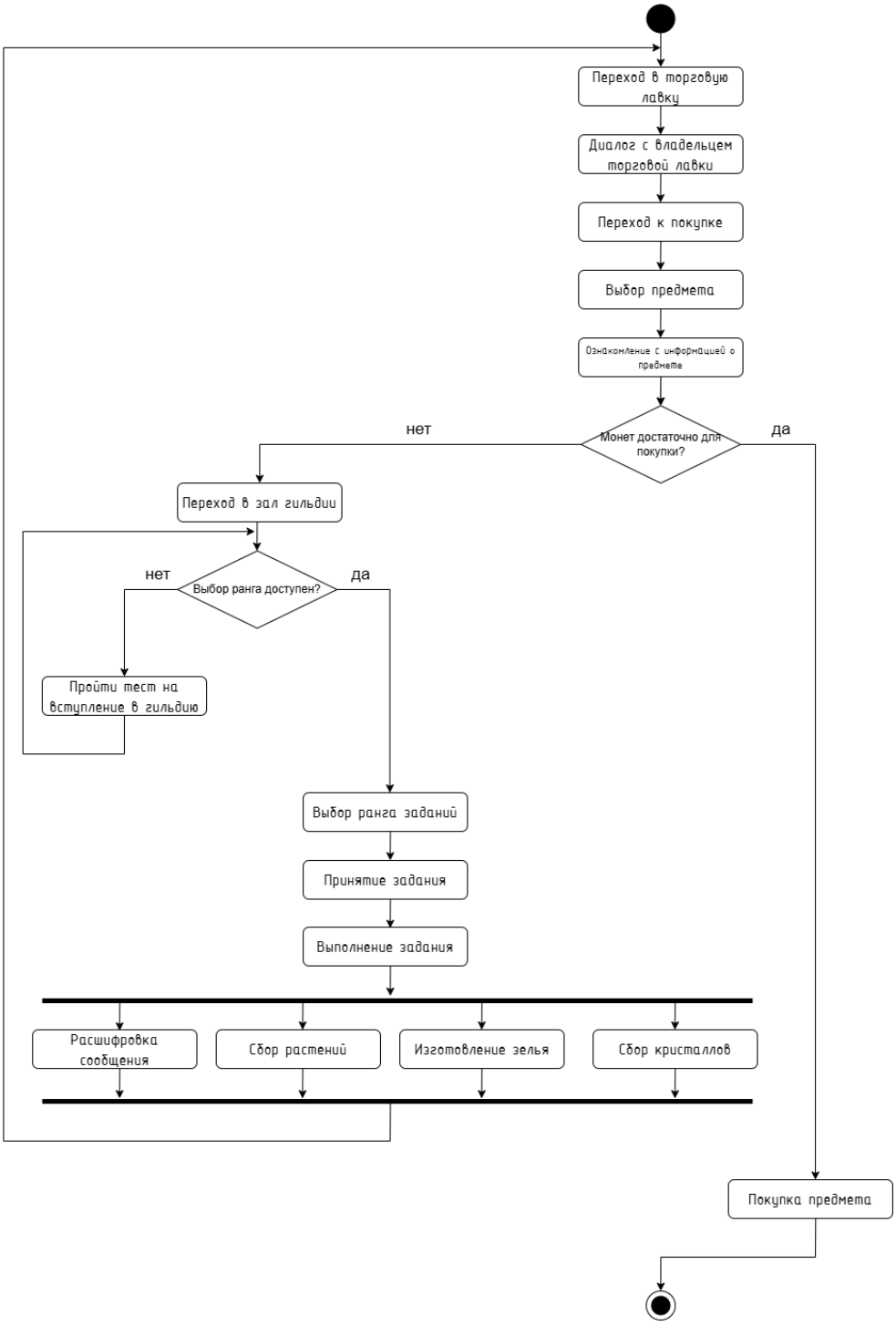

Рисунок 2 – Диаграмма деятельности

2.3 Описание тестов

При разработке приложения необходимо будет провести тест по игровому процессу

Таблица 1 – Тест-кейсы с крайне высоким приоритетом

Заглавие и шаги выполнения	Ожидаемый результат
1	2
Прохождение игрового приложения 1. Запустить игровое приложение «Путь авантюриста». 2. Нажать на кнопку «Справка». 3. Нажать на раздел «Общие сведения». 4. Нажать на подраздел «О программе». 5. Нажать на подраздел «Об авторе». 6. Нажать на раздел «Руководство к применению». 7. Нажать на подраздел «Играть». 8. Нажать на подраздел «Музыка». 9. Нажать на кнопку «Закрыть». 10. Нажать на кнопку «Музыка». 11. Нажать на кнопку «Мелодия 2». 12. Нажать на кнопку «Стоп». 13. Нажать на кнопку «Стоп». 14. На сцене «Музыка» нажать на кнопку «Выход». 15. Нажать на кнопку «Играть». 16. Нажать на кнопку «Далее». 17. Нажать кнопку «В путь». 18. Нажать кнопку «Меню». 19. Нажать кнопку «Закрыть». 20. Нажать на дверь гильдии «Бесконечный путь» 21. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» подойти к администратору гильдии. 22. Нажать кнопку «Диалог» 23. Нажать кнопку «Далее»	1. Игра запускается. Отображается экран загрузки. Загружается главное меню игры с кнопками: «Справка», «Играть», «Музыка» и «Выход из игры». 2. Загружается окно справки С начальной страницей и разделами: «Общие сведения» с подразделами «О программе», «Об авторе», раздел «Руководство к применению» с подразделами «Играть» и «Музыка». 3. Осуществляется переход на страницу «Общие сведения» с базовой информацией. 4. Осуществляется переход на страницу «О программе» с основной информацией о приложении. 5. Осуществляется переход на страницу «Об авторе» с основной информацией о создателях приложения. 6. Осуществляется переход на страницу «Руководство к применению» с информацией по главному экрану. 7. Осуществляется переход на страницу «Играть» с основной информацией о механике игры. 8. Осуществляется переход на страницу «Музыка» с основной информацией о настройке музыки. 9. Закрывается справка. 10. Загружается сцена «Музыка» с кнопками «Мелодия 1», «Мелодия 2», «Мелодия 3», «Мелодия 4», «Стоп», «Выход». 11. Запускается Lindsey Stirling - Roundtable Rival.mp3. 12. Прекращается воспроизведение Lindsey Stirling - Roundtable Rival.mp3. 13. Продолжается воспроизведение Lindsey Stirling - Roundtable Rival.mp3. 14. Загружается главное меню игры с кнопками: «Справка», «Играть», «Музыка» и «Выход из игры». 15. Загружается сцена «Вход в город» с кнопками: «Меню» и изображением предисловия с кнопкой «Далее». 16. Изменяется текст предисловия на изображении, появляются кнопки «Начать с нуля», «В путь». 17. Загружается сцена «Город» с кнопками «Инвентарь» и «Меню». 18. Игра останавливается, отображается меню с кнопками «Справка», «Музыка», «Выход из игры» и «Закрыть». 19. Меню закрывается, игровое время запускается.

Продолжение таблицы 1

24. Нажать кнопку «Тест» 25. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» подойти к врагу. 26. Повторить шаг 25 27. Нажать кнопку «Гильдия» 28. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» подойти к администратору гильдии. 29. Нажать на кнопку «F» на доске заданий 30. Нажать на кнопку «Задание 1» 31. Нажать кнопку «Принять» 32. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» подойти к кнопкам «Радужный колокольчик» и «Снежный зверобой» и нажать на них 33. Нажать на кнопку «Гильдия» 34. Нажать на кнопку «Инвентарь» 35. Нажать на кнопку предмета 36. Нажать на кнопку «выход» 37. Повторить шаг 29 38. Нажать на кнопку «Задание 3» 39. Нажать кнопку «Принять» 40. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» подойти к кнопкам «Золотой вороний глаз» и «Ледяной вороний глаз» и нажать на них 41. Повторить шаг 33 42. Нажать на кнопку «Выход» 43. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S»,	20. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход» и администратором гильдии. 21. Появляется кнопка «Диалог» 22. Отображается изображение окна диалога с репликой, и именем персонажа и кнопкой «Далее». 23. Изменяется текст на изображении окна диалога, появляется кнопка «Тест». 24. Загружается сцена «Арена», с изображением единиц здоровья и врагом, а также кнопкой «Меню». 25. Теряется одна единица здоровья, игрок отталкивается от врага. 26. Повторяется шаг 25, игра останавливается, отображается изображение с сообщением о принятии в гильдию с рангом F и кнопка «Гильдия». 27. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг F. 28. Кнопки «Диалог» и «Тест» не отображаются. 29. Отображаются кнопки «Задание 1», «Задание 2», «Задание 3». 30. Отображается изображение с описанием задания и кнопка «Принять». 31. Загружается сцена «Лес», с кнопками «Меню», «Инвентарь», «Гильдия», и карточками растений «Радужный колокольчик» и «Снежный зверобой» с указанием добытого количества. 32. Значения на карточках задания увеличиваются. Игра останавливается. Отображается изображение сообщения о выполнении награды и получении награды и кнопкой «Гильдия». 33. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг F. 34. Загружается сцена «Инвентарь», отображается количество добытых ресурсов. Количество сонет равно 100, количество остальных предметов 0. 35. Отображается описание предмета 36. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг F. 37. Повторяется шаг 29 38. Отображается изображение с описанием задания и кнопка «Принять». 39. Отображается изображение с описанием задания и кнопка «Принять». 40. Значения на карточках задания увеличиваются. Игра останавливается. Отображается изображение сообщения

Продолжение таблицы 1

»DownArrow» подойти к торговой лавке. 44. Нажать на дверь торговой лавки. 45. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» подойти к владельцу лавки. 46. Нажать кнопку «Диалог». 47. Нажать кнопку «К покупке». 48. Нажать на предмет «Пособие по теории шифрования». 49. Нажать на кнопку «Купить». 50. Нажать на предмет «Стол». 51. Нажать на кнопку «Купить». 52. Нажать на предмет «Стол». 53. Нажать на кнопку «Руководство начинающего травника». 54. Нажать на кнопку «Выход». 55. Повторить шаг 34 56. Нажать на кнопку «Шифр» 57. Нажать на кнопку «Использовать». 58. Нажать на кнопку «Выход». 59. Нажать на кнопку «Руководство начинающего травника» 60. Нажать на кнопку «Использовать». 61. Нажать на кнопку растения 62. Нажать на кнопку «Выход». 63. Нажать на кнопку «Выход». 64. Повторить шаги 45-46. 65. Нажать кнопку «К продаже». 66. Нажать на кнопку добытого ресурса. 67. Ввести в поле «Шт.» добытое количество ресурса. 68. Нажать кнопку «Продать» 69. Ввести в поле «Шт.» значение больше добытого количества ресурса. 70. Нажать на кнопку «Выход». 71. Нажать на кнопку «Выход». 72. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow»,	о выполнении награды и получении награды и кнопкой «Гильдия». 41. Повторяется шаг 33 42. Загружается сцена «Город» с кнопками «Инвентарь» и «Меню». Отображается текущий ранг. 43. Игровой персонаж переместился к торговой лавке. 44. Загружается сцена «Торговая лавка» с владельцем лавки, кнопками «Меню» и «Инвентарь», отображается текущий ранг. 45. Появляется кнопка «Диалог» 46. Отображается изображение окна диалога и кнопки «К покупке», «К продаже». 47. Загружается сцена «Торговля», отображается интерфейс покупки из торговой лавки. Отображается количество добытых монет и кнопка «Выход». 48. Отображается информация о предмете и иго цена 49. Количество добытых монет уменьшается на цену предмета. В инвентаре значение данного предмета увеличивается на 1. 50. Отображается информация о предмете и иго цена 51. Количество добытых монет уменьшается на цену предмета. В инвентаре значение данного предмета увеличивается на 1. 52. Отображается информация о предмете и иго цена 53. Количество добытых монет уменьшается на цену предмета. В инвентаре значение данного предмета увеличивается на 1. 54. Загружается сцена «Торговая лавка» с владельцем лавки, кнопками «Меню» и «Инвентарь», отображается текущий ранг. 55. Загружается сцена «Инвентарь», отображается количество добытых ресурсов. 56. Отображается информация о предмете и кнопка «Использовать». 57. Загружается сцена книга. Отображается изображение с информацией о методе шифрования и кнопка «Выход». 58. Загружается сцена «Инвентарь», отображается количество добытых ресурсов. 59. Отображается информация о предмете и кнопка «Использовать». 60. Загружается сцена книга. Отображается книга с кнопками растений и кнопка «Выход». 61. Отображается информация о растении. 62. Загружается сцена «Инвентарь», отображается количество добытых ресурсов. 63. Загружается сцена «Торговля», отображается интерфейс продажи из торговой лавки. Отображается количество добытых монет и количество добытых ресурсов и кнопка «Выход». 64. Повторяются шаги 45-46.

Продолжение таблицы 1

»W», «Space», «S», «DownArrow» подойти к дому. 73. Нажать на дверь дома. 74. Нажать на кнопку «Плюс» 75. Нажать на кнопку «Стол» 76. Нажать на кнопку «Плюс» 77. Нажать на клавишу «Плюс», наведя курсор на украшение. 78. Нажать на клавишу «Минус», наведя курсор на украшение. 79. Нажать на клавишу «Tab», наведя курсор на украшение. 80. Нажать на клавишу «Q», наведя курсор на украшение. 81. Зажав левую клавишу мыши, наведя курсор на украшение потянуть мышь. 82. Нажать кнопку «Выход» 83. Повторить шаги 72-73. 84. Нажать на клавишу «Backspace». 85. Повторить шаг 82. 86. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» подойти к магазину магии. 87. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» подойти к владельцу магазина. 88. Нажать кнопку «Диалог». 89. Нажать кнопку «к покупке» 90. Нажать на кнопку «Карта» 91. Нажать на кнопку «Купить» 92. Нажать на кнопку «Котелок» 93. Нажать на кнопку «Купить» 94. Нажать на кнопку «Руководство по зелье варению» 95. Нажать на кнопку «Купить» 96. Нажать на кнопку «Выход» 97. Нажать на кнопку «Выход» 98. Нажать на дверь гильдии 99. Нажать на кнопку «Инвентарь» 100. Нажать на кнопку «Руководство по зелье варению»	65. Загружается сцена «Торговля», отображается интерфейс продажи из торговой лавки. Отображается количество добытых монет и количество добытых ресурсов и кнопка «Выход». 66. Отображается информация о ресурсе и его цена за единицу. 67. Введённое значение отображается в поле «Шт.» 68. Количество ресурса уменьшается на введённое число. Количество добытых монет увеличивается на произведение цены ресурса и введённого количества. 69. Ничего не изменяется 70. Загружается сцена «Торговая лавка» с владельцем лавки, кнопками «Меню» и «Инвентарь», отображается текущий ранг. 71. Загружается сцена «Город» с кнопками «Инвентарь» и «Меню». Отображается текущий ранг. 72. Игровой персонаж переместился к дому. 73. Загружается сцена «Дом» с кнопками «Инвентарь», «Плюс», «Выход» и «Меню». Отображается текущий ранг. 74. Отображается список украшений с указанием имеющегося количества. 75. Украшение «Стол» появляется на сцене. 76. Список украшений с указанием имеющегося количества скрывается. 77. Украшение увеличивается. 78. Украшение уменьшается. 79. Украшение поворачивается в противоположное направление. 80. Украшение поворачивается по окружности. 81. Украшение перемещается. 82. Загружается сцена «Город» с кнопками «Инвентарь» и «Меню». Отображается текущий ранг. 83. Шаги 72-73 повторяются. Украшение отображается в трансформированном виде. 84. Украшение удаляется со сцены. В инвентаре значение данного украшения увеличивается на 1. 85. Шаг 82 повторяется. 86. Загружается сцена «Магазин магии» с владельцем магазина, кнопками «Меню» и «Инвентарь», отображается текущий ранг. 87. Появляется кнопка «Диалог». 88. Отображается изображение окна диалога и кнопки «К покупке». 89. Загружается сцена «Торговля», отображается интерфейс покупки из магазина магии. Отображается количество добытых монет и количество добытых ресурсов и кнопка «Выход». 90. Отображается информация о предмете и иго цена

Продолжение таблицы 1

101. Нажать на кнопку «Использовать» 102. Нажать на кнопку зелья 103. Нажать на кнопку «Выход» 104. Нажать на кнопку «Котелок» 105. Нажать на кнопку «Использовать» 106. Нажать на кнопку ингредиента. 107. Расположить ингредиенты с помощью мыши слоты котла в соответствии с рецептом. 108. Нажать на кнопку «Создать». 109. Нажать на кнопку зелья. 110. Нажать на кнопку «Выход». 111. Повторить шаг 29. 112. Нажать кнопку «Задание 2». 113. Нажать кнопку «Принять». 114. Ввести в поле ввода ответа значение «аааа». 115. Нажать на кнопку «Готово» 116. Нажать на кнопку «Заново» 117. Ввести в поле ввода ответа 1 значение «Великие начинания даже не надо обдумывать, надо взяться да дело, иначе, заметив трудность отступишь. «. 118. Ввести в поле ввода ответа 1 значение «- Гай Юлий цезарь. «. 119. Нажать на кнопку «Готово». 120. Нажать на кнопку «Гильдия». 121. Нажать на кнопку «Меню». 122. Нажать на кнопку «Карта». 123. Нажать на кнопку «Город». 124. Нажать на дверь гильдии. 125. Нажать на кнопку ранга E на доске заданий. 126. Нажать кнопку «Задание 4» 127. Нажать кнопку «Принять». 128. Повторить шаги 106-109 для требуемого зелья. 129. Нажать на кнопку «гильдия». 130. Повторить шаг 125 131. Нажать кнопку «Задание 5» 132. Повторить шаги 127-129 133. Повторить шаг 125	91. Количество добытых монет уменьшается на цену предмета. В инвентаре значение данного предмета увеличивается на 1. 92. Отображается информация о предмете и иго цена 93. Количество добытых монет уменьшается на цену предмета. В инвентаре значение данного предмета увеличивается на 1. 94. Отображается информация о предмете и иго цена 95. Количество добытых монет уменьшается на цену предмета. В инвентаре значение данного предмета увеличивается на 1. 96. Загружается сцена «Магазин магии» с владельцем магазина, кнопками «Меню» и «Инвентарь», отображается текущий ранг. 97. Загружается сцена «Город» с кнопками «Инвентарь» и «Меню». Отображается текущий ранг. 98. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг F. 99. Загружается сцена «Инвентарь», отображается количество добытых ресурсов. 100. Отображается информация о предмете 101. Загружается сцена книга. Отображается книга с кнопками зелий и кнопка «Выход». 102. Отображается информация о зелье. 103. Загружается сцена «Инвентарь», отображается количество добытых ресурсов. 104. Отображается информация о предмете 105. Загружается сцена «Котелок», отображается количество добытых ингредиентов. 106. Ингредиент появляется на сцене, ото количества добытого ингредиента отнимается 1. 107. Слоты принимают ингредиенты. 108. Отображается кнопка зелья, ингредиенты исчезают. 109. В инвентаре количество зелья увеличивается на 1, кнопка зелья исчезает. 110. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг F. 111. Повторяется шаг 29. 112. Отображается изображение с описанием задания и кнопкой «Принять». 113. Загружается сцена «Шифр» с кнопами «Меню», «Гильдия», полями ввода ответа и кнопкой «Готово» 114. В поле ввода ответа отображается значение «аааа». 115. Игра останавливается. Отображается изображение сообщения «Поражение» с кнопками «Заново» и «Гильдия».

Продолжение таблицы 1

134. Нажать кнопку «Задание 6» 135. Используя для передвижения клавиши «A», «D», «UpArrow», «LeftArrow», «RightArrow», «W», «Space», «S», «DownArrow» избегая врагов коснуться кристалла. 136. Собрать нужное число кристаллов. 137. Нажать кнопку «Гильдия» 138. Нажать кнопку «Продолжить» 139. Нажать кнопку «Выйти» 140. Нажать кнопку «Гильдия» 141. Нажать кнопку «Начать с нуля» 142. Повторить шаги 20-25 143. Нажать левую клавишу мыши.	116. Игра запускается. Поле для ввода ответа отображается пустым. 117. В поле ввода ответа1 отображается значение «Великие начинания даже не надо обдумывать, надо взяться да дело, иначе, заметив трудность отступишь.». 118. В поле ввода ответа 2 отображается значение «- Гай Юлий цезарь. «. 119. Игра останавливается. Отображается изображение сообщения о выполненном задании с указанием зачисленной награды и кнопкой «Гильдия». 120. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг E. 121. Игра останавливается. Отображается изображение меню с кнопками «Справка», «Музыка», «Выход из игры», «Закрыть» и «Карта». 122. Загружается сцена «Карта» с кнопками «Город» и «Лес». 123. Загружается сцена «Город» с кнопками «Инвентарь» и «Меню». Отображается текущий ранг. 124. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг E. 125. Отображаются кнопки заданий ранга «E» 126. Отображается изображение с описанием задания и кнопкой «Принять». 127. Загружается сцена «Котелок», с кнопками ингредиентов и указанием их добытого количества, и карточка задания с указание количества созданных зелий требуемых в задании и кнопкой «Гильдия». 128. Повторяются шаги 106-109 для требуемого зелья. Игра останавливается. Отображается сообщение о выполнении задания и начислении наград с Кнопками «Карта» и «Гильдия». 129. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг E. 130. Повторяется шаг 125. 131. Отображается изображение с описанием задания и кнопкой «Принять». 132. Повторяются шаги 127-129. 133. Повторяется шаг 125. 134. Загружается сцена «Руины», Отображается полоса жизни, текущий ранг, карточка с количеством исцеляющего зелья, кнопка «Инвентарь», кнопка «Меню», и карточка задания с указанием количества собранных кристаллов.

Продолжение таблицы 1

	135. Количество кристаллов увеличивается на карточке задания, кристалл исчезает и появляется в новом месте. 136. Отображается изображение с сообщение о выполнении задания и начислении наград и кнопками «Гильдия» и «Карта». 137. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг E. Игра останавливается. Отображается изображение сообщением о том, что игра пройдена и кнопками «Начать с нуля» и «Продолжить». 138. Игра запускается. 139. Загружается сцена «Город» с кнопками «Инвентарь» и «Меню». Отображается текущий ранг. 140. Загружается сцена «Зал гильдии» с кнопами «Меню», «Инвентарь», «Выход», администратором гильдии. Отображается текущий ранг, на доске заданий доступен ранг E. Игра останавливается. Отображается изображение сообщением о том, что игра пройдена и кнопками «Начать с нуля» и «Продолжить». 141. Игровой прогресс сброшен. 142. Повторяются шаги 20-25. 143. Происходит выстрел, враг теряет единицу жизни.

3 Построение программы

Диаграмма компонентов — это один из видов диаграмм в языке моделирования UML (Unified Modeling Language), который используется для визуализации архитектуры системы. Она отображает компоненты системы и их взаимосвязи, а также показывает, как они взаимодействуют друг с другом.

Диаграмма компонентов представлена прилажении В.

4 Тестирование

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения испытания реализации программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в классах.

