Министерство образования и науки Челябинской области

государственное бюджетное профессиональное образовательное учреждение «Челябинский радиотехнический техникум»

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |  |
| --- | --- | --- |
| ДОПУСТИТЬ К ЗАЩИТЕ Заместитель директора техникума по УР  \_\_\_\_\_\_\_ Н.М. Иванова  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г. |  | Проект защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

**Курсовая работа**

**по МДК 01.01 «Разработка программных модулей**

**ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА  
ТЕКСТОВОЙ RPG «ЭКСПЕДИЦИЯ»**

*Пояснительная записка*

*ЧРТ.09.02.07.001.02 ПЗ*

|  |  |
| --- | --- |
|  | Разработал:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Темников  (подпись, дата) |
|  | Руководитель:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.Е. Ганеев  (подпись, дата) |

2024

АННОТАЦИЯ

Темников Сергей Алексеевич

группы ПР-365

Тема курсовой работы:

Проектирование и разработка текстовой RPG «Экспедиция»

г. Челябинск

ГБПОУ «Челябинский радиотехнический техникум» специальность 09.02.07 Информационные системы и программирование

2024 г.

Пояснительная записка к курсовой работе содержит 64 страницы, 1 таблицу, 5 рисунков, 2 приложения, 5 наименований источников литературы.

Объект исследования является текстовые RPG и почему они интересны людям. Предмет исследования устройство текстовых RPG. Цель исследования разработка текстовой RPG.

Был выполнен анализ предметной области, спроектирована и разработана текстовая RPG. В разделе разработка сайта строится функциональная модель текстовой RPG и происходит ее инициализация. В экономическом части рассчитывается стоимость разработки текстовой RPG. Сайт знакомств «Название» был разработан за семь дней и протестирован за один день.

ГБПОУ «Челябинский радиотехнический техникум»

|  |  |  |
| --- | --- | --- |
|  |  | Утверждаю:  Зам. директора по УР  \_\_\_\_\_\_\_\_\_\_ И.М. Иванова  «\_\_\_» \_\_\_\_\_\_\_\_\_2024 г. |

**Задание к КУРСОВОЙ работе**

Студенту группы **ПР-365** специальности **09.02.07 «Информационные системы и программирование»**

**Темников Сергею Алексеевичу**

Тема курсовой работы: **Проектирование и разработка текстовой RPG «Экспедиция»**

Исходные данные к курсовой работе:

1 Программное обеспечение, необходимое для