Расписание проведения и время, затраченное на тестирование, описано в таблице 2.

Таблица 2 – Расписание работ над проектом

Имя	Дата	Деятельность	Продолжительность, ч
Савило Ксения	29.04.2025	Разработка тестов	2
Савило Ксения	28.04.2025	Тестирование игрового приложения	1
Савило Ксения	04.05.2025	Исправление найденных ошибок	5
Савило Ксения	06.05.2025	Проведение регрессионного тестирования	2
Савило Ксения	08.05.2025	Составление отчета о результатах тестирования	3

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в процедурах.

Статистика по всем дефектам представлена в таблице 3.

Таблица 3 – Статистика по всем дефектам

Статус	Количество	Важность			
		Низкая	Средняя	Высокая	Критическая
Найдено	0	0	0	0	0
Исправлено	0	0	0	0	0
Проверено	0	0	0	0	0
Открыто заново	0	0	0	0	0
Отклонено	0	0	0	0	0

5 Применение

5.1 Назначение и условия применения программы

Цель данного программного продукта заключается в разработке программного продукта, который позволит пользователю получить эстетическое удовольствие, создаст неповторимый игровой опыт, сочетающий приключения, исследование, торговлю и сражения.

Создаваемое приложение будет рассчитано на любого пользователя.

Качество и скорость работы приложения всегда зависит от самих характеристик персонального компьютера. Поэтому приложение должно было быть протестировано на разных машинах. Тестирование проводилось на разных персональных компьютерах и результаты были удовлетворительные.

Сама программа была разработана на программном устройстве со следующими характеристиками:

– процессор AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz;
– ОЗУ: 8Gb;
– память: HDD 512Gb;
– ОС: Windows 10.

5.2 Инсталляция

Для того, чтобы установить программу необходимо запустить файл Setup.exe. Появится мастер установки игрового приложения «Путь авантюриста», представленный на рисунке 4.

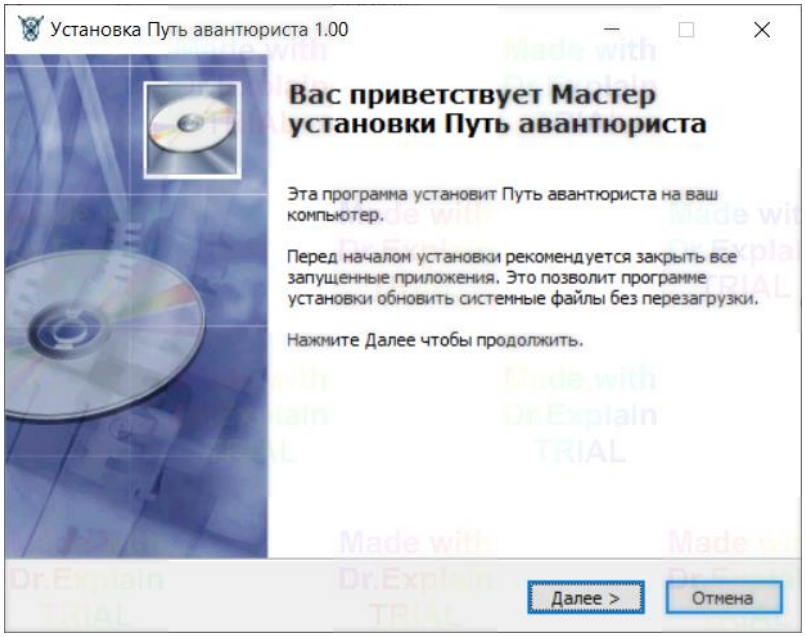

Рисунок 4 – Мастер установки игрового приложения «Путь авантюриста»

После нажатия кнопки «Далее» появляется возможность выбора места для установки программного продукта, представленное на рисунке 5.

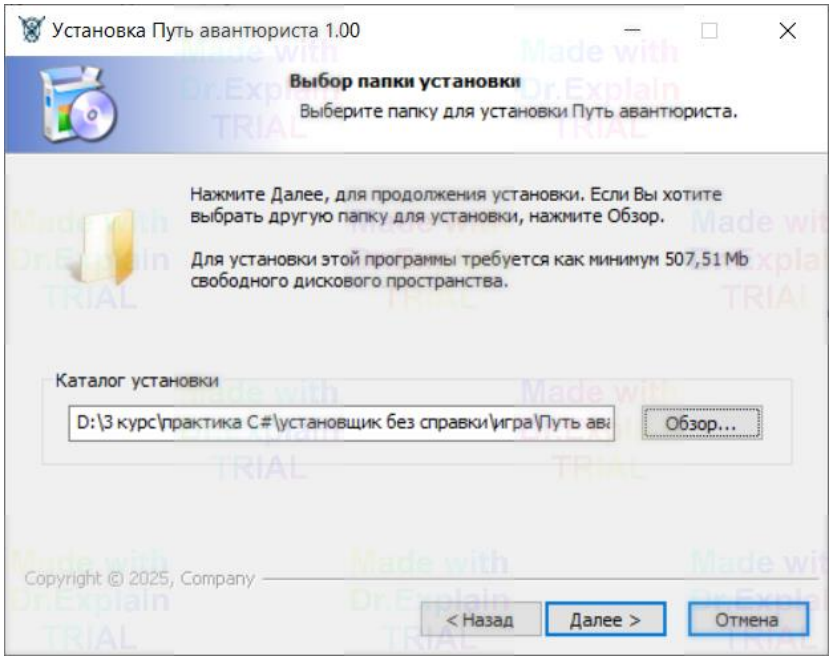

Рисунок 5 – Мастер установки игрового приложения «Путь авантюриста»

После нажатия на кнопку «Далее» появляется возможность выбрать создавать ли ярлыка игрового приложения, представленная на рисунке 6.

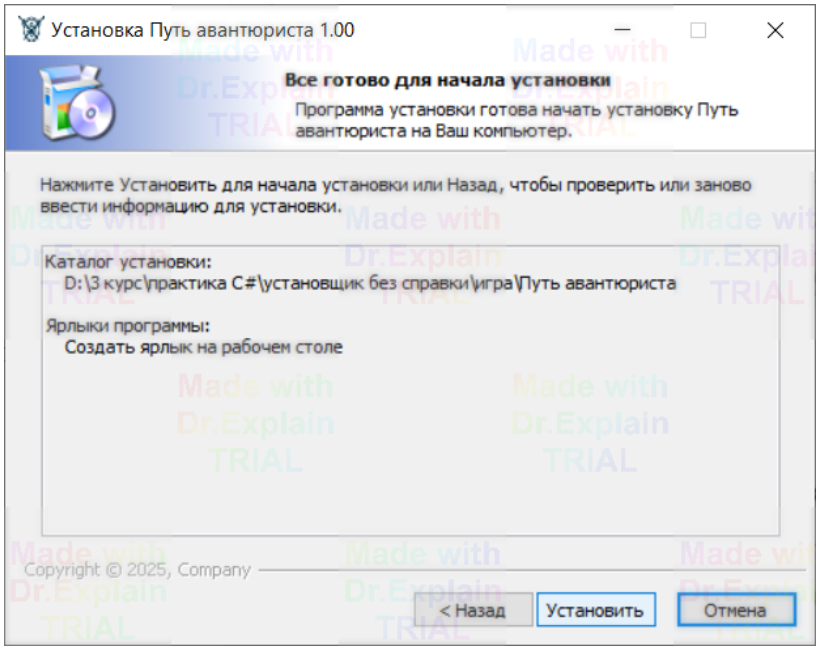

Рисунок 6 – Создание ярлыка

После нажатия на кнопку «Далее» открывается форма завершения установки, представленная на рисунке 7.

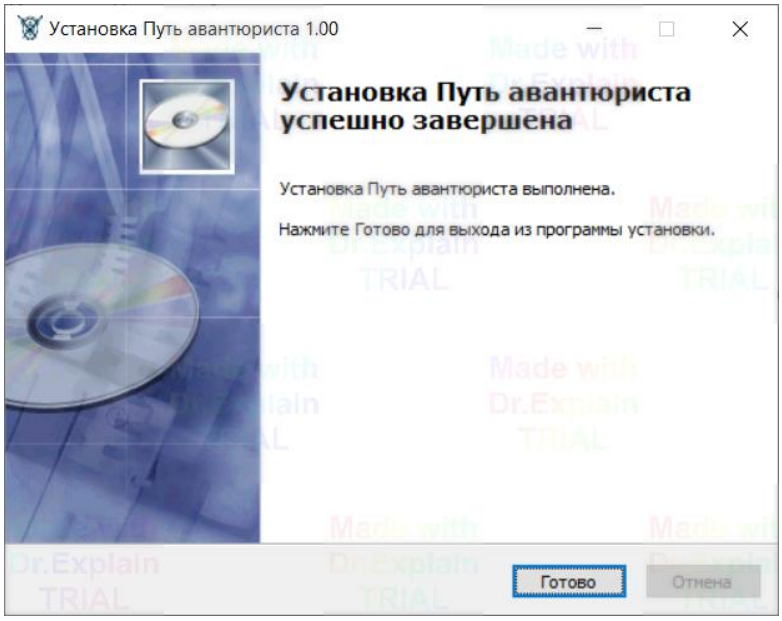

Рисунок 7 – Завершение установки

После установки на «Рабочем столе» появится ярлык для запуска приложения, который представлен на рисунке 7.

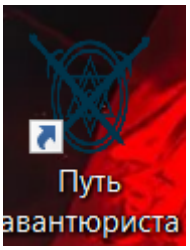

Рисунок 8 – Ярлык приложения

5.3 Выполнение программы

При запуске игры пользователя встречает экран загрузки. Экран загрузки представлен на рисунке 9.

Рисунок 9 – Экран загрузки

После игрок попадает на Главное меню, которое представлено на рисунке 10.

Рисунок 10– Главное меню

У пользователя есть возможность кликнуть по кнопке «Выйти из игры», «Музыка», «Играть», «Справка».

При нажатии на кнопку в виде изображения листка с вопросительным знаком будет открыта справка.

При нажатии на кнопку «Выход из игры» приложение будет закрыто.

При нажатии на кнопу играть вы попадаете на сцену «Входа в город», где вас встречает предисловие к игре. Сцена «Входа в город» представлена на рисунке 11.

Рисунок 11– Сцена «Входа в город»

Для перелистывания предисловия необходимо нажать на кнопки стрелки, расположенную в нижнем правом углу листа предисловия.

После прочтения предисловия, можно продолжить игру нажав на кнопку «В путь», или сбросить игровой прогресс к изначальному (если таковой был), нажав на кнопку «Начать с нуля».

Нажав на одну из данных кнопок вы попадаете на улицу города где расположены здания, в которые можно войти нажав на дверь.

Управление персонажем осуществляется при помощи стрелок клавиатуры, или же клавиш WASD: левая стрелка или клавиша A и правая стрелка или клавиша D это движения в лево и право соответственно, верхняя стрелка или W - прыжок, нижняя стрелка или клавиша S - приседание. Для атаки используется левая клавиша мыши, атака доступна только на сценах сражения.

В левом верхнем углу расположена кнопка перехода в инвентарь (изображена в виде сумки), а в правом верхнем углу кнопка «Меню».

На сцене инвентарь можно просмотреть количество добытых во время игры ресурсов, а также информацию о них, кликнув по значку предмета из левой части инвентаря. Для некоторых предметов, предусмотрена кнопка «Использовать», которая будет располагаться в нижнем правом углу, и нажав на которую будет произведён переход на сцену использования предмета. Покинуть инвентарь можно, а также другие аналогичные сцены можно нажав кнопку «Выход». Инвентарь представлен на рисунке 12.

Рисунок 12 – Инвентарь

Нажав на кнопку «Меню» будет открыто меню, на котором расположены кнопки аналогичные кнопкам главного меню. Нажав на крестик меню будет закрыто. При покупке предмета «Карта» в меню будет добавлена кнопка «Карта»,

которая ведёт на сцену карты, где отмечаются посещённые локации, кликнув по названиям которых можно их посетить.

Нажав на дверь здания гильдии, располагающегося на улице города, будет произведён переход на сцену «Зал гильдии», представленный на рисунке 13.

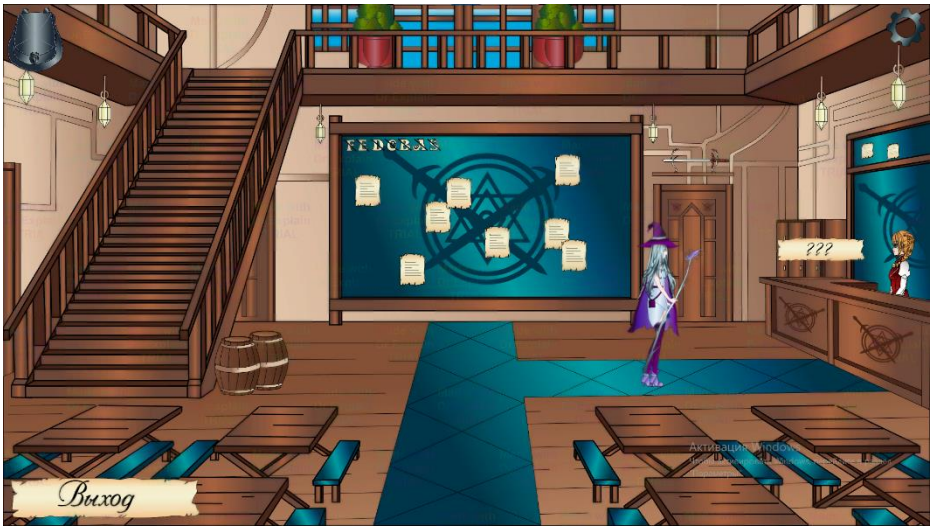

Рисунок 13 – Сцена «Зал Гильдии»

На данной сцене располагается доска заданий, с доступными заданиями и администратор гильдии. Чтобы получить доступ к задания необходимо зарегистрироваться в гильдии, подойдя к администратору, и нажав на кнопку «Начать диалог» (изображена в виде трёх вопросительных знаков), вы получите необходимую информацию о дальнейших действиях. После будет отображена кнопка «Тест», нажав на которую вы попадёте на сцену для прохождения теста.

После прохождения теста вы будете зарегистрированы в гильдии, а ваш текущий ранг будет отображён в верхнем левом углу около кнопки «Инвентарь».

Также будет открыт доступ к заданиям вашего ранга, чтобы перейти к выполнению задания, нужно выбрать ранг на доске заданий, кликнув по нему, выбранный ранг обводится рамкой. Далее нужно кликнуть по листку задания, после чего вы можете ознакомиться с условиями выполнения и наградой за выполнение, и нажать кнопку «Принять», после чего вы будете направлены на сцену соответствующего задания.

Вернуться не выполнив задание, можно нажав на кнопку «Гильдия». При успешном выполнении задания вы получите сообщение о зачислении награды, провалив задание вы также получите соответствующее сообщение с возможностью начать заново нажав на соответствующую кнопку или вернуться в гильдию.

В игре присутствуют функции торговли, перейдя на сцены магазинов, подойдя к торговцам и начав диалог описанным выше способом будут отображены

кнопки «К покупке», «К продаже». Нажав на каждую из которых вы перейдёте на соответствующую сцену.

Для покупки нудно нажать на предмет и в правом углу его описания нажать кнопку «Купить». Слева отображено количество ваших монет, справа цена предмета за единицу.

Аналогично устроена сцена продажи. Для продажи нужно нажать на предмет ввести количество в поле «Шт.» и нажать кнопку «Продать».

Также в игре присутствует возможность украшать дом персонажа. Нажав на кнопку в виде плюса, расположенного около инвентаря, будет открыт список украшений, нажав на плюс повторно, список будет скрыт, нажав на кнопку из списка на сцену будет добавлено соответствующее украшение. Украшения можно перемещать с помощью курсора, удалять нажав клавишу backspace, изменять размер нажимая на + и -, поворачивать в лево и право нажав клавишу Tab, а также поворачивать по окружности, нажав клавишу Q, все эти действия можно применять к украшению на которое наведён курсор.

В игре есть функция Крафта зелий. Приобретя котёл и нажав кнопку «Использовать» в инвентаре, будет произведён переход на сцену котла, где нажав на кнопу ингредиента, и расположив его с помощью мыши в ячейки в соответствии с рецептом будет создано зелье, которое необходимо забрать, кликнув по нему. Скена котелка отображена на рисунке 14.

Рисунок 14 – Сцена «Котелок»

На сценах с врагами в левом верхнем углу будут отображаться единицы здоровья, при наличии зелья здоровья их можно будет восстанавливать, нажав на клавишу F. потеряв все очки здоровья игрок провалит задание.

Заключение

Целью данной учебной практики была разработка программного продукта «Путь авантюриста». В ходе тестирования не было выявлено исключительных ситуаций. Проект работает без сбоев и ошибок. В процессе разработки поставленной задачи были получены знания по использованию платформы Unity и закреплены знания по языку программирования C# и закреплены умения создания собственных функций.

Поставленная задача выполнена в соответствии со всеми ранее задуманными требованиями, созданы и протестированы все необходимые компоненты проекта.

В ходе тестирования все исключительные ситуации были обработаны. Проект работает без сбоев и ошибок. В поставленной задаче был реализован простой и понятный пользовательский интерфейс.

Исходя из этого, можно сделать вывод, что программа реализована успешно.

Список использованных источников

1. Для программистов, не знакомых с Unity [Электронный ресурс]. – Режим доступа: https://unity.com/ru/how-to/programming-unity/ – Дата доступа: 10.04.2025.

2. Unity с нуля: Занятие 2. C# и ООП [Электронный ресурс]. – Режим доступа: https://aelit.by/unity-s-nulya-zanyatie-2-c-i-oop/D.html – Дата доступа: 20.04.2025.

3. Полное руководство по языку программирования C# 13 и платформе .NET 9 [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 25.04.2025.

4. Создаем 2D-игру на Unity: инструкция для новичка [Электронный ресурс]. – Режим доступа: https://proglib.io/p/sozdaem-2d-igru-na-unity-instrukciya-dlya-novichka-2020-09-01 – Дата доступа: 30.04.2025.

5. Руководство по C# [Электронный ресурс]. – Режим доступа: https://vscode.ru/prog-lessons/sozdanie-redaktirovanie-i-udalenie-dannyih-xml-fayla-c-sharp.html – Дата доступа: 01.05.2025.

Приложение А
Листинг программы

Файл DialogAnimator.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class DialogAnimator : MonoBehaviour
{
   public Animator startAnim;
   public DialogManager dm;
   public void OnTriggerEnter2D(Collider2D other)
   {
      startAnim.SetBool("startOpen",true);
   }
   public void OnTriggerExit2D(Collider2D other)
   {
      startAnim.SetBool("startOpen", false);
      dm.EndDialogue();
   }
}
```

Файл Bullet.cs:

```
  using System;
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;

  public class Bullet : MonoBehaviour
  {
     public float speed;
     public float lifetime;
     public float lifetime1;
     public float distance;
     public int damage;
     public LayerMask whatIsSolid;

     private Vector2 direction; // Направление движения
  пули

     public GameObject bulletEffect;

     private void Start()
     {
        // Уничтожаем пулю через заданное время
        Destroy(gameObject, lifetime1);
     }

     public void SetDirection(Vector2 dir)
     {
        direction = dir; // Устанавливаем направление
     }

     private void Update()
     {
        RaycastHit2D hitInfo =
  Physics2D.Raycast(transform.position, direction,
  distance, whatIsSolid);
        if (hitInfo.collider != null)
        {

           if (hitInfo.collider.CompareTag("Enemy"))
           {
              hitInfo.collider.GetComponent<Enemy>().TakeDamage(
  damage);
           }
           Instantiate(bulletEffect, transform.position,
  Quaternion.identity);
           Destroy(gameObject);
        }

        // Двигаем пулю в заданном направлении
        transform.Translate(direction * speed *
  Time.deltaTime);
     }
  }
```

Файл DialogManager.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using UnityEngine.UI;

  public class DialogManager : MonoBehaviour
  {
     public Text dialogueText;
     public Text nameText;

     public Animator boxAnim;
     public Animator startAnim;

     public int x;
     public Button test;

     private Queue<string> sentences;

     private void Start()
     {
        sentences = new Queue<string>();
     }

     public void StartDialogue(Dialogue dialogue)
     {
        boxAnim.SetBool("boxOpen", true);
        startAnim.SetBool("startOpen", false);

        nameText.text = dialogue.name;
        sentences.Clear();

        foreach(string sentence in dialogue.sentences)
        {
           sentences.Enqueue(sentence);
        }
        DisplayNextSentence();
     }

     public void DisplayNextSentence()
     {
        if (sentences.Count == 0)
        {
           EndDialogue();
           if (x == 1)
           { test.gameObject.SetActive(true); }
           return;
```

```
        }
        string sentence = sentences.Dequeue();
        StopAllCoroutines();
        StartCoroutine(TypeSentence(sentence));
    }

    IEnumerator TypeSentence(string sentence)
    {
        dialogueText.text = "";
        foreach(char letter in sentence.ToCharArray())
        {
            dialogueText.text += letter;
            yield return null;
        }
    }

    public void EndDialogue()
    {
        boxAnim.SetBool("boxOpen", false);

    }
}
```

Файл DialogTrigger.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DialogTrigger : MonoBehaviour
{
    public Dialogue dialogue;
    public void TriggerDialogue()
    {

FindObjectOfType<DialogManager>().StartDialogue(di
alogue);
    }
}
```

Файл Dialogue.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
public class Dialogue
{

    public string name;
    [TextArea(3, 10)]
    public string[] sentences;
}
```

Файл taxt.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class taxt : MonoBehaviour
{
    public Text dialogueText;
    private Queue<string> sentences;
    public Dialogue dialogue;
    public Button dalee ;
    public Button Wputi;

    private void Start()
    {
        sentences = new Queue<string>();
        StartDialogue(dialogue);
    }

    public void StartDialogue(Dialogue dialogue)
    {
        foreach (string sentence in dialogue.sentences)
        {
            sentences.Enqueue(sentence);
        }
        DisplayNextSentence();
    }

    public void DisplayNextSentence()
    {
        if (sentences.Count == 0)
        {
            dalee.gameObject.SetActive(false);
            Wputi.gameObject.SetActive(true);
            return;
        }
        string sentence = sentences.Dequeue();
        StopAllCoroutines();
        StartCoroutine(TypeSentence(sentence));
    }

    IEnumerator TypeSentence(string sentence)
    {
        dialogueText.text = "";
        foreach (char letter in sentence.ToCharArray())
        {
            dialogueText.text += letter;
            yield return null;
        }
    }

}
```

Файл hil.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using UnityEngine.UI;

public class hil : MonoBehaviour
{
    public Image a;
    public Image b;
    public Text text;
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.F))
        {
          hiler();
        }
```

```
        string[] stroka1 = File.ReadAllLines("зелья.txt");
        text.text = stroka1[0];
    }
    private void hiler()
    {
        string[] stroka1 = File.ReadAllLines("зелья.txt");
        if
((int.Parse(stroka1[0])>0)&&(FindObjectOfType<нр>().
hp<2))
        {
            FindObjectOfType<нр>().hp += 1;
            stroka1[0]= (int.Parse(stroka1[0])-1).ToString();
            StreamWriter f1 = new StreamWriter("зелья.txt",
false);
            for (int k = 0; k < stroka1.Length; k++)
            {
                f1.WriteLine(stroka1[k]);
            }
            f1.Close();

            if (FindObjectOfType<нр>().hp == 1)
            { a.gameObject.SetActive(true); }
            if (FindObjectOfType<нр>().hp==2)
            { b.gameObject.SetActive(true); }
        }
    }
}
```

Файл porajenie.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class porajenie : MonoBehaviour
{
    public Image a;
    public Image b;

    public Image imgObg1;
    public Sprite spriteImage1;
    public Sprite spriteImage2;

    public Button karta1;

    public GameObject prajenie1;

    public GameObject prajenie2;

    public int kol_vo = 0;

    bool par=false;


    private void Update()
    {
        string[] stroka = File.ReadAllLines("номер
принятого задания.txt");
        string[] stroka1 =
File.ReadAllLines("предметы.txt");

        if ((FindObjectOfType<нр>().hp <= 0
)&&(par==false) )
        {
            stroka1[4]=(int.Parse(stroka1[4]) -
kol_vo).ToString();

            StreamWriter f1 = new
StreamWriter("предметы.txt", false);
            for (int k = 0; k < stroka1.Length; k++)
            {
                f1.WriteLine(stroka1[k]);
            }
            f1.Close();

            if (int.Parse(stroka[0]) == 6)
            {
                prajenie1.gameObject.SetActive(true);

                string[] stroka5 =
File.ReadAllLines("карта.txt");
                if (stroka5[0] == "0")
                {
                    imgObg1.sprite = spriteImage1;
                }
                else
                {
                    imgObg1.sprite = spriteImage2;
                    karta1.gameObject.SetActive(true);
                }

                StartCoroutine(stop());

            }

            else {
                prajenie2.gameObject.SetActive(true);
                StartCoroutine(stop());
            }

            par = true;
        }
    }

    private IEnumerator stop()
    {
        yield return new WaitForSeconds(0f);
        Time.timeScale = 0f;
    }

    public void zanovo()
    {
        par = false;
        prajenie1.SetActive(false);
        prajenie2.SetActive(false);
        karta1.gameObject.SetActive(false);
        FindObjectOfType<нр>().hp = 2;
        { a.gameObject.SetActive(true); }
        { b.gameObject.SetActive(true); }
        Time.timeScale = 1f;
```

```
    }
  }
```

Файл zad_po_sbor.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using System.IO;
  using UnityEngine;
  using UnityEngine.UI;

  public class zad_po_sbor : MonoBehaviour
  {
   public GameObject z1;
  public Text[] text;

  public Image imgObg;
  public Sprite spriteImage1;
  public Sprite spriteImage2;

  public Button karta;

  public GameObject pobeda;

  void Start()
  {
     string[] stroka = File.ReadAllLines("номер
  принятого задания.txt");

     if (int.Parse(stroka[0]) == 6)
     {
        z1.gameObject.SetActive(true);
     }

  }


  void Update()
  {
     string[] stroka = File.ReadAllLines("номер
  принятого задания.txt");

     string[] stroka1 = File.ReadAllLines("предметы.txt");

     if (int.Parse(stroka[0]) == 6)
     {
        text[0].text = stroka1[4] + "/10";
     }


     if ((int.Parse(stroka[0]) == 6) &&
  (int.Parse(stroka1[4]) >= 10))
     {


        stroka1[4] = (int.Parse(stroka1[4]) - 10).ToString();

        StreamWriter f1 = new
  StreamWriter("предметы.txt", false);
        for (int k = 0; k < stroka1.Length; k++)
        {
           f1.WriteLine(stroka1[k]);
        }
        f1.Close();

        stroka[0] = "0";
        StreamWriter f = new StreamWriter("номер
  принятого задания.txt", false);
        for (int k = 0; k < stroka.Length; k++)
        {
           f.WriteLine(stroka[k]);
        }
        f.Close();


        /////////
        string[] stroka2 = File.ReadAllLines("манеты.txt");
        stroka2[0] = (int.Parse(stroka2[0]) +
  300).ToString();

        StreamWriter f2 = new StreamWriter("манеты.txt",
  false);
        for (int k = 0; k < stroka2.Length; k++)
        {
           f2.WriteLine(stroka2[k]);
        }
        f2.Close();

        string[] stroka3 = File.ReadAllLines("выполнение
  зд.txt");
        stroka3[5] = (int.Parse(stroka3[5]) + 1).ToString();

        StreamWriter f3 = new StreamWriter("выполнение
  зд.txt", false);
        for (int k = 0; k < stroka3.Length; k++)
        {
           f3.WriteLine(stroka3[k]);
        }
        f3.Close();

        string[] stroka4 = File.ReadAllLines("ранг.txt");
        stroka4[0] = (int.Parse(stroka4[0]) + 3).ToString();

        StreamWriter f4 = new StreamWriter("ранг.txt",
  false);
        for (int k = 0; k < stroka4.Length; k++)
        {
           f4.WriteLine(stroka4[k]);
        }
        f4.Close();


        pobeda.gameObject.SetActive(true);

        string[] stroka5 = File.ReadAllLines("карта.txt");
        if (stroka5[0] == "0")
        {
           imgObg.sprite = spriteImage1;
        }
        else
        {
           imgObg.sprite = spriteImage2;
           karta.gameObject.SetActive(true);
        }
```

```
        StartCoroutine(stop());

    }



  }

  private IEnumerator stop()
  {
    yield return new WaitForSeconds(1f);
    Time.timeScale = 0f;
  }

}
```

Файл dostun_rang.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using UnityEngine.UI;
  using System.IO;

  public class dostun_rang : MonoBehaviour
  {
    public Button[]  btn;
    public Image imgObg;
    public Sprite[] spriteImage1;

    void Update()
    {
      string[] stroka = File.ReadAllLines("ранг.txt");

      if (int.Parse(stroka[0]) > 0)
      {
        btn[7].gameObject.SetActive(false);
      }

        if
  ((int.Parse(stroka[0])>=1)&&((int.Parse(stroka[0]) < 5)))
      {
        imgObg.sprite = spriteImage1[0];
        btn[0].gameObject.SetActive(true);
      }
      if ((int.Parse(stroka[0]) >= 5) &&
  ((int.Parse(stroka[0]) < 12)))
      {
        imgObg.sprite = spriteImage1[1];
        btn[0].gameObject.SetActive(true);
        btn[1].gameObject.SetActive(true);
      }
      if ((int.Parse(stroka[0]) >= 12) &&
  ((int.Parse(stroka[0]) < 22)))
      {
        imgObg.sprite = spriteImage1[2];
        btn[0].gameObject.SetActive(true);
        btn[1].gameObject.SetActive(true);
        btn[2].gameObject.SetActive(true);
      }
      if ((int.Parse(stroka[0]) >= 22) &&
  ((int.Parse(stroka[0]) < 35)))
      {
        imgObg.sprite = spriteImage1[3];
        btn[0].gameObject.SetActive(true);
        btn[1].gameObject.SetActive(true);
        btn[2].gameObject.SetActive(true);
        btn[3].gameObject.SetActive(true);
      }
      if ((int.Parse(stroka[0]) >= 35) &&
  ((int.Parse(stroka[0]) < 51)))
      {
        imgObg.sprite = spriteImage1[4];
        btn[0].gameObject.SetActive(true);
        btn[1].gameObject.SetActive(true);
        btn[2].gameObject.SetActive(true);
        btn[3].gameObject.SetActive(true);
        btn[4].gameObject.SetActive(true);
      }
      if ((int.Parse(stroka[0]) >= 51) &&
  ((int.Parse(stroka[0]) < 70)))
      {
        imgObg.sprite = spriteImage1[5];
        btn[0].gameObject.SetActive(true);
        btn[1].gameObject.SetActive(true);
        btn[2].gameObject.SetActive(true);
        btn[3].gameObject.SetActive(true);
        btn[4].gameObject.SetActive(true);
        btn[5].gameObject.SetActive(true);
      }
      if (int.Parse(stroka[0]) > 70)
      {
        imgObg.sprite = spriteImage1[5];
        btn[0].gameObject.SetActive(true);
        btn[1].gameObject.SetActive(true);
        btn[2].gameObject.SetActive(true);
        btn[3].gameObject.SetActive(true);
        btn[4].gameObject.SetActive(true);
        btn[5].gameObject.SetActive(true);
        btn[6].gameObject.SetActive(true);
      }
    }
  }
```

Файл wbr_ranga.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using UnityEngine.UI;
  using System.IO;


  public class wbr_ranga : MonoBehaviour
  {
    public int r;
    public Image[] imgObg;
    public Button[] btn;

    public void RangF_clik()
    {
      btn[0].gameObject.SetActive(false);
      btn[1].gameObject.SetActive(false);
      btn[2].gameObject.SetActive(false);

      r = 1;
```

```
        imgObg[0].gameObject.SetActive(true);
        imgObg[1].gameObject.SetActive(false);
        imgObg[2].gameObject.SetActive(false);
        imgObg[3].gameObject.SetActive(false);
        imgObg[4].gameObject.SetActive(false);
        imgObg[5].gameObject.SetActive(false);
        imgObg[6].gameObject.SetActive(false);

        string[] stroka = File.ReadAllLines("выполнение
зд.txt");
        if (int.Parse(stroka[0]) == 0)
        {
            btn[0].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[1]) == 0)
        {
            btn[1].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[2]) == 0)
        {
            btn[2].gameObject.SetActive(true);
        }

    }
    public void RangE_clik()
    {
        btn[0].gameObject.SetActive(false);
        btn[1].gameObject.SetActive(false);
        btn[2].gameObject.SetActive(false);

        r = 2;
        imgObg[0].gameObject.SetActive(false);
        imgObg[1].gameObject.SetActive(true);
        imgObg[2].gameObject.SetActive(false);
        imgObg[3].gameObject.SetActive(false);
        imgObg[4].gameObject.SetActive(false);
        imgObg[5].gameObject.SetActive(false);
        imgObg[6].gameObject.SetActive(false);

        string[] stroka = File.ReadAllLines("выполнение
зд.txt");
        if (int.Parse(stroka[3]) == 0)
        {
            btn[0].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[4]) == 0)
        {
            btn[1].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[5]) == 0)
        {
            btn[2].gameObject.SetActive(true);
        }
    }
    public void RangD_clik()
    {
        btn[0].gameObject.SetActive(false);
        btn[1].gameObject.SetActive(false);
        btn[2].gameObject.SetActive(false);

        r = 3;
        imgObg[0].gameObject.SetActive(false);
        imgObg[1].gameObject.SetActive(false);
        imgObg[2].gameObject.SetActive(true);
        imgObg[3].gameObject.SetActive(false);
        imgObg[4].gameObject.SetActive(false);
        imgObg[5].gameObject.SetActive(false);
        imgObg[6].gameObject.SetActive(false);

        string[] stroka = File.ReadAllLines("выполнение
зд.txt");
        if (int.Parse(stroka[6]) == 0)
        {
            btn[0].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[7]) == 0)
        {
            btn[1].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[8]) == 0)
        {
            btn[2].gameObject.SetActive(true);
        }
    }
    public void RangC_clik()
    {
        btn[0].gameObject.SetActive(false);
        btn[1].gameObject.SetActive(false);
        btn[2].gameObject.SetActive(false);

        r = 4;
        imgObg[0].gameObject.SetActive(false);
        imgObg[1].gameObject.SetActive(false);
        imgObg[2].gameObject.SetActive(false);
        imgObg[3].gameObject.SetActive(true);
        imgObg[4].gameObject.SetActive(false);
        imgObg[5].gameObject.SetActive(false);
        imgObg[6].gameObject.SetActive(false);

        string[] stroka = File.ReadAllLines("выполнение
зд.txt");
        if (int.Parse(stroka[9]) == 0)
        {
            btn[0].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[10]) == 0)
        {
            btn[1].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[11]) == 0)
        {
            btn[2].gameObject.SetActive(true);
        }
    }
    public void RangB_clik()
    {
        btn[0].gameObject.SetActive(false);
        btn[1].gameObject.SetActive(false);
        btn[2].gameObject.SetActive(false);

        r = 5;
        imgObg[0].gameObject.SetActive(false);
        imgObg[1].gameObject.SetActive(false);
        imgObg[2].gameObject.SetActive(false);
```

```
        imgObg[3].gameObject.SetActive(false);
        imgObg[4].gameObject.SetActive(true);
        imgObg[5].gameObject.SetActive(false);
        imgObg[6].gameObject.SetActive(false);

        string[] stroka = File.ReadAllLines("выполнение
зд.txt");
        if (int.Parse(stroka[12]) == 0)
        {
            btn[0].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[13]) == 0)
        {
            btn[1].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[14]) == 0)
        {
            btn[2].gameObject.SetActive(true);
        }
    }
    public void RangA_clik()
    {
        btn[0].gameObject.SetActive(false);
        btn[1].gameObject.SetActive(false);
        btn[2].gameObject.SetActive(false);

        r = 6;
        imgObg[0].gameObject.SetActive(false);
        imgObg[1].gameObject.SetActive(false);
        imgObg[2].gameObject.SetActive(false);
        imgObg[3].gameObject.SetActive(false);
        imgObg[4].gameObject.SetActive(false);
        imgObg[5].gameObject.SetActive(true);
        imgObg[6].gameObject.SetActive(false);

        string[] stroka = File.ReadAllLines("выполнение
зд.txt");
        if (int.Parse(stroka[15]) == 0)
        {
            btn[0].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[16]) == 0)
        {
            btn[1].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[17]) == 0)
        {
            btn[2].gameObject.SetActive(true);
        }

    }
    public void RangS_clik()
    {
        btn[0].gameObject.SetActive(false);
        btn[1].gameObject.SetActive(false);
        btn[2].gameObject.SetActive(false);

        r = 7;
        imgObg[0].gameObject.SetActive(false);
        imgObg[1].gameObject.SetActive(false);
        imgObg[2].gameObject.SetActive(false);
        imgObg[3].gameObject.SetActive(false);
        imgObg[4].gameObject.SetActive(false);
        imgObg[5].gameObject.SetActive(false);
        imgObg[6].gameObject.SetActive(true);

        string[] stroka = File.ReadAllLines("выполнение
зд.txt");
        if (int.Parse(stroka[18]) == 0)
        {
            btn[0].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[19]) == 0)
        {
            btn[1].gameObject.SetActive(true);
        }
        if (int.Parse(stroka[20]) == 0)
        {
            btn[2].gameObject.SetActive(true);
        }

    }
}
```

Файл Zadanija.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class Zadanija : MonoBehaviour
{
    public Image imgObg;
    public Sprite[] spriteImage1;

    public Переход perehod;

    public void Z1_clic()
    {
        if (FindObjectOfType<wbr_ranga>().r == 1)
        {
            perehod.levelToLoad = 13;

            string[] stroka = File.ReadAllLines("номер
принятого задания.txt");
            stroka[0]=1.ToString();
            StreamWriter f1 = new StreamWriter("номер
принятого задания.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
            }
            f1.Close();

            string[] stroka1 =
File.ReadAllLines("посещённые локации.txt");
            stroka1[0] = 1.ToString();
            StreamWriter f11 = new
StreamWriter("посещённые локации.txt", false);
            for (int k = 0; k < stroka1.Length; k++)
            {
                f11.WriteLine(stroka1[k]);
            }
            f11.Close();
```

```
        }
        if (FindObjectOfType<wbr_ranga>().r == 2)
        {
            perehod.levelToLoad = 14;
            string[] stroka = File.ReadAllLines("номер
принятого задания.txt");
            stroka[0] = 4.ToString();
            StreamWriter f1 = new StreamWriter("номер
принятого задания.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
            }
            f1.Close();
        }
        if (FindObjectOfType<wbr_ranga>().r == 3)
        {
            perehod.levelToLoad = 5;
        }
        if (FindObjectOfType<wbr_ranga>().r == 4)
        {
            perehod.levelToLoad = 5;
        }
        if (FindObjectOfType<wbr_ranga>().r == 5)
        {
            perehod.levelToLoad = 5;
        }
        if (FindObjectOfType<wbr_ranga>().r == 6)
        {
            perehod.levelToLoad = 5;
        }
        if (FindObjectOfType<wbr_ranga>().r == 7)
        {
            perehod.levelToLoad = 5;
        }

        int s = 0;
        for (int i = 1; i < 8; i++)
        {
            if (FindObjectOfType<wbr_ranga>().r == i)
            {
                imgObg.sprite = spriteImage1[s];

            }
            s += 3;
        }
        imgObg.gameObject.SetActive(true);

    }
    public void Z2_clic()
    {
        if (FindObjectOfType<wbr_ranga>().r == 1)
        {
            perehod.levelToLoad = 12;


        }
        if (FindObjectOfType<wbr_ranga>().r == 2)
        {
            perehod.levelToLoad = 14;
            string[] stroka = File.ReadAllLines("номер
принятого задания.txt");
            stroka[0] = 5.ToString();
            StreamWriter f1 = new StreamWriter("номер
принятого задания.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
            }
            f1.Close();
        }
        if (FindObjectOfType<wbr_ranga>().r == 3)
        {
            perehod.levelToLoad = 5;
        }
        if (FindObjectOfType<wbr_ranga>().r == 4)
        {
            perehod.levelToLoad = 5;
        }
        if (FindObjectOfType<wbr_ranga>().r == 5)
        {
            perehod.levelToLoad = 5;
        }
        if (FindObjectOfType<wbr_ranga>().r == 6)
        {
            perehod.levelToLoad = 5;
        }
        if (FindObjectOfType<wbr_ranga>().r == 7)
        {
            perehod.levelToLoad = 5;
        }

        int s = 1;
        for (int i = 1; i < 8; i++)
        {
            if (FindObjectOfType<wbr_ranga>().r == i)
            {
                imgObg.sprite = spriteImage1[s];

            }
            s += 3;
        }
        imgObg.gameObject.SetActive(true);

    }
    public void Z3_clic()
    {
        if (FindObjectOfType<wbr_ranga>().r == 1)
        {
            perehod.levelToLoad = 13;

            string[] stroka = File.ReadAllLines("номер
принятого задания.txt");
            stroka[0] = 2.ToString();
            StreamWriter f1 = new StreamWriter("номер
принятого задания.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
            }
            f1.Close();
```

```
            string[] stroka1 =
File.ReadAllLines("посещённые локации.txt");
            stroka1[0] = 1.ToString();
            StreamWriter f11 = new
StreamWriter("посещённые локации.txt", false);
            for (int k = 0; k < stroka1.Length; k++)
            {
               f11.WriteLine(stroka1[k]);
            }
            f11.Close();
         }
         if (FindObjectOfType<wbr_ranga>().r == 2)
         {
            perehod.levelToLoad = 15;

            string[] stroka = File.ReadAllLines("номер
принятого задания.txt");
            stroka[0] = 6.ToString();
            StreamWriter f1 = new StreamWriter("номер
принятого задания.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
               f1.WriteLine(stroka[k]);
            }
            f1.Close();

            string[] stroka1 =
File.ReadAllLines("посещённые локации.txt");
            stroka1[1] = 1.ToString();
            StreamWriter f11 = new
StreamWriter("посещённые локации.txt", false);
            for (int k = 0; k < stroka1.Length; k++)
            {
               f11.WriteLine(stroka1[k]);
            }
            f11.Close();
         }
         if (FindObjectOfType<wbr_ranga>().r == 3)
         {
            perehod.levelToLoad = 5;
         }
         if (FindObjectOfType<wbr_ranga>().r == 4)
         {
            perehod.levelToLoad = 5;
         }
         if (FindObjectOfType<wbr_ranga>().r == 5)
         {
            perehod.levelToLoad = 5;
         }
         if (FindObjectOfType<wbr_ranga>().r == 6)
         {
            perehod.levelToLoad = 5;
         }
         if (FindObjectOfType<wbr_ranga>().r == 7)
         {
            perehod.levelToLoad = 5;
         }

         int s = 2;
         for (int i = 1; i < 8; i++)
         {
            if (FindObjectOfType<wbr_ranga>().r == i)
            {
               imgObg.sprite = spriteImage1[s];

            }
            s += 3;
         }
         imgObg.gameObject.SetActive(true);
      }

      public void kr()
      {
         imgObg.gameObject.SetActive(false);
      }

   }
```

Файл kniga.cs:

```
   using System.Collections;
   using System.Collections.Generic;
   using UnityEngine;
   using UnityEngine.UI;
   using System.IO;

   public class kniga : MonoBehaviour
   {
      public GameObject[] scrol;
      public Image img;
      public Sprite[] sprites;
      public Image fon;
      public Sprite[] sprites_fon;
      public Button exit;
      public Button exit1;

      void Start()
      {
         string[] stroka =
   File.ReadAllLines("использовать.txt");

         if (int.Parse(stroka[0])==0)
         {
            fon.sprite = sprites_fon[0];
            exit1.gameObject.SetActive(true);
            img.gameObject.SetActive(false);
         }
         if (int.Parse(stroka[0]) == 1)
         {
            fon.sprite = sprites_fon[1];
            exit.gameObject.SetActive(true);
            scrol[0].gameObject.SetActive(true);
            img.sprite = sprites[3];
         }
         if (int.Parse(stroka[0]) == 2)
         {
            fon.sprite = sprites_fon[2];
            exit.gameObject.SetActive(true);
            scrol[1].gameObject.SetActive(true);
            img.sprite = sprites[0];
         }
      }

      public void button_click(int i)
      {
```

```
            img.sprite = sprites[i];
        }

    }
```

Файл Draggable1.cs:

```
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using System.IO;
    using UnityEngine.EventSystems;

    public class Draggable1 : MonoBehaviour,
    IDragHandler, IBeginDragHandler

    {
        private Canvas canvas;
        private RectTransform rectTransform;
        private Vector3 originalPosition;

        private Vector2 offset;
        void Awake()
        {
            rectTransform = GetComponent<RectTransform>();
            canvas = GetComponentInParent<Canvas>();
            originalPosition = rectTransform.anchoredPosition;


        }

        public void OnBeginDrag(PointerEventData
    eventData)
        {
            // Сохранение оригинальной позиции при начале
    перетаскивания


            // Сохраняем смещение между курсором и
    объектом
            RectTransform rectTransform =
    GetComponent<RectTransform>();
            originalPosition = rectTransform.anchoredPosition;
            offset = eventData.position -
    RectTransformUtility.WorldToScreenPoint(eventData.pr
    essEventCamera, rectTransform.position);
        }

        public void OnDrag(PointerEventData eventData)
        {
            Vector2 position;

    RectTransformUtility.ScreenPointToLocalPointInRectan
    gle(canvas.transform as RectTransform,
    eventData.position, canvas.worldCamera, out position);


            Vector2 newPosition = eventData.position - offset;
            RectTransform rectTransform =
    GetComponent<RectTransform>();
            rectTransform.position = newPosition;
        }
        public Vector3 GetCurrentPosition()
        {
            return rectTransform.anchoredPosition;
        }
    }
```

Файл Draggable2.cs:

```
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using System.IO;
    using UnityEngine.EventSystems;

    public class Draggable2 : MonoBehaviour,
    IDragHandler, IBeginDragHandler

    {
        private Canvas canvas;
        private RectTransform rectTransform;
        private Vector3 originalPosition;

        private Vector2 offset;
        void Awake()
        {
            rectTransform = GetComponent<RectTransform>();
            canvas = GetComponentInParent<Canvas>();
            originalPosition = rectTransform.anchoredPosition;


        }

        public void OnBeginDrag(PointerEventData
    eventData)
        {
            // Сохранение оригинальной позиции при начале
    перетаскивания


            // Сохраняем смещение между курсором и
    объектом
            RectTransform rectTransform =
    GetComponent<RectTransform>();
            originalPosition = rectTransform.anchoredPosition;
            offset = eventData.position -
    RectTransformUtility.WorldToScreenPoint(eventData.pr
    essEventCamera, rectTransform.position);
        }

        public void OnDrag(PointerEventData eventData)
        {
            Vector2 position;

    RectTransformUtility.ScreenPointToLocalPointInRectan
    gle(canvas.transform as RectTransform,
    eventData.position, canvas.worldCamera, out position);
            ;

            Vector2 newPosition = eventData.position - offset;
            RectTransform rectTransform =
    GetComponent<RectTransform>();
            rectTransform.position = newPosition;
        }
```

```
    public Vector3 GetCurrentPosition()
    {
        return rectTransform.anchoredPosition;
    }
}
```

Файл JewelryManager1.cs:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
using static System.Net.Mime.MediaTypeNames;

public class JewelryManager1 : MonoBehaviour
{
    public GameObject[] jewelryPrefabs; // массив
префабов
    public Button[] buttons;
    private Dictionary<int, int> quantities = new
Dictionary<int, int>();
    public Transform parentTransform;
    public Vector2 spawnPosition;


    public List<Draggable1> draggableJewelry = new
List<Draggable1>();

    void Start()
    {
        LoadQuantities();
        LoadJewelryPositions();
        for (int i = 0; i < buttons.Length; i++)
        {
            int index = i;
            buttons[i].onClick.AddListener(() =>
OnJewelryButtonClick(index));
        }
    }

    void Update()
    {
        // Проверяем, нажата ли клавиша Backspace
        if (Input.GetKeyDown(KeyCode.Backspace))
        {
            RemoveSelectedJewelry();
        }


    }


    void LoadQuantities()
    {
        string[] lines = File.ReadAllLines("растения.txt");
        for (int i = 0; i < lines.Length; i++)
        {
            if (int.TryParse(lines[i], out int quantity))
            {
                quantities[i] = quantity;
            }
        }
    }


    void LoadJewelryPositions()
    {
        if (File.Exists("positions_kotel.txt"))
        {
            string[] lines =
File.ReadAllLines("positions_kotel.txt");
            foreach (string line in lines)
            {
                string[] parts = line.Split('|');
                if (parts.Length == 3)
                {
                    string name = parts[0].Substring(0,
parts[0].Length - 7);
                    float x = float.Parse(parts[1]);
                    float y = float.Parse(parts[2]);


                    GameObject prefab =
System.Array.Find(jewelryPrefabs, p => p.name ==
name);
                    if (prefab != null)
                    {
                        GameObject jewelryInstance =
Instantiate(prefab, parentTransform);

jewelryInstance.GetComponent<RectTransform>().anch
oredPosition = new Vector2(x, y);


jewelryInstance.AddComponent<Draggable1>(); //
добавляем компонент для перетаскивания

draggableJewelry.Add(jewelryInstance.GetComponent<
Draggable1>()); // добавляем в список
                    }
                }
            }
        }
    }

    public void OnJewelryButtonClick(int index)
    {
        if (quantities.ContainsKey(index) &&
quantities[index] > 0)
        {
            GameObject jewelryInstance =
Instantiate(jewelryPrefabs[index], parentTransform);

jewelryInstance.GetComponent<RectTransform>().anch
oredPosition = spawnPosition;
            quantities[index]--; // Уменьшаем количество
            UpdateQuantityFile();
            jewelryInstance.AddComponent<Draggable1>();

draggableJewelry.Add(jewelryInstance.GetComponent<
Draggable1>());
```

```
        }
    }

    void UpdateQuantityFile()
    {
        using (StreamWriter writer = new
StreamWriter("растения.txt"))
        {
            foreach (var quantity in quantities)
            {
                writer.WriteLine(quantity.Value);
            }
        }
    }

    private void OnApplicationQuit()
    {
        SavePositions(); // Сохраняем позиции при
выходе из приложения
    }

    public void SavePositions()
    {
        File.WriteAllText("positions_kotel.txt",
string.Empty);
        foreach (var draggable in draggableJewelry)
        {
            if (draggable != null)
            {
                string positionData =
$"{draggable.gameObject.name}|{draggable.GetCurrent
Position().x}|{draggable.GetCurrentPosition().y}\n"; //
Добавляем размер
                File.AppendAllText("positions_kotel.txt",
positionData); // Сохраняем позицию в файл
            }
        }
        draggableJewelry.Clear(); // Очищаем список
после сохранения
    }

    private void RemoveSelectedJewelry()
    {
        PointerEventData pointerData = new
PointerEventData(EventSystem.current) { position =
Input.mousePosition };
        List<RaycastResult> results = new
List<RaycastResult>();
        EventSystem.current.RaycastAll(pointerData,
results);

        foreach (RaycastResult result in results)
        {
            Draggable1 draggable =
result.gameObject.GetComponent<Draggable1>();
            if (draggable != null)
            {
                int index =
GetJewelryIndex(draggable.gameObject.name);
                if (index != -1)
                {
                    // Увеличиваем количество в словаре
                    quantities[index]++;

                    // Обновляем файл после изменения
количества
                    UpdateQuantityFile();

                    // Удаляем объект из списка
draggableJewelry и уничтожаем его
                    draggableJewelry.Remove(draggable);


                    string[] stroka1 =
File.ReadAllLines("крафт.txt");


                    if
((Math.Round(draggable.GetCurrentPosition().x,2)==
437)&&
(Math.Round(draggable.GetCurrentPosition().y,2) ==
146))
                    {
                        stroka1[0] = 0.ToString();


                    }
                    if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
625.2) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
144.8))
                    {
                        stroka1[1] = 0.ToString();
                    }
                    if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
351.3) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
23.9))
                    {
                        stroka1[2] = 0.ToString();
                    }
                    if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
528.8) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
23.3))
                    {
                        stroka1[3] = 0.ToString();
                    }
                    if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
712.8) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
23.4))
                    {
                        stroka1[4] = 0.ToString();
                    }
                    if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
437) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) == -
107.5))
```

```
                {
                    stroka1[5] = 0.ToString();
                }
                if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
627) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) == -
107.6))
                {
                    stroka1[6] = 0.ToString();
                }

                StreamWriter f = new
StreamWriter("крафт.txt", false);
                for (int i = 0; i < stroka1.Length; i++)
                {
                    f.WriteLine(stroka1[i]);
                }
                f.Close();

                Destroy(draggable.gameObject);
                break;
            }
        }
    }
    private int GetJewelryIndex(string name)
    {
        for (int i = 0; i < jewelryPrefabs.Length; i++)
        {
            if (name.StartsWith("раст" + (i + 1).ToString()+
"(Clone)"))
            {
                return i;
            }
        }
        return -1;
    }



}
```

Файл JewelryManager2.cs:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
using static System.Net.Mime.MediaTypeNames;

public class JewelryManager2 : MonoBehaviour
{
    public GameObject jewelryPrefab; // единственный
префаб
    public Button button; // единственная кнопка
    private int quantity; // количество
    public Transform parentTransform;
    public Vector2 spawnPosition;
    private List<Draggable2> draggableJewelry = new
List<Draggable2>();

    void Start()
    {
        LoadQuantity();
        LoadJewelryPositions();

button.onClick.AddListener(OnJewelryButtonClick);
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Backspace))
        {
            RemoveSelectedJewelry();
        }
    }

    void LoadQuantity()
    {
        string[] lines = File.ReadAllLines("предметы.txt");
        if (lines.Length >= 5 && int.TryParse(lines[4], out
quantity)) // берем 5 строку
        {
            Debug.Log($"Quantity loaded: {quantity}");
        }
    }

    void LoadJewelryPositions()
    {
        if (File.Exists("positions_kotel_krist.txt"))
        {
            string[] lines =
File.ReadAllLines("positions_kotel_krist.txt");
            foreach (string line in lines)
            {
                string[] parts = line.Split('|');
                if (parts.Length == 3)
                {
                    string name = parts[0].Substring(0,
parts[0].Length - 7); // обрезаем "(Clone)"
                    float x = float.Parse(parts[1]);
                    float y = float.Parse(parts[2]);
                    if (jewelryPrefab.name == name)
                    {
                        GameObject jewelryInstance =
Instantiate(jewelryPrefab, parentTransform);

jewelryInstance.GetComponent<RectTransform>().anch
oredPosition = new Vector2(x, y);

jewelryInstance.AddComponent<Draggable2>();

draggableJewelry.Add(jewelryInstance.GetComponent<
Draggable2>());
                    }
                }
            }
        }
    }
```

```
    public void OnJewelryButtonClick()
    {
        if (quantity > 0)
        {
            GameObject jewelryInstance =
Instantiate(jewelryPrefab, parentTransform);

jewelryInstance.GetComponent<RectTransform>().anch
oredPosition = spawnPosition;
            quantity--;
            UpdateQuantityFile();
            jewelryInstance.AddComponent<Draggable2>();

draggableJewelry.Add(jewelryInstance.GetComponent<
Draggable2>());
        }
    }

    void UpdateQuantityFile()
    {
        var lines = File.ReadAllLines("предметы.txt");
        if (lines.Length >= 5)
        {
            lines[4] = quantity.ToString(); // обновляем 5
строку
            File.WriteAllLines("предметы.txt", lines);
        }
    }

    private void OnApplicationQuit()
    {
        SavePositions();
    }

    public void SavePositions()
    {
        using (StreamWriter writer = new
StreamWriter("positions_kotel_krist.txt"))
        {
            foreach (var draggable in draggableJewelry)
            {
                if (draggable != null)
                {
                    string positionData =
$"{draggable.gameObject.name}|{draggable.GetCurrent
Position().x}|{draggable.GetCurrentPosition().y}\n";
                    writer.Write(positionData);
                }
            }
        }
        draggableJewelry.Clear();
    }

    private void RemoveSelectedJewelry()
    {
        PointerEventData pointerData = new
PointerEventData(EventSystem.current) { position =
Input.mousePosition };
        List<RaycastResult> results = new
List<RaycastResult>();
        EventSystem.current.RaycastAll(pointerData,
results);
        foreach (RaycastResult result in results)
        {
            Draggable2 draggable =
result.gameObject.GetComponent<Draggable2>();
            if (draggable != null)
            {
                quantity++; // увеличиваем количество
                UpdateQuantityFile();
                draggableJewelry.Remove(draggable);

                string[] stroka1 =
File.ReadAllLines("крафт.txt");


                if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
437) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
146))
                {
                    stroka1[0] = 0.ToString();



                }
                if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
625.2) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
144.8))
                {
                    stroka1[1] = 0.ToString();
                }
                if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
351.3) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
23.9))
                {
                    stroka1[2] = 0.ToString();
                }
                if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
528.8) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
23.3))
                {
                    stroka1[3] = 0.ToString();
                }
                if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
712.8) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) ==
23.4))
                {
                    stroka1[4] = 0.ToString();
                }
                if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
437) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) == -
107.5))
                {
```

```
                stroka1[5] = 0.ToString();
            }
            if
((Math.Round(draggable.GetCurrentPosition().x, 2) ==
627) &&
(Math.Round(draggable.GetCurrentPosition().y, 2) == -
107.6))
            {
                stroka1[6] = 0.ToString();
            }

            StreamWriter f = new
StreamWriter("крафт.txt", false);
            for (int i = 0; i < stroka1.Length; i++)
            {
                f.WriteLine(stroka1[i]);
            }
            f.Close();

            Destroy(draggable.gameObject);
            break;
        }
    }
  }
}
```

Файл scitkolrastenij.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class scitkolrastenij : MonoBehaviour
{
    public Text[] text;

    void Update()
    {
        string[] stroka = File.ReadAllLines("растения.txt");
        string[] stroka1 =
File.ReadAllLines("предметы.txt");

        for (int i = 0; i < stroka.Length; i++)
        {
            text[i].text = stroka[i];
        }
        text[15].text = stroka1[4];
    }
}
```

Файл Slot.cs:

```
using System.Collections;
using System.Collections.Generic;
using System.IO;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.EventSystems;
using static UnityEngine.RuleTile.TilingRuleOutput;

public class Slot : MonoBehaviour, IDropHandler
{
    public float targetX; // Задайте значение x
    public float targetY; // Задайте значение y
    public string y;
    public int x;
    public void OnDrop(PointerEventData eventData)
    {
        string[] stroka = File.ReadAllLines("крафт.txt");

        Draggable1 item =
eventData.pointerDrag.GetComponent<Draggable1>();

        Draggable2 item2 =
eventData.pointerDrag.GetComponent<Draggable2>();

        if (item != null)
        {
            if (((y == "Slot1") && (int.Parse(stroka[0]) ==
0)) || ((y == "Slot2") && (int.Parse(stroka[1]) == 0)) ||
((y == "Slot3") && (int.Parse(stroka[2]) == 0)) || ((y ==
"Slot4") && (int.Parse(stroka[3]) ==0)) || ((y == "Slot5")
&& (int.Parse(stroka[4]) == 0)) || ((y == "Slot6") &&
(int.Parse(stroka[5]) == 0)) || ((y == "Slot7") &&
(int.Parse(stroka[6]) == 0)))
            {


                // Устанавливаем позицию элемента в
заданные координаты (targetX, targetY)
                RectTransform itemRectTransform =
item.GetComponent<RectTransform>();
                itemRectTransform.anchoredPosition = new
Vector2(targetX, targetY);

                if (y == "Slot1")
                {
                    if (item.name == "раст9(Clone)")
                    {
                        x = 1;
                    }
                    else if (item.name == "раст13(Clone)")
                    { x = 2; }
                    else if (item.name == "раст2(Clone)")
                    { x = 3; }
                    else
                    { x = -1; }

                    stroka[0] = x.ToString();
                }

                if (y == "Slot2")
                {
                    if (item.name == "раст14(Clone)")
                    {
                        x = 1;
                    }
                    else if (item.name == "раст4(Clone)")
                    { x = 2; }
                    else if (item.name == "раст15(Clone)")
                    { x = 3; }
                    else
                    { x = -1; }
```

```
            stroka[1] = x.ToString();
        }

        if (y == "Slot3")
        {
            if (item.name == "раст3(Clone)")
            {
                x = 1;
            }
            else
            { x = -1; }

            stroka[2] = x.ToString();
        }

        if (y == "Slot4")
        {
            if (item.name == "раст(Clone)")
            {
                x = 1;
            }
            else
            { x = -1; }

            stroka[3] = x.ToString();
        }

        if (y == "Slot5")
        {
            if (item.name == "раст7(Clone)")
            {
                x = 1;
            }
            else if (item.name == "раст6(Clone)")
            { x = 2; }
            else if (item.name == "раст1(Clone)")
            { x = 3; }
            else
            { x = -1; }

            stroka[4] = x.ToString();
        }
        if (y == "Slot6")
        {
            if (item.name == "раст10(Clone)")
            {
                x = 1;
            }
            else
            { x = -1; }

            stroka[5] = x.ToString();
        }
        if (y == "Slot7")
        {
            if (item.name == "раст8(Clone)")
            {
                x = 1;
            }
            else
            { x = -1; }

            stroka[6] = x.ToString();
        }

        StreamWriter f = new
StreamWriter("крафт.txt", false);
        for (int i = 0; i < stroka.Length; i++)
        {
            f.WriteLine(stroka[i]);
        }
        f.Close();
    }
}

if (item2 != null)
{
    if (((y == "Slot1") && (int.Parse(stroka[0]) ==
0)) || ((y == "Slot2") && (int.Parse(stroka[1]) == 0)) ||
((y == "Slot3") && (int.Parse(stroka[2]) == 0)) || ((y ==
"Slot4") && (int.Parse(stroka[3]) == 0)) || ((y ==
"Slot5") && (int.Parse(stroka[4]) == 0)) || ((y ==
"Slot6") && (int.Parse(stroka[5]) == 0)) || ((y ==
"Slot7") && (int.Parse(stroka[6]) == 0)))
    {


        // Устанавливаем позицию элемента в
заданные координаты (targetX, targetY)
        RectTransform itemRectTransform2 =
item2.GetComponent<RectTransform>();
        itemRectTransform2.anchoredPosition = new
Vector2(targetX, targetY);

        if (y == "Slot1")
        {
            x = -1;

            stroka[0] = x.ToString();
        }

        if (y == "Slot2")
        {
            x = -1;

            stroka[1] = x.ToString();
        }

        if (y == "Slot3")
        {
            x = -1;

            stroka[2] = x.ToString();
        }

        if (y == "Slot4")
        {

            x = 1;


            stroka[3] = x.ToString();
        }
```

```
            if (y == "Slot5")
            {
                x = -1;

                stroka[4] = x.ToString();
            }
            if (y == "Slot6")
            {
                x = -1;

                stroka[5] = x.ToString();
            }
            if (y == "Slot7")
            {
                x = -1;

                stroka[6] = x.ToString();
            }

            StreamWriter f = new
  StreamWriter("крафт.txt", false);
            for (int i = 0; i < stroka.Length; i++)
            {
                f.WriteLine(stroka[i]);
            }
            f.Close();
        }
    }
  }
}
```

Файл sosdanie.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using System;

public class sosdanie : MonoBehaviour
{

    public Button[] dt;


    private void Start()
    {
        string[]                stroka2                =
  File.ReadAllLines("зелья_кнопки.txt");

        for (int i = 0; i < stroka2.Length; i++)
        {
            if (int.Parse(stroka2[i]) >0)
            {
                dt[i].gameObject.SetActive(true);
            }
        }

    }

    public void Sosdanie()
    {
        string[] stroka = File.ReadAllLines("крафт.txt");
        string[]                stroka2                =
File.ReadAllLines("зелья_кнопки.txt");
        int n = 0;
        for (int i = 0; i < stroka.Length; i++)
        {
            if (int.Parse(stroka[i]) ==1)
            {
                n += 1;
            }
        }
        if (n == 7)
        {
            dt[0].gameObject.SetActive(true);
            stroka2[0]=(int.Parse(stroka2[0])+1).ToString();
            n = 0;

            GameObject[]                objects                =
GameObject.FindObjectsOfType<GameObject>();

            foreach (GameObject obj in objects)
            {
                if (obj.name == "раст9(Clone)")
                {
                    // Получаем координаты объекта
                    double                objX                =
Math.Round(obj.transform.position.x, 2);
                    double                objY                =
Math.Round(obj.transform.position.y, 2);
                    // Проверяем, совпадают ли координаты
                    if (objX == 1397 && objY == 686)
                    {
                        // Удаляем первый найденный объект
                        Destroy(obj);
                        break; // Выходим из цикла после
удаления
                    }
                }
            }
            foreach (GameObject obj in objects)
            {
                if (obj.name == "раст14(Clone)")
                {
                    // Получаем координаты объекта
                    double                objX                =
Math.Round(obj.transform.position.x, 2);
                    double                objY                =
Math.Round(obj.transform.position.y, 2);


                    // Проверяем, совпадают ли координаты
                    if (objX == 1585.2 && objY == 684.8)
                    {
                        // Удаляем первый найденный объект
                        Destroy(obj);
                        break; // Выходим из цикла после
удаления
                    }
                }
            }
            foreach (GameObject obj in objects)
            {
```

```
        if (obj.name == "раст3(Clone)")
        {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1311.3 && objY == 563.9)
            {
                // Удаляем первый найденный объект
                Destroy(obj);
                break; // Выходим из цикла после удаления
            }
        }
    }

    foreach (GameObject obj in objects)
    {
        if (obj.name == "раст(Clone)")
        {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1488.8 && objY == 563.3)
            {
                // Удаляем первый найденный объект
                Destroy(obj);
                break; // Выходим из цикла после удаления
            }
        }
    }
    foreach (GameObject obj in objects)
    {
        if (obj.name == "раст7(Clone)")
        {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1672.8 && objY == 563.4)
            {
                // Удаляем первый найденный объект
                Destroy(obj);
                break; // Выходим из цикла после удаления
            }
        }
    }
    foreach (GameObject obj in objects)
    {
        if (obj.name == "раст10(Clone)")
        {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1397 && objY == 432.5)
            {
                // Удаляем первый найденный объект
                Destroy(obj);
                break; // Выходим из цикла после удаления
            }
        }
    }
    foreach (GameObject obj in objects)
    {
        if (obj.name == "раст8(Clone)")
        {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1587 && objY == 432.4)
            {
                // Удаляем первый найденный объект
                Destroy(obj);
                break; // Выходим из цикла после удаления
            }
        }
    }
    StreamWriter f = new StreamWriter("крафт.txt", false);
    for (int i = 0; i < stroka.Length; i++)
    {
        stroka[i] = "0";
        f.WriteLine(stroka[i]);
    }
    f.Close();
}

if ((int.Parse(stroka[0]) == 2)&& (int.Parse(stroka[1]) == 2) && (int.Parse(stroka[4]) == 2) && (int.Parse(stroka[3]) == 1))
{
    dt[1].gameObject.SetActive(true);
    stroka2[1] = (int.Parse(stroka2[1]) + 1).ToString();

    GameObject[] objects = GameObject.FindObjectsOfType<GameObject>();

    foreach (GameObject obj in objects)
    {
        if (obj.name == "раст13(Clone)")
        {
            // Получаем координаты объекта
```

```
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);
            // Проверяем, совпадают ли координаты
            if (objX == 1397 && objY == 686)
            {
              // Удаляем первый найденный объект
              Destroy(obj);
              break; // Выходим из цикла после удаления
            }
          }
        }
        foreach (GameObject obj in objects)
        {
          if (obj.name == "раст4(Clone)")
          {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);


            // Проверяем, совпадают ли координаты
            if (objX == 1585.2 && objY == 684.8)
            {
              // Удаляем первый найденный объект
              Destroy(obj);
              break; // Выходим из цикла после удаления
            }
          }
        }
        foreach (GameObject obj in objects)
        {
          if (obj.name == "раст(Clone)")
          {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1488.8 && objY == 563.3)
            {
              // Удаляем первый найденный объект
              Destroy(obj);
              break; // Выходим из цикла после удаления
            }
          }
        }
        foreach (GameObject obj in objects)
        {
          if (obj.name == "раст6(Clone)")
          {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1672.8 && objY == 563.4)
            {
              // Удаляем первый найденный объект
              Destroy(obj);
              break; // Выходим из цикла после удаления
            }
          }
        }
        StreamWriter f = new StreamWriter("крафт.txt", false);
        for (int i = 0; i < stroka.Length; i++)
        {
          stroka[i] = "0";
          f.WriteLine(stroka[i]);
        }
        f.Close();
      }
      if ((int.Parse(stroka[0]) == 3) && (int.Parse(stroka[1]) == 3) && (int.Parse(stroka[4]) == 3) && (int.Parse(stroka[3]) == 1))
      {
        dt[2].gameObject.SetActive(true);
        stroka2[2] = (int.Parse(stroka2[2]) + 1).ToString();

        GameObject[] objects = GameObject.FindObjectsOfType<GameObject>();

        foreach (GameObject obj in objects)
        {
          if (obj.name == "раст2(Clone)")
          {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);
            // Проверяем, совпадают ли координаты
            if (objX == 1397 && objY == 686)
            {
              // Удаляем первый найденный объект
              Destroy(obj);
              break; // Выходим из цикла после удаления
            }
          }
        }
        foreach (GameObject obj in objects)
        {
          if (obj.name == "раст15(Clone)")
          {
            // Получаем координаты объекта
            double objX = Math.Round(obj.transform.position.x, 2);
            double objY = Math.Round(obj.transform.position.y, 2);
```

```
            // Проверяем, совпадают ли координаты
            if (objX == 1585.2 && objY == 684.8)
            {
               // Удаляем первый найденный объект
               Destroy(obj);
               break; // Выходим из цикла после
удаления
            }
         }
      }
      foreach (GameObject obj in objects)
      {
         if (obj.name == "раст(Clone)")
         {
            // Получаем координаты объекта
            double objX =
Math.Round(obj.transform.position.x, 2);
            double objY =
Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1488.8 && objY == 563.3)
            {
               // Удаляем первый найденный объект
               Destroy(obj);
               break; // Выходим из цикла после
удаления
            }
         }
      }
      foreach (GameObject obj in objects)
      {
         if (obj.name == "раст1(Clone)")
         {
            // Получаем координаты объекта
            double objX =
Math.Round(obj.transform.position.x, 2);
            double objY =
Math.Round(obj.transform.position.y, 2);

            // Проверяем, совпадают ли координаты
            if (objX == 1672.8 && objY == 563.4)
            {
               // Удаляем первый найденный объект
               Destroy(obj);
               break; // Выходим из цикла после
удаления
            }
         }
      }
      StreamWriter f = new StreamWriter("крафт.txt",
false);
      for (int i = 0; i < stroka.Length; i++)
      {
         stroka[i] = "0";
         f.WriteLine(stroka[i]);
      }
      f.Close();
   }

      StreamWriter f2 = new
StreamWriter("зелья_кнопки.txt", false);
      for (int i = 0; i < stroka2.Length; i++)
      {
         f2.WriteLine(stroka2[i]);
      }
      f2.Close();
   }

   public void sele_clik(int n)
   {
      string[] stroka1 = File.ReadAllLines("зелья.txt");
      string[] stroka2 =
File.ReadAllLines("зелья_кнопки.txt");

      if (n == 0) {
         stroka1[0] = (int.Parse(stroka1[0]) + 1).ToString();
         stroka2[0] = 0.ToString();

      }
      if (n == 1)
      {
         stroka1[1] = (int.Parse(stroka1[1]) + 1).ToString();
         stroka2[1] = 0.ToString();

      }
      if (n == 2)
      {
         stroka1[2] = (int.Parse(stroka1[2]) + 1).ToString();
         stroka2[2] = 0.ToString();
      }
      StreamWriter f1 = new StreamWriter("зелья.txt",
false);
      for (int i = 0; i < stroka1.Length; i++)
      {
         f1.WriteLine(stroka1[i]);
      }
      f1.Close();

      StreamWriter f2 = new
StreamWriter("зелья_кнопки.txt", false);
      for (int i = 0; i < stroka2.Length; i++)
      {
         f2.WriteLine(stroka2[i]);
      }
      f2.Close();

      dt[n].gameObject.SetActive(false);
   }
}
```

Файл invent.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class invent : MonoBehaviour
```

```
{
    public Text[] text;

    public Image img;
    public Sprite[] sprites;
    public Переход perehod;
    public Button ispols;
    private void Start()
    {
        img.sprite = sprites[0];
        ispols.gameObject.SetActive(false);
    }

    void Update()
    {
        string[] stroka = File.ReadAllLines("растения.txt");
        string[] stroka1 =
File.ReadAllLines("предметы.txt");
        string[] stroka2 = File.ReadAllLines("манеты.txt");
        string[] stroka3 = File.ReadAllLines("карта.txt");
        string[] stroka4 = File.ReadAllLines("зелья.txt");
        string[] stroka5 =
File.ReadAllLines("количество.txt");

        for (int i = 0; i < stroka.Length; i++)
        {
            text[i].text=stroka[i];
        }

        for (int i = 0; i < stroka1.Length; i++)
        {
            text[i+ stroka.Length].text = stroka1[i];
        }

        for (int i = 0; i < stroka2.Length; i++)
        {
            text[i+ stroka.Length+ stroka1.Length].text =
stroka2[i];
        }

        for (int i = 0; i < stroka3.Length; i++)
        {
            text[i+stroka.Length+ stroka1.Length+
stroka2.Length].text = stroka3[i];
        }

        for (int i = 0; i < stroka4.Length; i++)
        {
            text[i+ stroka.Length+ stroka1.Length+
stroka2.Length+ stroka3.Length].text = stroka4[i];
        }

        for (int i = 0; i < stroka5.Length; i++)
        {
            text[i + stroka.Length + stroka1.Length +
stroka2.Length + stroka3.Length+ stroka4.Length].text =
stroka5[i];
        }
    }

    public void button_clic(int i)
    {
        //для смены номера сцены на кнопке
использовать
        ispols.gameObject.SetActive(false);
        img.sprite=sprites[i];

        string[] stroka1 =
File.ReadAllLines("предметы.txt");
        string[] stroka3 = File.ReadAllLines("карта.txt");
        if ((i == 1) && (int.Parse(stroka3[0])>0))
        {
            ispols.gameObject.SetActive(true);
            perehod.levelToLoad = 16;
        }
        img.sprite = sprites[i];
        if ((i == 2) && (int.Parse(stroka1[0]) > 0))
        {
            ispols.gameObject.SetActive(true);
            perehod.levelToLoad = 10;

            string[] stroka =
File.ReadAllLines("использовать.txt");
            stroka[0]=0.ToString();
            StreamWriter f1 = new
StreamWriter("использовать.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
            }
            f1.Close();

        }
        img.sprite = sprites[i];
        if ((i == 4) && (int.Parse(stroka1[1]) > 0))
        {
            ispols.gameObject.SetActive(true);
            perehod.levelToLoad = 14;
        }
        img.sprite = sprites[i];
        if ((i == 5) && (int.Parse(stroka1[2]) > 0))
        {
            ispols.gameObject.SetActive(true);
            perehod.levelToLoad = 10;

            string[] stroka =
File.ReadAllLines("использовать.txt");
            stroka[0] = 1.ToString();
            StreamWriter f1 = new
StreamWriter("использовать.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
            }
            f1.Close();
        }
        img.sprite = sprites[i];
        if ((i == 6) && (int.Parse(stroka1[3]) > 0))
        {
            ispols.gameObject.SetActive(true);
            perehod.levelToLoad = 10;

            string[] stroka =
File.ReadAllLines("использовать.txt");
```

```
            stroka[0] = 2.ToString();
            StreamWriter f1 = new
  StreamWriter("использовать.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
               f1.WriteLine(stroka[k]);
            }
            f1.Close();
         }
      }
   }
```

Файл kupl_prod.cs:

```
   using System.Collections;
   using System.Collections.Generic;
   using UnityEngine;
   using System.IO;

   public class kupl_prod : MonoBehaviour
   {
      public void k_p(int i)
      {
         string[] stroka = File.ReadAllLines("Купля-
   прдажа.txt");
         stroka[0]=i.ToString();
         StreamWriter f1 = new StreamWriter("Купля-
   прдажа.txt", false);
         for (int k = 0; k < stroka.Length; k++)
         {
            f1.WriteLine(stroka[k]);
         }
         f1.Close();

      }
   }
```

Файл magas.cs:

```
   using System.Collections;
   using System.Collections.Generic;
   using UnityEngine;
   using UnityEngine.UI;
   using System.IO;

   public class magas : MonoBehaviour
   {
      public GameObject[] scrol;
      public Text text1;
      public Image img;
      public Sprite[] sprites;
      public Button[] bt;
      void Start()
      {
         string[] stroka = File.ReadAllLines("Номер
   сцены.txt");
         string[] stroka1 = File.ReadAllLines("Купля-
   прдажа.txt");
         string[] stroka2 = File.ReadAllLines("цена в
   торговой лавке.txt");
         string[] stroka3 = File.ReadAllLines("цена
   продажи.txt");
         string[] stroka4 = File.ReadAllLines("цена в
   магазине магии.txt");
         if ((int.Parse(stroka[0]) == 6) &&
   (int.Parse(stroka1[0]) == 0))
         {
            scrol[0].SetActive(true);
            img.sprite=sprites[0];
            bt[0].gameObject.SetActive(true);
            text1.text= stroka2[0];
         }
         if ((int.Parse(stroka[0]) == 6) &&
   (int.Parse(stroka1[0]) == 1))
         {
            scrol[1].SetActive(true);
            img.sprite = sprites[1];
            bt[1].gameObject.SetActive(true);
            text1.text = stroka3[0];
         }
         if ((int.Parse(stroka[0]) == 7))
         {
            scrol[2].SetActive(true);
            img.sprite = sprites[2];
            bt[2].gameObject.SetActive(true);
            text1.text = stroka4[0];
         }
      }


   }
```

Файл pokupca.cs:

```
   using System.Collections;
   using System.Collections.Generic;
   using UnityEngine;
   using UnityEngine.UI;
   using System.IO;

   public class pokupca : MonoBehaviour
   {
      public int num;

      public Text text0;
      public Text text1;
      public Image img;
      public Sprite[] sprites;



     private void Start()
      {
         num = 0;


      }

      void Update()
      {
         string[] stroka = File.ReadAllLines("манеты.txt");
         text0.text = stroka[0];
      }

      public void button_click(int i)
```

```
    {
        num = i;
        img.sprite = sprites[i];

        string[] stroka1 = File.ReadAllLines("Номер
сцены.txt");
        if ((int.Parse(stroka1[0]) == 6))
        {
            string[] stroka = File.ReadAllLines("цена в
торговой лавке.txt");
            text1.text = stroka[i];
        }
        if ((int.Parse(stroka1[0]) == 7))
        {
            string[] stroka = File.ReadAllLines("цена в
магазине магии.txt");
            text1.text = stroka[i];
        }
    }

    public void pokur()
    {
        string[] stroka = File.ReadAllLines("манеты.txt");
        string[] stroka1 =
File.ReadAllLines("предметы.txt");
        string[] stroka2 =
File.ReadAllLines("количество.txt");

        if (int.Parse(stroka[0])>= int.Parse(text1.text))
        {

            stroka[0] = (int.Parse(stroka[0]) -
int.Parse(text1.text)).ToString();

            text0.text = stroka[0];

            StreamWriter f1 = new
StreamWriter("манеты.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
            }
            f1.Close();

            if (num == 0)
            {
                stroka1[0] = (int.Parse(stroka1[0]) +
1).ToString();

                StreamWriter f = new
StreamWriter("предметы.txt", false);
                for (int k = 0; k < stroka1.Length; k++)
                {
                    f.WriteLine(stroka1[k]);
                }
                f.Close();

            }
            if (num == 1)
            {
                stroka1[2] = (int.Parse(stroka1[2]) +
1).ToString();

                StreamWriter f = new
StreamWriter("предметы.txt", false);
                for (int k = 0; k < stroka1.Length; k++)
                {
                    f.WriteLine(stroka1[k]);
                }
                f.Close();
            }
            if ((num >= 2) && (num <= 27))
            {
                stroka2[num - 2] = (int.Parse(stroka2[num - 2])
+ 1).ToString();

                StreamWriter f = new
StreamWriter("количество.txt", false);
                for (int k = 0; k < stroka2.Length; k++)
                {
                    f.WriteLine(stroka2[k]);
                }
                f.Close();
            }
        }
    }

    public void pokur_magii()
    {
        string[] stroka = File.ReadAllLines("манеты.txt");
        string[] stroka1 =
File.ReadAllLines("предметы.txt");
        string[] stroka2 = File.ReadAllLines("карта.txt");
        string[] stroka3 = File.ReadAllLines("зелья.txt");

        if (int.Parse(stroka[0]) >= int.Parse(text1.text))
        {

            stroka[0] = (int.Parse(stroka[0]) -
int.Parse(text1.text)).ToString();

            text0.text = stroka[0];

            StreamWriter f1 = new
StreamWriter("манеты.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
            }
            f1.Close();


            if (num == 0)
            {
                stroka2[0] = (int.Parse(stroka2[0]) +
1).ToString();

                StreamWriter f = new
StreamWriter("карта.txt", false);
                for (int k = 0; k < stroka2.Length; k++)
                {
                    f.WriteLine(stroka2[k]);
                }
                f.Close();
```

```
            }
            if (num == 1)
            {
                stroka3[0] = (int.Parse(stroka3[0]) +
    1).ToString();

                StreamWriter f = new
    StreamWriter("зелья.txt", false);
                for (int k = 0; k < stroka3.Length; k++)
                {
                    f.WriteLine(stroka3[k]);
                }
                f.Close();
            }
            if ((num ==2))
            {
                stroka1[1] = (int.Parse(stroka1[1]) +
    1).ToString();

                StreamWriter f = new
    StreamWriter("предметы.txt", false);
                for (int k = 0; k < stroka1.Length; k++)
                {
                    f.WriteLine(stroka1[k]);
                }
                f.Close();
            }
            if ((num == 3))
            {
                stroka1[3] = (int.Parse(stroka1[3]) +
    1).ToString();

                StreamWriter f = new
    StreamWriter("предметы.txt", false);
                for (int k = 0; k < stroka1.Length; k++)
                {
                    f.WriteLine(stroka1[k]);
                }
                f.Close();
            }
        }



    }
}
```

Файл prodaja.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class prodaja : MonoBehaviour
{
    public Text[] text;

    public int num;

    public Text text0;
    public Text text1;
    public Text text2;
    public Image img;
    public Sprite[] sprites;

    void Start()
    {
        num = 0;
    }

    void Update()
    {
        string[] stroka = File.ReadAllLines("растения.txt");
        string[] stroka1 =
File.ReadAllLines("предметы.txt");
        string[] stroka2 = File.ReadAllLines("зелья.txt");

        for (int i = 0; i < stroka.Length; i++)
        {
            text[i].text = stroka[i];
        }

            text[stroka.Length].text = stroka1[4];

        for (int i = 1; i < stroka2.Length; i++)//зелья в
файле со 2 строки
        {
            text[i + stroka.Length + 1-1].text = stroka2[i];
        }
    }

    public void button_click(int i)
    {
        num = i;
        img.sprite = sprites[i];
        string[] stroka = File.ReadAllLines("цена
продажи.txt");
        text1.text = stroka[i];
    }

    public void prod()
    {
        string[] stroka = File.ReadAllLines("манеты.txt");
        string[] stroka1 =
File.ReadAllLines("растения.txt");
        string[] stroka2 =
File.ReadAllLines("предметы.txt");
        string[] stroka3 = File.ReadAllLines("зелья.txt");

        if (int.Parse(text[num].text) >= int.Parse(text2.text))
        {

            stroka[0] = (int.Parse(stroka[0]) +
(int.Parse(text1.text)* int.Parse(text2.text))).ToString();

            text0.text = stroka[0];

            StreamWriter f1 = new
StreamWriter("манеты.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f1.WriteLine(stroka[k]);
```

```
            }
            f1.Close();


            if ((num >= 0) && (num <= 14))
            {
                stroka1[num] = (int.Parse(stroka1[num]) -
int.Parse(text2.text)).ToString();

                StreamWriter f = new
StreamWriter("растения.txt", false);
                for (int k = 0; k < stroka1.Length; k++)
                {
                    f.WriteLine(stroka1[k]);
                }
                f.Close();
            }

            if (num == 15)
            {
                stroka2[4] = (int.Parse(stroka2[4]) -
int.Parse(text2.text)).ToString();

                StreamWriter f = new
StreamWriter("предметы.txt", false);
                for (int k = 0; k < stroka2.Length; k++)
                {
                    f.WriteLine(stroka2[k]);
                }
                f.Close();

            }

            if ((num >= 16) && (num <= 17))
            {
                stroka3[num-15] = (int.Parse(stroka3[num-15])
- int.Parse(text2.text)).ToString();

                StreamWriter f = new
StreamWriter("растения.txt", false);
                for (int k = 0; k < stroka3.Length; k++)
                {
                    f.WriteLine(stroka3[k]);
                }
                f.Close();
            }
        }
    }

}
```

Файл Draggable.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using UnityEngine.EventSystems;

public class Draggable : MonoBehaviour, IDragHandler,
IBeginDragHandler
{
    private Canvas canvas;
    private RectTransform rectTransform;
    private Vector3 originalPosition;

    void Awake()
    {
        rectTransform = GetComponent<RectTransform>();
        canvas = GetComponentInParent<Canvas>();
        originalPosition = rectTransform.anchoredPosition;
    }

    public void OnBeginDrag(PointerEventData
eventData)
    {
        // Сохранение оригинальной позиции при начале
перетаскивания
        originalPosition = rectTransform.anchoredPosition;
    }

    public void OnDrag(PointerEventData eventData)
    {
        Vector2 position;

RectTransformUtility.ScreenPointToLocalPointInRectan
gle(canvas.transform as RectTransform,
eventData.position, canvas.worldCamera, out position);
        rectTransform.anchoredPosition = position;
    }

    public Vector3 GetCurrentPosition()
    {
        return rectTransform.anchoredPosition;
    }
}
```

Файл inwentari.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class inwentari : MonoBehaviour
{
    public Text[] text;
    public GameObject plus;
    public bool pl=true;



    void Update()
    {
        string[] stroka =
File.ReadAllLines("количество.txt");

        for (int i = 0; i < stroka.Length; i++)
        {
            text[i].text = stroka[i];
        }
    }

    public void Plus()
    {
```

```
        if (!pl)
        {
            plus.gameObject.SetActive(false);
            pl = true;

        }
        else
        {
            plus.gameObject.SetActive(true);
            pl = false;
        }
    }

}
```

Файл JewelryManager.cs:

```
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
using static System.Net.Mime.MediaTypeNames;

public class JewelryManager : MonoBehaviour
{
    public GameObject[] jewelryPrefabs; // массив
префабов украшений
    public Button[] buttons;
    private Dictionary<int, int> quantities = new
Dictionary<int, int>();
    public Transform parentTransform;
    public Vector2 spawnPosition;

    private List<Draggable> draggableJewelry = new
List<Draggable>();

    void Start()
    {
        LoadQuantities();
        LoadJewelryPositions();
        for (int i = 0; i < buttons.Length; i++)
        {
            int index = i;
            buttons[i].onClick.AddListener(() =>
OnJewelryButtonClick(index));
        }
    }

    void Update()
    {
        // Проверяем, нажата ли клавиша Backspace
        if (Input.GetKeyDown(KeyCode.Backspace))
        {
            RemoveSelectedJewelry();
        }

        // Проверяем, нажата ли клавиша Tab для
поворота украшения
        if (Input.GetKeyDown(KeyCode.Tab))
        {
            RotateSelectedJewelry();
        }

        // Проверяем, нажаты ли клавиши + и - для
изменения размера
        if (Input.GetKeyDown(KeyCode.Equals)) //
Клавиша + на большинстве раскладок
        {
            ChangeSelectedJewelrySize(1.1f); //
Увеличиваем размер
        }
        else if (Input.GetKeyDown(KeyCode.Minus))
        {
            ChangeSelectedJewelrySize(0.9f); // Уменьшаем
размер
        }

        //  Проверяем, нажата ли клавиша Tab для
поворота украшения по окружности
        if (Input.GetKeyDown(KeyCode.Q))
        {
            RotateSelectedJewelry1();
        }
    }


    void LoadQuantities()
    {
        string[] lines =
File.ReadAllLines("количество.txt");
        for (int i = 0; i < lines.Length; i++)
        {
            if (int.TryParse(lines[i], out int quantity))
            {
                quantities[i] = quantity;
            }
        }
    }


    void LoadJewelryPositions()
    {
        if (File.Exists("positions.txt"))
        {
            string[] lines =
File.ReadAllLines("positions.txt");
            foreach (string line in lines)
            {
                string[] parts = line.Split('|');
                if (parts.Length == 5) // Изменили на 4 для
учета размера
                {
                    string name = parts[0].Substring(0,
parts[0].Length - 7);
                    float x = float.Parse(parts[1]);
                    float y = float.Parse(parts[2]);
                    float size = float.Parse(parts[3]);
                    float angle = float.Parse(parts[4]);
                    float size1 = size;
                    if (size < 0)
                    {
                        size1 *= -1;
                    }
```

```
                GameObject prefab =
System.Array.Find(jewelryPrefabs, p => p.name ==
name);
                if (prefab != null)
                {
                    GameObject jewelryInstance =
Instantiate(prefab, parentTransform);

jewelryInstance.GetComponent<RectTransform>().anch
oredPosition = new Vector2(x, y);
                    jewelryInstance.transform.rotation =
Quaternion.Euler(0, 0, angle); // Устанавливаем угол
                    jewelryInstance.transform.localScale =
new Vector3(size, size1, 1); // Устанавливаем размер

jewelryInstance.AddComponent<Draggable>(); //
добавляем компонент для перетаскивания

draggableJewelry.Add(jewelryInstance.GetComponent<
Draggable>()); // добавляем в список
                }
            }
        }
    }
}

public void OnJewelryButtonClick(int index)
{
    if (quantities.ContainsKey(index) &&
quantities[index] > 0)
    {
        GameObject jewelryInstance =
Instantiate(jewelryPrefabs[index], parentTransform);

jewelryInstance.GetComponent<RectTransform>().anch
oredPosition = spawnPosition;
        quantities[index]--; // Уменьшаем количество
        UpdateQuantityFile();
        jewelryInstance.AddComponent<Draggable>();

draggableJewelry.Add(jewelryInstance.GetComponent<
Draggable>());
    }
}

void UpdateQuantityFile()
{
    using (StreamWriter writer = new
StreamWriter("количество.txt"))
    {
        foreach (var quantity in quantities)
        {
            writer.WriteLine(quantity.Value);
        }
    }
}

private void OnApplicationQuit()
{
    SavePositions(); // Сохраняем позиции при
выходе из приложения
}

public void SavePositions()
{
    File.WriteAllText("positions.txt", string.Empty);
    foreach (var draggable in draggableJewelry)
    {
        string positionData =
$"{draggable.gameObject.name}|{draggable.GetCurrent
Position().x}|{draggable.GetCurrentPosition().y}|{dragg
able.transform.localScale.x}|{draggable.transform.rotatio
n.eulerAngles.z}\n"; // Добавляем размер
        File.AppendAllText("positions.txt",
positionData); // Сохраняем позицию в файл
    }
    draggableJewelry.Clear(); // Очищаем список
после сохранения
}

private void RemoveSelectedJewelry()
{
    PointerEventData pointerData = new
PointerEventData(EventSystem.current) { position =
Input.mousePosition };
    List<RaycastResult> results = new
List<RaycastResult>();
    EventSystem.current.RaycastAll(pointerData,
results);

    foreach (RaycastResult result in results)
    {
        Draggable draggable =
result.gameObject.GetComponent<Draggable>();
        if (draggable != null)
        {
            int index =
GetJewelryIndex(draggable.gameObject.name);
            if (index != -1)
            {
                // Увеличиваем количество в словаре
                quantities[index]++;

                // Обновляем файл после изменения
количества
                UpdateQuantityFile();

                // Удаляем объект из списка
draggableJewelry и уничтожаем его
                draggableJewelry.Remove(draggable);
                Destroy(draggable.gameObject);
                break;
            }
        }
    }
}
private int GetJewelryIndex(string name)
{
    for (int i = 0; i < jewelryPrefabs.Length; i++)
    {
        if (name.StartsWith("укр" + (i + 1).ToString()+
"(Clone)"))
        {
```

```
            return i;
        }
    }
    return -1;
}

private void RotateSelectedJewelry()
{
    PointerEventData pointerData = new
PointerEventData(EventSystem.current) { position =
Input.mousePosition };
    List<RaycastResult> results = new
List<RaycastResult>();

    EventSystem.current.RaycastAll(pointerData,
results);

    foreach (RaycastResult result in results)
    {
        Draggable draggable =
result.gameObject.GetComponent<Draggable>();
        if (draggable != null)
        {
            draggable.transform.localScale = new
Vector3(-draggable.transform.localScale.x,
draggable.transform.localScale.y, 1); // Поворачиваем в
противоположную сторону
            break; // Поворачиваем только первое
найденное украшение
        }
    }
}

private void RotateSelectedJewelry1()
{
    PointerEventData pointerData = new
PointerEventData(EventSystem.current) { position =
Input.mousePosition };
    List<RaycastResult> results = new
List<RaycastResult>();
    EventSystem.current.RaycastAll(pointerData,
results);

    foreach (RaycastResult result in results)
    {
        Draggable draggable =
result.gameObject.GetComponent<Draggable>();
        if (draggable != null)
        {
            draggable.transform.Rotate(0, 0, 90); //
Поворачиваем на 90 градусов
            break; // Поворачиваем только первое
найденное украшение
        }
    }
}
private void ChangeSelectedJewelrySize(float
scaleFactor)
{
    PointerEventData pointerData = new
PointerEventData(EventSystem.current) { position =
Input.mousePosition };
    List<RaycastResult> results = new
List<RaycastResult>();
    EventSystem.current.RaycastAll(pointerData,
results);
    foreach (RaycastResult result in results)
    {
        Draggable draggable =
result.gameObject.GetComponent<Draggable>();
        if (draggable != null)
        {
            draggable.transform.localScale *= scaleFactor;
// Изменяем размер
            break; // Изменяем только первое найденное
украшение
        }
    }
}
}
```

Файл Enemy.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Enemy : MonoBehaviour
{
    public int health;
    public GameObject deathEffect;

    private void Update()
    {
        if (health <= 0)
        {
            Instantiate(deathEffect, transform.position,
Quaternion.identity);

            FindObjectOfType<Rang>().heal = 0;

            Destroy(gameObject);
        }

    }
    public void TakeDamage(int damage)
    {  health -= damage; }
}
```

Файл Exit.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Exit : MonoBehaviour
{
    public void ExitGame()
    {
        Application.Quit();
    }
}
```

Файл Gan.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class Gan : MonoBehaviour
{
    public float offset;
    public GameObject bullet;
    public Transform shotPoint;
    private float timeBtwShots;
    public float startTimeBtwShots;
    // Укажите объект, к которому будут
прикрепляться пули
    public Transform parentObject;

    private void Update()
    {
        Vector3 difference =
Camera.main.ScreenToWorldPoint(Input.mousePosition)
- transform.position;
        float rotZ = Mathf.Atan2(difference.y, difference.x)
* Mathf.Rad2Deg;
        transform.rotation = Quaternion.Euler(0f, 0f, rotZ +
offset);
        if (timeBtwShots <= 0)
        {
            if (Input.GetMouseButtonDown(0))
            {
                // Создаем пулю с фиксированным
вращением
                GameObject newBullet = Instantiate(bullet,
shotPoint.position, Quaternion.Euler(0f, 0f, 0f));

                // Устанавливаем родителя для новой пули
                newBullet.transform.SetParent(parentObject);

                // Задаем направление полета для пули
                Bullet bulletScript =
newBullet.GetComponent<Bullet>();

bulletScript.SetDirection(difference.normalized); //
Передаем направление

                timeBtwShots = startTimeBtwShots;
            }
        }
        else
        {
            timeBtwShots -= Time.deltaTime;
        }
    }
}
```

Файл Karta.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;
using UnityEngine.UI;

public class Karta : MonoBehaviour
{
    public Button[] bt;
    public Image imgObg1;
    public Sprite[] sprites;
    void Start()
    {
        string[] stroka = File.ReadAllLines("посещённые
локации.txt");
        if (int.Parse(stroka[0])==1)
        {
            bt[0].gameObject.SetActive(true);
            imgObg1.sprite = sprites[0];
        }
        if (int.Parse(stroka[1]) == 1)
        {
            bt[1].gameObject.SetActive(true);
            imgObg1.sprite = sprites[1];
        }

    }
}
```

Файл Konec_igr.cs:

```
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.UI;

public class Konec_igr : MonoBehaviour
{

    public GameObject endddd;

    bool prod = false;


    private void Update()
    {
        string[] stroka = File.ReadAllLines("выполнение
зд.txt");


        int z21 = 0;
        for (int i = 0; i <stroka.Length; i++)
        {
            if (int.Parse(stroka[i]) == 1)
            {  z21++; }
        }
        if ((z21== 21) && (prod == false))
        {
            endddd.gameObject.SetActive(true);
            StartCoroutine(stop());
             prod = true;
        }
    }
    private IEnumerator stop()
    {
        yield return new WaitForSeconds(3f);
        Time.timeScale = 0f;
    }

    public void prodolj()
    {
        Time.timeScale = 1f;
        endddd.gameObject.SetActive(false);
    }
```

```
}
```

Файл Kristall.cs:

```
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
public class Kristall : MonoBehaviour
{
    public GameObject kristall_pref;
    public Transform parentTransform;
    private void Awake()
    {
        // Получаем родителя текущего объекта и
записываем его в переменную
        parentTransform = transform.parent;
    }
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            string[] stroka =
File.ReadAllLines("предметы.txt");
            stroka[4]=(int.Parse(stroka[4])+1).ToString();
            FindObjectOfType<porajenie>().kol_vo += 1;

            StreamWriter f = new
StreamWriter("предметы.txt", false);
            for (int i = 0; i < stroka.Length; i++)
            {
                f.WriteLine(stroka[i]);
            }
            f.Close();
            // Задаем массив возможных позиций
            Vector2[] positions = new Vector2[]
            {
            new Vector2(453f, 171f),
            new Vector2(-1104f, -553f),
            new Vector2(1533f, -540f)
            };

            // Генерируем случайный индекс
            int randomIndex = Random.Range(0,
positions.Length);


            GameObject prefab = kristall_pref;
            if (prefab != null)
            {
                GameObject jewelryInstance =
Instantiate(prefab, parentTransform);

jewelryInstance.GetComponent<RectTransform>().anch
oredPosition = positions[randomIndex];    //
Устанавливаем позицию
            }

            Destroy(gameObject);
        }
    }
}
```

Файл LoadingScreen1.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class LoadingScreen1 : MonoBehaviour
{
    public Slider progressBar;
    public string sceneToLoad;

    private void Start()
    {
        StartCoroutine(LoadSceneAsync() );

    }
    IEnumerator LoadSceneAsync()
    {
        AsyncOperation operation =
SceneManager.LoadSceneAsync( sceneToLoad );
        while (!operation.isDone)
        {
            float progress = Mathf.Clamp01(
operation.progress/0.9f);
            progressBar.value = progress;
            yield return null;
        }
    }
}
```

Файл MusicManager.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MusicManager : MonoBehaviour
{
    public AudioClip[] musicClips;
    public AudioSource audioSource;
    private void Awake()
    {
        if (FindObjectsOfType<MusicManager>().Length >
1)
        {
            Destroy(gameObject);
            return;
        }
        DontDestroyOnLoad(gameObject);
        audioSource =
gameObject.AddComponent<AudioSource>();
        audioSource.loop = true;
        PlayMusic(0);
    }
    public void PlayMusic(int index)
    {
        if (index < musicClips.Length)
        {
            audioSource.clip = musicClips[index];
            audioSource.Play();
        }
    }
```

```
    public void SetVolume(float volume)
    {
        audioSource=GetComponent<AudioSource>();//
        audioSource.volume = volume;
    }

    public void TogglePause()
    {
        if (audioSource.isPlaying) audioSource.Pause();
            else audioSource.UnPause();
    }
}
```

Файл MusicSelectionUI.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class MusicSelectionUI : MonoBehaviour
{
    private MusicManager musicManager;

    private void Start()
    {
        musicManager =
FindObjectOfType<MusicManager>();
    }
    public void SelectMusic(int index)
    {
        if (musicManager != null)
        {
            musicManager.PlayMusic(index);
        }
    }
    public void TogglePause()
    {
        if (musicManager != null)
        {
            musicManager.TogglePause();
        }
    }
}
```

Файл Oto_rang.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class Oto_rang : MonoBehaviour
{
    public GameObject[] images;
    public Text rang;
    void Update()
    {
        string[] stroka = File.ReadAllLines("ранг.txt");

        if (int.Parse(stroka[0]) > 0)
        {
            rang.gameObject.SetActive(true);
        }
        if
((int.Parse(stroka[0])>0)&&(int.Parse(stroka[0])<=3))
        {
            rang.text = "F";
        }
        if ((int.Parse(stroka[0]) >= 5) &&
(int.Parse(stroka[0]) <= 9))
        {
            rang.text = "E";
        }
        if ((int.Parse(stroka[0]) >= 12) &&
(int.Parse(stroka[0]) <= 18))
        {
            rang.text = "D";
        }
        if ((int.Parse(stroka[0]) >= 22) &&
(int.Parse(stroka[0]) <= 30))
        {
            rang.text = "C";
        }
        if ((int.Parse(stroka[0]) >= 35) &&
(int.Parse(stroka[0]) <= 45))
        {
            rang.text = "D";
        }
        if ((int.Parse(stroka[0]) >= 51) &&
(int.Parse(stroka[0]) <= 63))
        {
            rang.text = "A";
        }
        if ((int.Parse(stroka[0]) >=70) &&
(int.Parse(stroka[0]) <= 84))
        {
            rang.text = "S";
        }
        if ((int.Parse(stroka[0]) == 1)|| (int.Parse(stroka[0])
== 5) || (int.Parse(stroka[0]) == 6) || ((int.Parse(stroka[0])
>= 12) &&(int.Parse(stroka[0]) <= 14)) ||
((int.Parse(stroka[0]) >= 22) && (int.Parse(stroka[0]) <=
25)) || ((int.Parse(stroka[0]) >= 35) &&
(int.Parse(stroka[0]) <= 39)) || ((int.Parse(stroka[0]) >=
51) && (int.Parse(stroka[0]) <= 56)) ||
((int.Parse(stroka[0]) >= 70) && (int.Parse(stroka[0]) <=
76)))
        {
            images[0].gameObject.SetActive(true);
        }
        if ((int.Parse(stroka[0]) == 2) || (int.Parse(stroka[0])
== 7) || (int.Parse(stroka[0]) == 8) || ((int.Parse(stroka[0])
>= 15) && (int.Parse(stroka[0]) <= 17)) ||
((int.Parse(stroka[0]) >= 26) && (int.Parse(stroka[0]) <=
29)) || ((int.Parse(stroka[0]) >= 40) &&
(int.Parse(stroka[0]) <= 44)) || ((int.Parse(stroka[0]) >=
57) && (int.Parse(stroka[0]) <= 62)) ||
((int.Parse(stroka[0]) >= 77) && (int.Parse(stroka[0]) <=
83)))
        {
            images[0].gameObject.SetActive(true);
            images[1].gameObject.SetActive(true);
        }
        if ((int.Parse(stroka[0]) == 3) || (int.Parse(stroka[0])
== 4) || ((int.Parse(stroka[0]) >= 9) &&
```

```
(int.Parse(stroka[0]) <= 11)) || ((int.Parse(stroka[0]) >=
18) && (int.Parse(stroka[0]) <= 21)) ||
((int.Parse(stroka[0]) >= 30) && (int.Parse(stroka[0]) <=
34)) || ((int.Parse(stroka[0]) >= 45) &&
(int.Parse(stroka[0]) <= 50)) || ((int.Parse(stroka[0]) >=
63) && (int.Parse(stroka[0]) <= 69)) ||
(int.Parse(stroka[0]) >= 84))
        {
            images[0].gameObject.SetActive(true);
            images[1].gameObject.SetActive(true);
            images[2].gameObject.SetActive(true);
        }

    }
  }
```

Файл PausMenu.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using UnityEngine.UI;
  using System.IO;

  public class PausMenu : MonoBehaviour
  {
     public Image imgObg;
     public Sprite spriteImage1;
     public Sprite spriteImage2;
     public Button karta;
     public   bool gameIsPause = false;
     public GameObject pauseMenuUI;
     public void Start()
     {
        string[] stroka = File.ReadAllLines("карта.txt");
        if (stroka[0] == "0")
        {
           imgObg.sprite = spriteImage1;
        }
        else
        {
           imgObg.sprite = spriteImage2;
           karta.gameObject.SetActive(true);
        }
        }
     public void Button_menu()
     {
          Pause();
     }
     public void Button_resum()
     {
        Resume();
     }

     void Resume()
     {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        gameIsPause = false;
     }
     void Pause()
     {
      pauseMenuUI.SetActive(true);
        Time.timeScale = 0f;
        gameIsPause=true;
     }
  }
```

Файл playerControler.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using System.Threading.Tasks;
  public class playerControler : MonoBehaviour
  {
     public float speed;
     public float jumpForce;
     private float moveInput;
     private Rigidbody2D rb;
     private bool facingRight=true;
     private bool isGrounded;
     public Transform freePos;
     public float checkRadius;
     public LayerMask whatIsGround;
     private Animator anim;
     private void Start()
     {
        anim = GetComponent<Animator>();
        rb = GetComponent<Rigidbody2D>();
     }
     private void FixedUpdate()
     {
        moveInput = Input.GetAxis("Horizontal");
        rb.velocity = new Vector2(moveInput*speed,
  rb.velocity.y);

        if (facingRight==false  && moveInput>0 )
        { Flip(); }
        else if (facingRight==true && moveInput<0 ) {
  Flip(); }

        if(moveInput==0 ) {
           anim.SetBool("isRunning", false);
        }
        else
        {
           anim.SetBool("isRunning", true);
        }
     }
     private void Update()
     {
        isGrounded =
  Physics2D.OverlapCircle(freePos.position, checkRadius,
  whatIsGround);
        if (isGrounded == true &&
  ((Input.GetKeyDown(KeyCode.Space)) ||
  (Input.GetKeyDown(KeyCode.UpArrow)) ||
  (Input.GetKeyDown(KeyCode.W))))
        {
           rb.velocity = Vector2.up * jumpForce;
           anim.SetTrigger("taceOf");
        }
        if (isGrounded==true)
        {
           anim.SetBool("isJumping",false);
        }
        else
```

```
        {
            anim.SetBool("isJumping", true);
        }

        if (isGrounded == true &&
  ((Input.GetKeyDown(KeyCode.DownArrow) ||
  (Input.GetKeyDown(KeyCode.S)))))
        {
            anim.SetBool("isPrised", true);
        }
        else { anim.SetBool("isPrised", false); }

    }
    void Flip()
    {
        facingRight=!facingRight;
        Vector3 Scaler = transform.localScale;
        Scaler.x *= -1;
        transform.localScale = Scaler;
    }
  }
```

Файл Rang.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using System.IO;
  using UnityEngine.UI;
  using UnityEngine.UIElements;

  public class Rang : MonoBehaviour
    {
    public GameObject pauseMenuUI;
    public Text text1;
    public int heal;
    public bool b;

    public void Start()
    {
        heal = 10;
        b=true;
    }
    void Update()
    {
        if (((FindObjectOfType<нр>().hp <= 0) || (heal ==
  0))&&(b==true))
        {
            b = false;
            StartCoroutine(HandleGameOver());
        }
    }
    private IEnumerator HandleGameOver()
    {
        // Ждем  секунд перед остановкой времени и
  отображением меню
        yield return new WaitForSeconds(1f);

        // Останавливаем время
        Time.timeScale = 0f;
        string[] stroka = File.ReadAllLines("ранг.txt");

        if (FindObjectOfType<нр>().hp < 2)
        {
            text1.text = "F";
            stroka[0] = "1";
        }
        else
        {
            text1.text = "E";
            stroka[0] = "5";
        }
        using (StreamWriter f = new
    StreamWriter("ранг.txt", false))
        {
            for (int k = 0; k < stroka.Length; k++)
            {
                f.WriteLine(stroka[k]);
            }
        }
        // Отображаем меню паузы
        pauseMenuUI.SetActive(true);
    }
}
```

Файл Restart_game.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using System.IO;
  using UnityEngine;
  using UnityEngine.UI;
  public class Restart_game : MonoBehaviour
  {
    public GameObject endddd;
    public void restart_game()
    {
        string[] stroka3 = File.ReadAllLines("время
  скрытия.txt");//
        string[] stroka4= File.ReadAllLines("выполнение
  зд.txt");//
        string[] stroka6 = File.ReadAllLines("зелья.txt");//
        string[] stroka7 =
  File.ReadAllLines("зелья_кнопки.txt");//
        string[] stroka8 =
  File.ReadAllLines("использовать.txt");//
        string[] stroka9 = File.ReadAllLines("карта.txt");//
        string[] stroka10 =
  File.ReadAllLines("количество.txt");//
        string[] stroka11 =
  File.ReadAllLines("крафт.txt");//
        string[] stroka12 = File.ReadAllLines("Купля-
  прдажа.txt");//
        string[] stroka13 =
  File.ReadAllLines("манеты.txt");//
        string[] stroka14 = File.ReadAllLines("номер
  принятого задания.txt");//
        string[] stroka15 = File.ReadAllLines("Номер
  сцены для муз.txt");//
        string[] stroka16 = File.ReadAllLines("Номер
  сцены.txt");//
        string[] stroka17 =
  File.ReadAllLines("посещённые локации.txt");//
        string[] stroka18 =
  File.ReadAllLines("предметы.txt");//
        string[] stroka19 = File.ReadAllLines("ранг.txt");//
```

```
        string[] stroka20 =
File.ReadAllLines("растения.txt");//
        File.WriteAllText("positions.txt", string.Empty);
        File.WriteAllText("positions_kotel.txt",
string.Empty);
        File.WriteAllText("positions_kotel_krist.txt",
string.Empty);

        StreamWriter f3 = new StreamWriter("время
скрытия.txt", false);
        for (int k = 0; k < stroka3.Length; k++)
        {
            stroka3[k] = "03.05.2025 17:41:54";
            f3.WriteLine(stroka3[k]);
        }
        f3.Close();
        StreamWriter f4 = new StreamWriter("выполнение
зд.txt", false);
        for (int k = 0; k < stroka4.Length; k++)
        {
            if ((k>=0)&& (k <6))
            { stroka4[k] = 0.ToString(); }
            else
            { stroka4[k] = 1.ToString(); }
            f4.WriteLine(stroka4[k]);
        }
        f4.Close();
        StreamWriter f6 = new StreamWriter("зелья.txt",
false);
        for (int k = 0; k < stroka6.Length; k++)
        {
            stroka6[k] = 0.ToString();
            f6.WriteLine(stroka6[k]);
        }
        f6.Close();
        StreamWriter f7 = new
StreamWriter("зелья_кнопки.txt", false);
        for (int k = 0; k < stroka7.Length; k++)
        {
            stroka7[k] = 0.ToString();
            f7.WriteLine(stroka7[k]);
        }
        f7.Close();
        StreamWriter f8 = new
StreamWriter("использовать.txt", false);
        for (int k = 0; k < stroka8.Length; k++)
        {
            stroka8[k] = 0.ToString();
            f8.WriteLine(stroka8[k]);
        }
        f8.Close();
        StreamWriter f9 = new StreamWriter("карта.txt",
false);
        for (int k = 0; k < stroka9.Length; k++)
        {
            stroka9[k] = 0.ToString();
            f9.WriteLine(stroka9[k]);
        }
        f9.Close();

        StreamWriter f10 = new
StreamWriter("количество.txt", false);
        for (int k = 0; k < stroka10.Length; k++)
        {
            stroka10[k] = 0.ToString();
            f10.WriteLine(stroka10[k]);
        }
        f10.Close();
        StreamWriter f11 = new StreamWriter("крафт.txt",
false);
        for (int k = 0; k < stroka11.Length; k++)
        {
            stroka11[k] = 0.ToString();
            f11.WriteLine(stroka11[k]);
        }
        f11.Close();
        StreamWriter f12 = new StreamWriter("Купля-
прдажа.txt", false);
        for (int k = 0; k < stroka12.Length; k++)
        {
            stroka12[k] = 0.ToString();
            f12.WriteLine(stroka12[k]);
        }
        f12.Close();
        StreamWriter f13= new
StreamWriter("манеты.txt", false);
        for (int k = 0; k < stroka13.Length; k++)
        {
            stroka13[k] = 0.ToString();
            f13.WriteLine(stroka13[k]);
        }
        f13.Close();
        StreamWriter f14 = new StreamWriter("номер
принятого задания.txt", false);
        for (int k = 0; k < stroka14.Length; k++)
        {
            stroka14[k] = 0.ToString();
            f14.WriteLine(stroka14[k]);
        }
        f14.Close();
        StreamWriter f15 = new StreamWriter("Номер
сцены для муз.txt", false);
        for (int k = 0; k < stroka15.Length; k++)
        {
            stroka15[k] = 0.ToString();
            f15.WriteLine(stroka15[k]);
        }
        f15.Close();
        StreamWriter f16 = new StreamWriter("Номер
сцены.txt", false);
        for (int k = 0; k < stroka16.Length; k++)
        {
            stroka16[k] = 0.ToString();
            f16.WriteLine(stroka16[k]);
        }
        f16.Close();
        StreamWriter f17 = new
StreamWriter("посещённые локации.txt", false);
        for (int k = 0; k < stroka17.Length; k++)
        {
            stroka17[k] = 0.ToString();
            f17.WriteLine(stroka17[k]);
        }
        f17.Close();
```

```
        StreamWriter f18 = new
  StreamWriter("предметы.txt", false);
        for (int k = 0; k < stroka18.Length; k++)
        {
           stroka18[k] = 0.ToString();
           f18.WriteLine(stroka18[k]);
        }
        f18.Close();
        StreamWriter f19 = new StreamWriter("ранг.txt",
  false);
        for (int k = 0; k < stroka19.Length; k++)
        {
           stroka19[k] = 0.ToString();
           f19.WriteLine(stroka19[k]);
        }
        f19.Close();
        StreamWriter f20 = new
  StreamWriter("растения.txt", false);
        for (int k = 0; k < stroka20.Length; k++)
        {
           stroka20[k] = 0.ToString();
           f20.WriteLine(stroka20[k]);
        }
        f20.Close();

        Time.timeScale = 1f;
        if (endddd.gameObject != null)
        {
           endddd.gameObject.SetActive(false);
        }
     }
  }
```

Файл sborrastenii.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using UnityEngine.UI;
  using System.IO;
  using System;

  public class sborrastenii : MonoBehaviour
  {
     public Button[] bt;

     void Update()
     {
        string[] vremia = File.ReadAllLines("время
  скрытия.txt");
        for (int i = 0; i < vremia.Length; i++)
        {
           DateTime d1 = DateTime.Now;
           TimeSpan raznosti = d1 -
  DateTime.Parse(vremia[i]);
           if (raznosti.TotalMinutes > 15)
           {
              bt[i].gameObject.SetActive(true);
           }
        }
     }
     public void ButtonClik(Button clickedButton)
     {
        string buttonName = clickedButton.name;//
  проверка какая из одинаковых кнопок нажата
        string[] stroka = File.ReadAllLines("растения.txt");

        int x = 0;
        for (int j = 0; j < stroka.Length; j++)
        {
           if ((buttonName == bt[0 + x].name) ||
  (buttonName == bt[1 + x].name))
           {
              stroka[j] = (int.Parse(stroka[j]) + 1).ToString();
           }
           x += 2;
        }
        StreamWriter f = new StreamWriter("растения.txt",
  false);
        for (int i = 0; i < stroka.Length; i++)
        {
           f.WriteLine(stroka[i]);
        }
        f.Close();//счёт собранных растений
        string[] vremia = File.ReadAllLines("время
  скрытия.txt");
        DateTime d = DateTime.Now;

        for (int k=0; k < vremia.Length; k++)
        {
           if (buttonName == bt[k].name)
           {
              vremia[k] = d.ToString();
              bt[k].gameObject.SetActive(false); ;

           }
        }
        StreamWriter f1 = new StreamWriter("время
  скрытия.txt", false);
        for (int i = 0; i < vremia.Length; i++)
        {
           f1.WriteLine(vremia[i]);
        }
        f1.Close();
     }
  }
```

Файл Spravka.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using System.Diagnostics;
  using UnityEngine;

  public class Spravka : MonoBehaviour
  {
     public string chmFilePath;

     public void OpenCHM()
     {
        Process.Start(chmFilePath);
     }
  }
```

Файл vosvr_vr.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
```

```
public class vosvr_vr : MonoBehaviour
{
    public void Button_vosvr()
    {
        Time.timeScale = 1f;
    }
}
```

Файл zadanieposborules.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;
public class zadanieposborules : MonoBehaviour
{
    public GameObject z1;
    public GameObject z2;
    public Text[] text;
    public Image imgObg;
    public Sprite spriteImage1;
    public Sprite spriteImage2;
    public Button karta;
    public GameObject pobeda;
    void Start()
    {
        string[] stroka = File.ReadAllLines("номер
принятого задания.txt");
        if (int.Parse(stroka[0]) == 1)
        {
            z1.gameObject.SetActive(true);
        }
        if (int.Parse(stroka[0]) == 2)
        {
            z2.gameObject.SetActive(true);
        }
    }
    void Update()
    {
        string[] stroka = File.ReadAllLines("номер
принятого задания.txt");
        string[] stroka1 =
File.ReadAllLines("растения.txt");
        if (int.Parse(stroka[0]) == 1)
        {
            text[0].text = stroka1[8] + "/2";
            text[1].text = stroka1[9] + "/2";
        }
        if (int.Parse(stroka[0]) == 2)
        {
            text[2].text = stroka1[7] + "/2";
            text[3].text = stroka1[12] + "/2";
        }
        if ((int.Parse(stroka[0]) == 1) &&
(int.Parse(stroka1[8]) >= 2) && (int.Parse(stroka1[9])
>= 2))
        {
            stroka1[8] = (int.Parse(stroka1[8]) -
2).ToString();
            stroka1[9] = (int.Parse(stroka1[9]) -
2).ToString();
            StreamWriter f1 = new
StreamWriter("растения.txt", false);
            for (int k = 0; k < stroka1.Length; k++)
            {
                f1.WriteLine(stroka1[k]);
            }
            f1.Close();

            stroka[0] = "0";
            StreamWriter f = new StreamWriter("номер
принятого задания.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f.WriteLine(stroka[k]);
            }
            f.Close();
            string[] stroka2 =
File.ReadAllLines("манеты.txt");
            stroka2[0] = (int.Parse(stroka2[0]) +
100).ToString();

            StreamWriter f2 = new
StreamWriter("манеты.txt", false);
            for (int k = 0; k < stroka2.Length; k++)
            {
                f2.WriteLine(stroka2[k]);
            }
            f2.Close();

            string[] stroka3 =
File.ReadAllLines("выполнение зд.txt");
            stroka3[0] = (int.Parse(stroka3[0]) +
1).ToString();
            StreamWriter f3 = new
StreamWriter("выполнение зд.txt", false);
            for (int k = 0; k < stroka3.Length; k++)
            {
                f3.WriteLine(stroka3[k]);
            }
            f3.Close();
            string[] stroka4 = File.ReadAllLines("ранг.txt");
            stroka4[0] = (int.Parse(stroka4[0]) +
1).ToString();

            StreamWriter f4 = new StreamWriter("ранг.txt",
false);
            for (int k = 0; k < stroka4.Length; k++)
            {
                f4.WriteLine(stroka4[k]);
            }
            f4.Close();
            pobeda.gameObject.SetActive(true);

            string[] stroka5 = File.ReadAllLines("карта.txt");
            if (stroka5[0] == "0")
            {
                imgObg.sprite = spriteImage1;
            }
            else
            {
                imgObg.sprite = spriteImage2;
                karta.gameObject.SetActive(true);
```

```
            }
             StartCoroutine(stop());
        }
        if ((int.Parse(stroka[0]) == 2) &&
(int.Parse(stroka1[7]) >= 2) && (int.Parse(stroka1[12])
>= 2))
        {
            stroka1[7] = (int.Parse(stroka1[7]) -
2).ToString();
            stroka1[12] = (int.Parse(stroka1[12]) -
2).ToString();

            StreamWriter f1 = new
StreamWriter("растения.txt", false);
            for (int k = 0; k < stroka1.Length; k++)
            {
                f1.WriteLine(stroka1[k]);
            }
            f1.Close();

            stroka[0] = 0.ToString();
            StreamWriter f = new StreamWriter("номер
принятого задания.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f.WriteLine(stroka[k]);
            }
            f.Close();
            string[] stroka2 =
File.ReadAllLines("манеты.txt");
            stroka2[0] = (int.Parse(stroka2[0]) +
100).ToString();

            StreamWriter f2 = new
StreamWriter("манеты.txt", false);
            for (int k = 0; k < stroka2.Length; k++)
            {
                f2.WriteLine(stroka2[k]);
            }
            f2.Close();

            string[] stroka3 =
File.ReadAllLines("выполнение зд.txt");
            stroka3[2] = (int.Parse(stroka3[2]) +
1).ToString();

            StreamWriter f3 = new
StreamWriter("выполнение зд.txt", false);
            for (int k = 0; k < stroka3.Length; k++)
            {
                f3.WriteLine(stroka3[k]);
            }
            f3.Close();

            string[] stroka4 = File.ReadAllLines("ранг.txt");
            stroka4[0] = (int.Parse(stroka4[0]) +
2).ToString();

            StreamWriter f4 = new StreamWriter("ранг.txt",
false);
            for (int k = 0; k < stroka4.Length; k++)
            {
                f4.WriteLine(stroka4[k]);
            }
            f4.Close();
            pobeda.gameObject.SetActive(true);

            string[] stroka5 = File.ReadAllLines("карта.txt");
            if (stroka5[0] == "0")
            {
                imgObg.sprite = spriteImage1;
            }
            else
            {
                imgObg.sprite = spriteImage2;
                karta.gameObject.SetActive(true);
            }
            StartCoroutine(stop());

        }
    }
    private IEnumerator stop()
    {
        yield return new WaitForSeconds(3f);
        Time.timeScale = 0f;
    }

}
```

Файл zadani_sell.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class zadani_sell : MonoBehaviour
{
    public GameObject z1;
    public GameObject z2;
    public Text[] text;

    public Image imgObg;
    public Sprite spriteImage1;
    public Sprite spriteImage2;
    public Sprite spriteImage3;
    public Sprite spriteImage4;
    public Button karta;

    public GameObject pobeda;

    public Переход perehod;

    void Start()
    {
        string[] stroka = File.ReadAllLines("номер
принятого задания.txt");

        if (int.Parse(stroka[0]) == 4)
        {
            z1.gameObject.SetActive(true);
            perehod.levelToLoad = 5;
        }
        if (int.Parse(stroka[0]) == 5)
```

```
        {
            z2.gameObject.SetActive(true);
            perehod.levelToLoad = 5;
        }
    }


    void Update()
    {
        string[] stroka = File.ReadAllLines("номер
принятого задания.txt");

        string[] stroka1 = File.ReadAllLines("зелья.txt");

        if (int.Parse(stroka[0]) == 4)
        {
            text[0].text = stroka1[1] + "/1";
        }

        if (int.Parse(stroka[0]) == 5)
        {
            text[1].text = stroka1[2] + "/1";
        }

        if ((int.Parse(stroka[0]) == 4) &&
(int.Parse(stroka1[1]) >= 1))
        {
            stroka1[1] = (int.Parse(stroka1[1]) -
1).ToString();

            StreamWriter f1 = new StreamWriter("зелья.txt",
false);
            for (int k = 0; k < stroka1.Length; k++)
            {
                f1.WriteLine(stroka1[k]);
            }
            f1.Close();

            stroka[0] = "0";
            StreamWriter f = new StreamWriter("номер
принятого задания.txt", false);
            for (int k = 0; k < stroka.Length; k++)
            {
                f.WriteLine(stroka[k]);
            }
            f.Close();
            string[] stroka2 =
File.ReadAllLines("манеты.txt");
            stroka2[0] = (int.Parse(stroka2[0]) +
225).ToString();

            StreamWriter f2 = new
StreamWriter("манеты.txt", false);
            for (int k = 0; k < stroka2.Length; k++)
            {
                f2.WriteLine(stroka2[k]);
            }
            f2.Close();

            string[] stroka3 =
File.ReadAllLines("выполнение зд.txt");
            stroka3[3] = (int.Parse(stroka3[3]) +
1).ToString();

            StreamWriter f3 = new
StreamWriter("выполнение зд.txt", false);
            for (int k = 0; k < stroka3.Length; k++)
            {
                f3.WriteLine(stroka3[k]);
            }
            f3.Close();

            string[] stroka4 = File.ReadAllLines("ранг.txt");
            stroka4[0] = (int.Parse(stroka4[0]) +
2).ToString();

            StreamWriter f4 = new StreamWriter("ранг.txt",
false);
            for (int k = 0; k < stroka4.Length; k++)
            {
                f4.WriteLine(stroka4[k]);
            }
            f4.Close();


            pobeda.gameObject.SetActive(true);

            string[] stroka5 = File.ReadAllLines("карта.txt");
            if (stroka5[0] == "0")
            {
                imgObg.sprite = spriteImage1;
            }
            else
            {
                imgObg.sprite = spriteImage2;
                karta.gameObject.SetActive(true);
            }

             StartCoroutine(stop());

        }


        if ((int.Parse(stroka[0]) == 5) &&
(int.Parse(stroka1[2]) >= 1) )
        {


            stroka1[2] = (int.Parse(stroka1[2]) -
1).ToString();


            StreamWriter f1 = new StreamWriter("зелья.txt",
false);
            for (int k = 0; k < stroka1.Length; k++)
            {
                f1.WriteLine(stroka1[k]);
            }
            f1.Close();

            stroka[0] = 0.ToString();
            StreamWriter f = new StreamWriter("номер
принятого задания.txt", false);
```

```
        for (int k = 0; k < stroka.Length; k++)
        {
            f.WriteLine(stroka[k]);
        }
        f.Close();
        string[] stroka2 =
File.ReadAllLines("манеты.txt");
        stroka2[0] = (int.Parse(stroka2[0]) +
200).ToString();

        StreamWriter f2 = new
StreamWriter("манеты.txt", false);
        for (int k = 0; k < stroka2.Length; k++)
        {
            f2.WriteLine(stroka2[k]);
        }
        f2.Close();

        string[] stroka3 =
File.ReadAllLines("выполнение зд.txt");
        stroka3[4] = (int.Parse(stroka3[4]) +
1).ToString();

        StreamWriter f3 = new
StreamWriter("выполнение зд.txt", false);
        for (int k = 0; k < stroka3.Length; k++)
        {
            f3.WriteLine(stroka3[k]);
        }
        f3.Close();

        string[] stroka4 = File.ReadAllLines("ранг.txt");
        stroka4[0] = (int.Parse(stroka4[0]) +
2).ToString();

        StreamWriter f4 = new StreamWriter("ранг.txt",
false);
        for (int k = 0; k < stroka4.Length; k++)
        {
            f4.WriteLine(stroka4[k]);
        }
        f4.Close();
        pobeda.gameObject.SetActive(true);

        string[] stroka5 = File.ReadAllLines("карта.txt");
        if (stroka5[0] == "0")
        {
            imgObg.sprite = spriteImage3;
        }
        else
        {
            imgObg.sprite = spriteImage4;
            karta.gameObject.SetActive(true);
        }


        StartCoroutine(stop());

    }
}

private IEnumerator stop()
{
    yield return new WaitForSeconds(3f);
    Time.timeScale = 0f;
}

 public void start()
{
    Time.timeScale = 1f;
}

}
```

Файл zagruska.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class zagruska : MonoBehaviour
{

    private Animator anim;

    void Start()
    {

      StartCoroutine(gar1());

    }
    private IEnumerator gar1()
    {
        yield return new WaitForSeconds(7.8f);

        anim = GetComponent<Animator>();
        SceneManager.LoadScene(1);
        anim.SetTrigger("fade");

    }

}
```

Файл Переход.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Переход : MonoBehaviour
{
    private Animator anim;
    public int levelToLoad;
    private void Start()
    {
        anim = GetComponent<Animator>();
    }

    public void FadeToLevel()
    {
        anim.SetTrigger("fade");
    }
```

```
    public void OnFadeComplete()
    {
        SceneManager.LoadScene(levelToLoad);
    }

}
```

Файл враг.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using static UnityEngine.RuleTile.TilingRuleOutput;

  public class враг : MonoBehaviour
  {
    [SerializeField] private float x;
     [SerializeField] private float x1;
     [SerializeField] private float y;
     [SerializeField] private float moveSpeed = 2f;
     [SerializeField] private Vector2 zoneSize = new
  Vector2(5f, 5f);
     private UnityEngine.Transform player;
     [SerializeField] private bool playerInRange = false;
     private Vector3 initialPosition;
     private Animator animator; // Аниматор врага

     void Start()
     {
        player =
  GameObject.FindWithTag("Player").transform;
        initialPosition = transform.position;
        animator = GetComponent<Animator>(); //
  Получаем компонент Animator
     }

     void Update()
     {
        if (player.position.x > x && player.position.y < y
  && player.position.x < x1)
        {
           MoveTowardsPlayer();
        }
        else
        {
           ReturnToInitialPosition();
        }
     }

     void MoveTowardsPlayer()
     {
        Vector2 direction = (player.position -
  transform.position).normalized;
        Vector3 targetPosition = transform.position + new
  Vector3(direction.x * moveSpeed * Time.deltaTime, 0,
  0);

        if (IsWithinZone(targetPosition))
        {
           transform.position = targetPosition;
           UpdateAnimationAndRotation(direction); //
  Обновляем анимацию и поворот
        }
     }

     bool IsWithinZone(Vector3 position)
     {
        float halfWidth = zoneSize.x / 2;
        float halfHeight = zoneSize.y / 2;
        return position.x >= initialPosition.x - halfWidth
  && position.x <= initialPosition.x + halfWidth &&
              position.y >= initialPosition.y - halfHeight &&
  position.y <= initialPosition.y + halfHeight;
     }

     void ReturnToInitialPosition()
     {
        if (Vector3.Distance(transform.position,
  initialPosition) > 0.1f)
        {
           if (transform.position.x > initialPosition.x)
           {
              transform.localScale = new Vector3(1, 1, 1);
           }
           if (transform.position.x < initialPosition.x)
           {
              transform.localScale = new Vector3(-1, 1, 1);
           }

           transform.position =
  Vector3.MoveTowards(transform.position,
  initialPosition, moveSpeed * Time.deltaTime);

           if (transform.position == initialPosition)
           {
              animator.SetBool("isRunning", false); //
  Остановка анимации бега
           }
        }
     }

     private void UpdateAnimationAndRotation(Vector2
  direction)
     {
        animator.SetBool("isRunning", true); // Запуск
  анимации бега

        // Поворачиваем врага в сторону движения
        if (direction.x > 0)
        {
           transform.localScale = new Vector3(-1, 1, 1); //
  Поворот вправо
        }
        else if (direction.x < 0)
        {
           transform.localScale = new Vector3(1, 1, 1); //
  Поворот влево
        }
     }

     private void OnCollisionEnter2D(Collision2D
  collision)
     {
        if (collision.collider.CompareTag("Player"))
        {
```

```
            нр playerController =
    collision.collider.GetComponent<нр>();
            playerController.boy(); // Наносим урон игроку

            // Получаем Rigidbody2D игрока
            Rigidbody2D playerRigidbody =
    collision.collider.GetComponent<Rigidbody2D>();
            if (playerRigidbody != null)
            {
                // Вычисляем направление от врага к игроку
                Vector2 knockbackDirection =
    (collision.transform.position -
    transform.position).normalized;

                // Устанавливаем компонент Y для
    отбрасывания
                float knockbackX = knockbackDirection.x *
    400f; // Настройте силу отталкивания по вашему
    усмотрению
                float knockbackY =400f; // Сила
    отталкивания вверх

                // Применяем силу отталкивания
                playerRigidbody.AddForce(new
    Vector2(knockbackX, knockbackY),
    ForceMode2D.Impulse);
            }
        }
    }
```

Файл нр.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


public class нр : MonoBehaviour
{
    public int hp;
    public Image a;
    public Image b;

    public void boy()
    {
        hp--;
        if (hp == 1)
        { a.gameObject.SetActive(false); }
        if (hp == 0)
        { b.gameObject.SetActive(false); }
    }
}
```

Файл переходизмузыки.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using System.IO;
public class переходизмузыки : MonoBehaviour
{
    private Animator anim;

    private void Start()
    {
        anim = GetComponent<Animator>();
    }

    public void FadeToLevel()
    {
        anim.SetTrigger("fade");
    }

    public void OnFadeComplete()
    {

        //считывание номера сцены из файла
        StreamReader f = new StreamReader("Номер
    сцены для муз.txt");
        int n = int.Parse(f.ReadToEnd());
        f.Close();
        SceneManager.LoadScene(n);
    }

    public void OnFadeComplete1()
    {


        //считывание номера сцены из файла
        StreamReader f = new StreamReader("Номер
    сцены.txt");
        int n = int.Parse(f.ReadToEnd());
        f.Close();
        SceneManager.LoadScene(n);
    }

}
```

Файл переходизторговли.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using System.IO;
public class переходизторговли : MonoBehaviour
{
    private Animator anim;

    private void Start()
    {
        anim = GetComponent<Animator>();
    }

    public void FadeToLevel()
    {
        anim.SetTrigger("fade");
    }

    public void OnFadeComplete()
    {       //считывание номера сцены из файла
        StreamReader f = new StreamReader("Номер
    сцены.txt");
        int n = int.Parse(f.ReadToEnd());
```

```
        f.Close();
        SceneManager.LoadScene(n);
    }

}
```

Файл переходсзаписью.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
  using UnityEngine.SceneManagement;
  using System.IO;

  public class переходсзаписью : MonoBehaviour
  {
     private Animator anim;
     public int levelToLoad;
     private void Start()
     {
        anim = GetComponent<Animator>();
     }

     public void FadeToLevel()
     {
        anim.SetTrigger("fade");

        //получение индекса
        Scene currentScene
  =SceneManager.GetActiveScene();
        int sceneIndex =currentScene.buildIndex;
        StreamWriter f = new StreamWriter("Номер
  сцены.txt",false);
        f.WriteLine(sceneIndex.ToString() );
        f.Close();
     }

     public void FadeToLevel1()
     {
        anim.SetTrigger("fade");

        //получение индекса
        Scene currentScene =
  SceneManager.GetActiveScene();
        int sceneIndex = currentScene.buildIndex;
        StreamWriter f = new StreamWriter("Номер сцены
  для муз.txt", false);
        f.WriteLine(sceneIndex.ToString());
        f.Close();
     }

     public void OnFadeComplete()
     {

        SceneManager.LoadScene(levelToLoad);
     }

}
```

Файл расшифровка.cs:

```
  using System.Collections;
  using System.Collections.Generic;
  using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class расшифровка : MonoBehaviour
{
   public Text text1;
   public Text text2;

   public Image imgObg;
   public Sprite spriteImage1;
   public Sprite spriteImage2;

   public Image imgObg2;
   public Sprite spriteImage21;
   public Sprite spriteImage22;
   public Button karta;


   public GameObject pobeda;
   public GameObject porajenie;
   public void Gotovo()
   {
      if ((text1.text == "Великие начинания даже не
надо обдумывать, надо взяться да дело, иначе,
заметив трудность отступишь.") && (text2.text == "-
Гай Юлий Цезарь"))
      {
         string[] stroka1 =
File.ReadAllLines("манеты.txt");
         stroka1[0] = (int.Parse(stroka1[0]) +
100).ToString();

         StreamWriter f1 = new
StreamWriter("манеты.txt", false);
         for (int k = 0; k < stroka1.Length; k++)
         {
            f1.WriteLine(stroka1[k]);
         }
         f1.Close();

         string[] stroka2 =
File.ReadAllLines("выполнение зд.txt");
         stroka2[1] = (int.Parse(stroka2[1]) +
1).ToString();

         StreamWriter f2 = new
StreamWriter("выполнение зд.txt", false);
         for (int k = 0; k < stroka2.Length; k++)
         {
            f2.WriteLine(stroka2[k]);
         }
         f2.Close();

         string[] stroka3 = File.ReadAllLines("ранг.txt");
         stroka3[0] = (int.Parse(stroka3[0]) +
1).ToString();

         StreamWriter f3 = new StreamWriter("ранг.txt",
false);
         for (int k = 0; k < stroka3.Length; k++)
         {
            f3.WriteLine(stroka3[k]);
```

```
            }
            f3.Close();
            pobeda.SetActive(true);
            string[] stroka = File.ReadAllLines("карта.txt");
            if (stroka[0] == "0")
            {
                imgObg.sprite = spriteImage1;
            }
            else
            {
                imgObg.sprite = spriteImage2;
                karta.gameObject.SetActive(true);
            }
        }

        else {
            porajenie.SetActive(true);

            string[] stroka = File.ReadAllLines("карта.txt");
            if (stroka[0] == "0")
            {
                imgObg2.sprite = spriteImage21;
            }
            else
            {
                imgObg2.sprite = spriteImage22;
                karta.gameObject.SetActive(true);
            }

        }

        Time.timeScale = 0f;
    }

    public void zanovo()
    {
        porajenie.SetActive(false);
        karta.gameObject.SetActive(false);
        text1.text = "";
        text2.text = "";
        Time.timeScale = 1f;
    }
}
```

Приложение В

Диаграмма компонентов

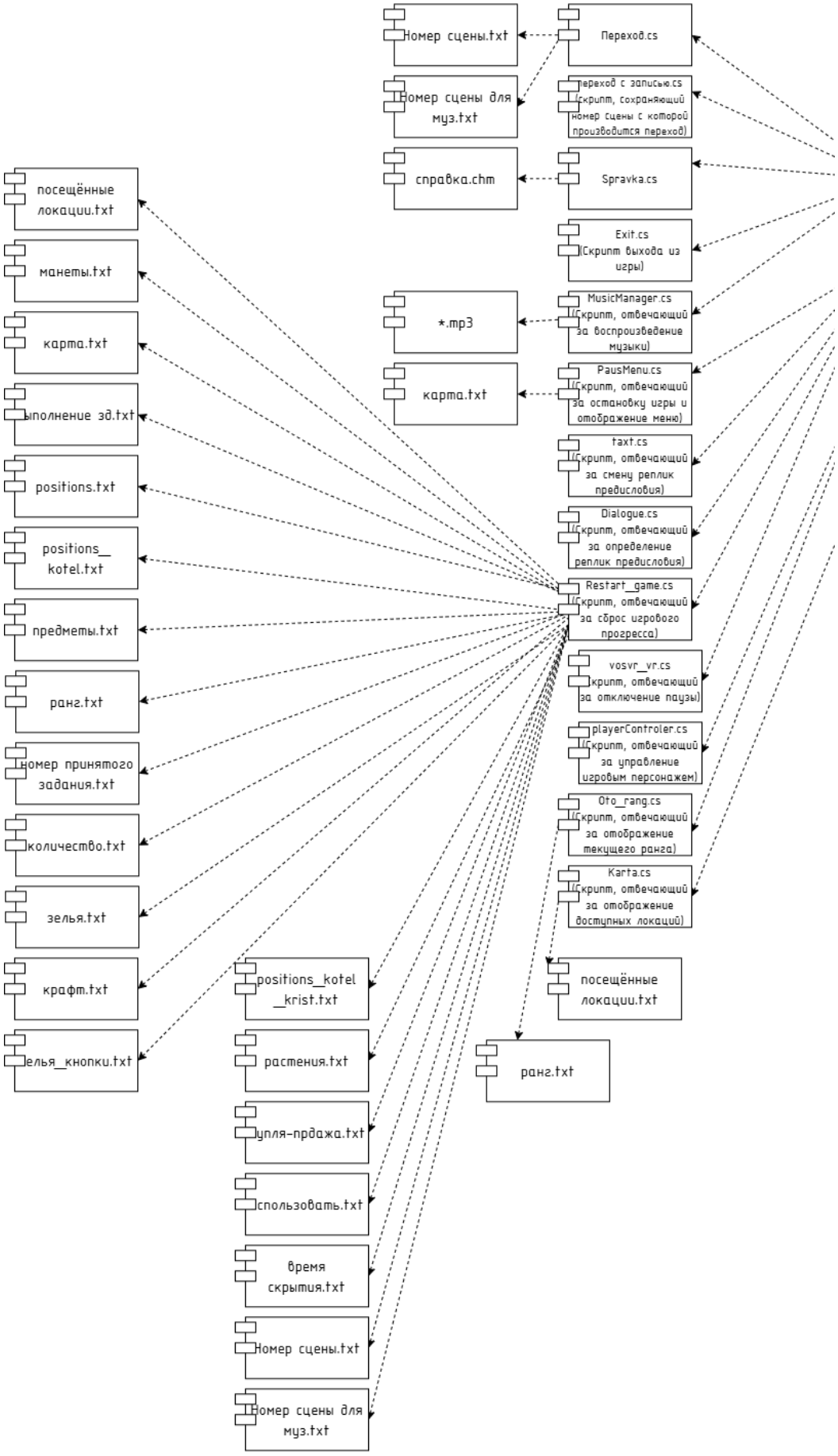

Рисунок В.1 – Диаграмма компонентов

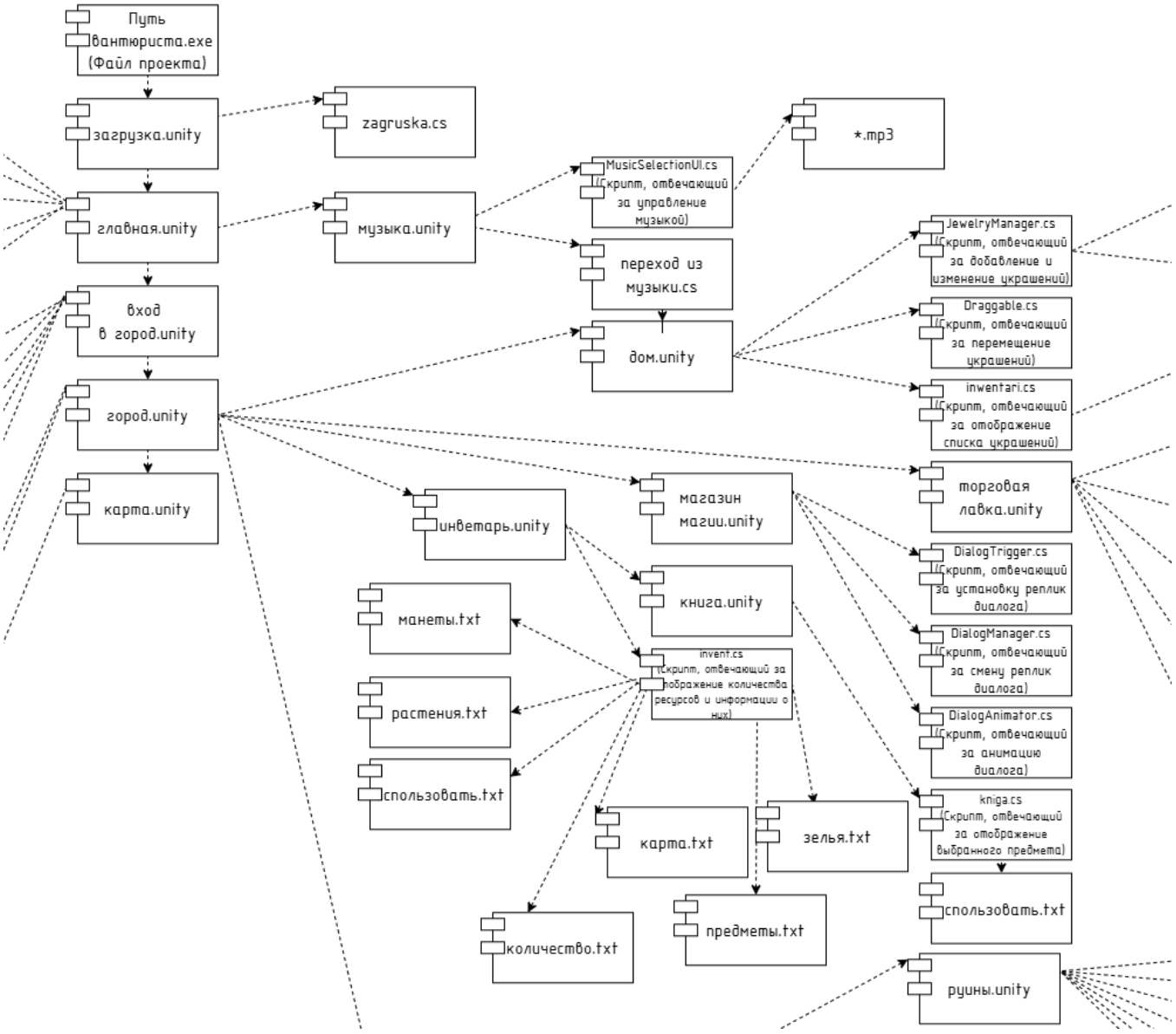

Рисунок В.2 – Продолжение диаграммы компонентов 1

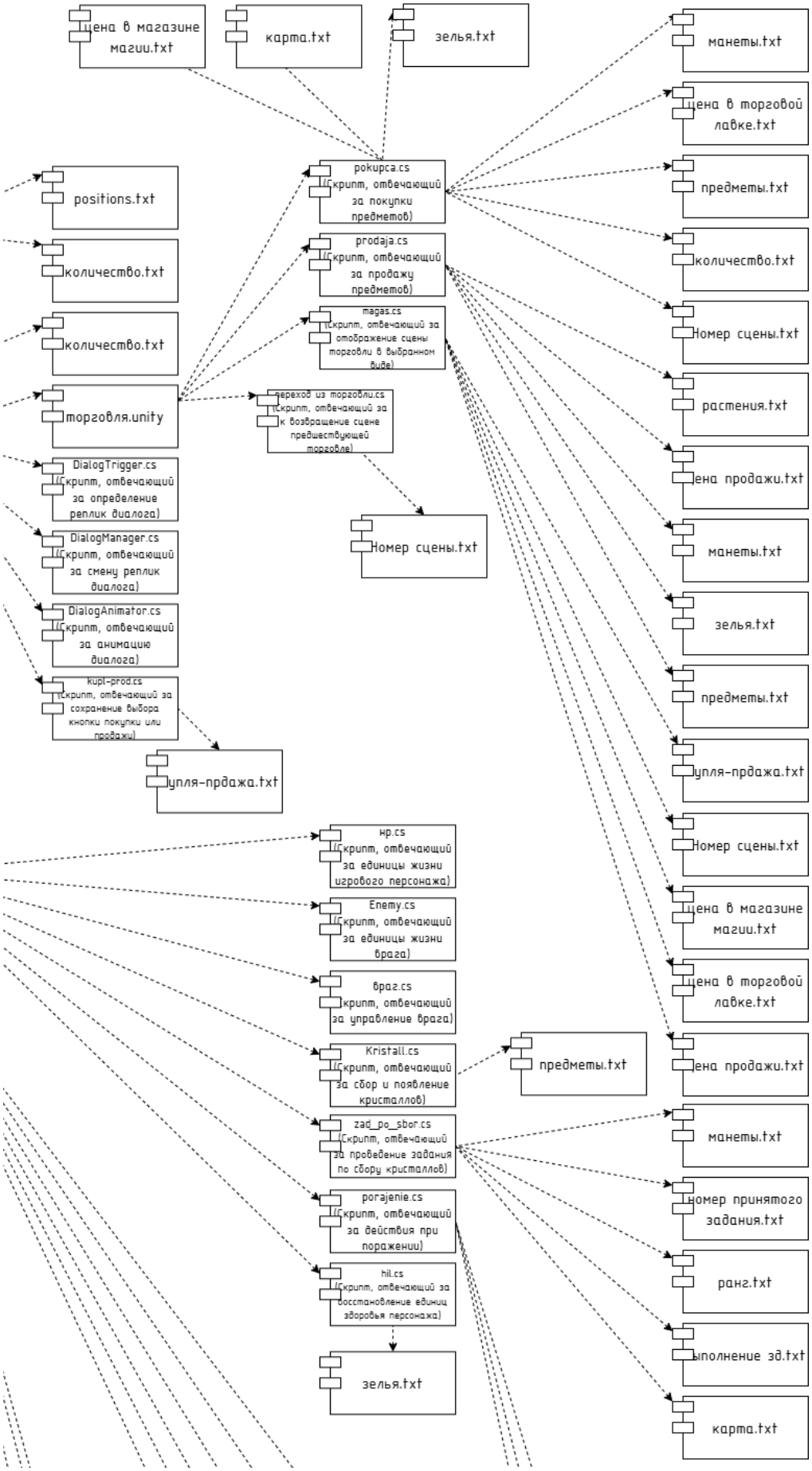

Рисунок В.3 – Продолжение диаграммы компонентов 2

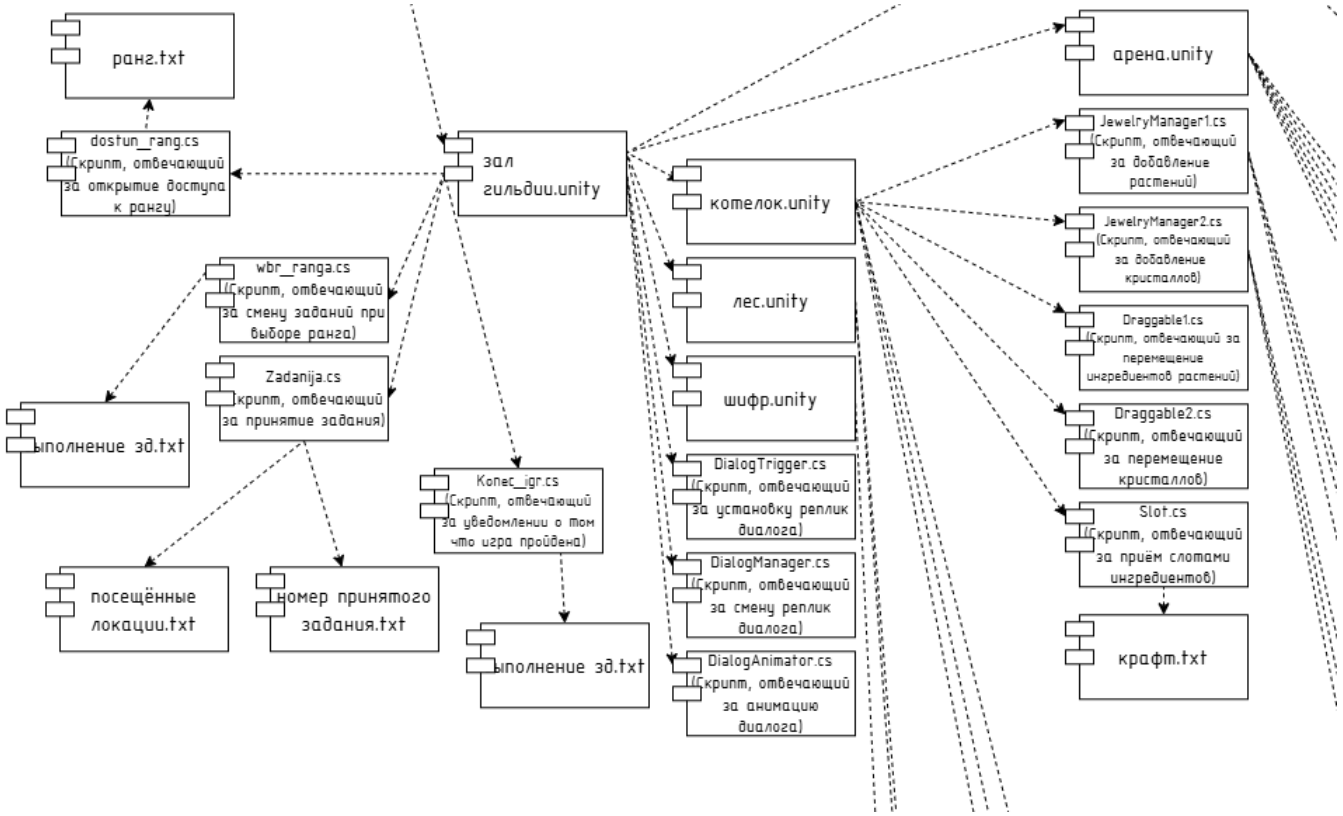

Рисунок В.4 – Продолжение диаграммы компонентов 3

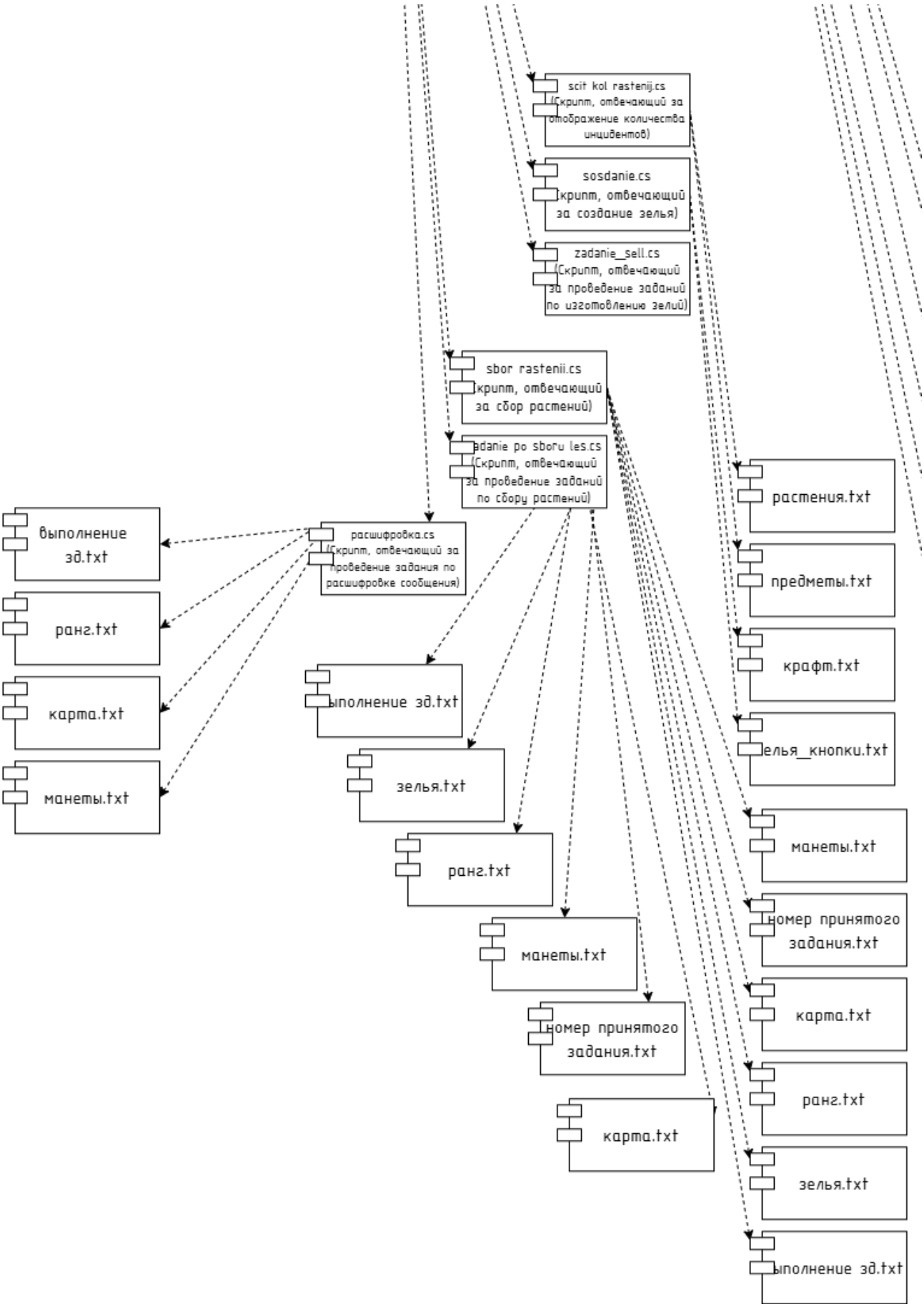

Рисунок В.5 – Продолжение диаграммы компонентов 4

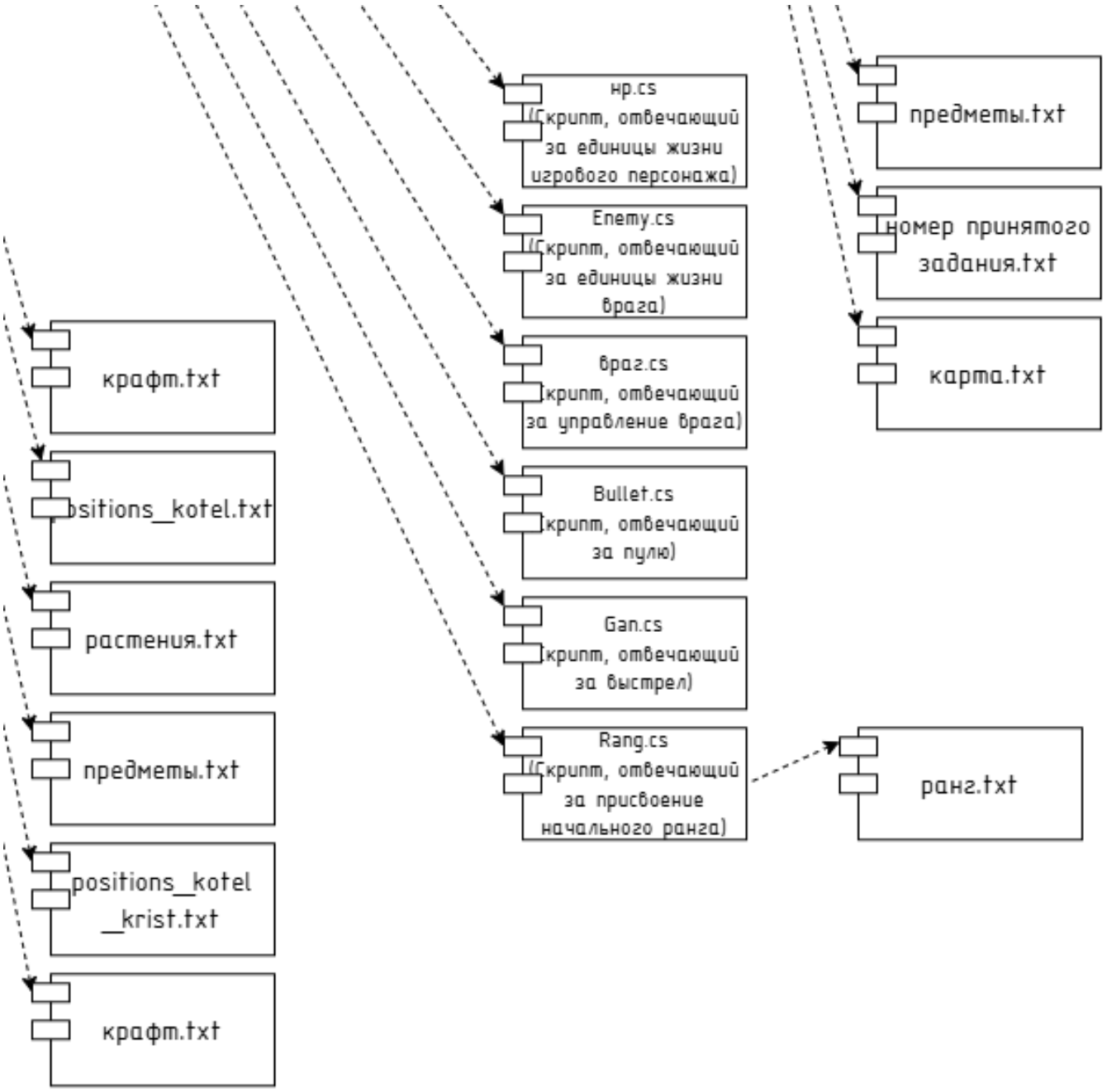

Рисунок В.6 – Продолжение диаграммы компонентов 5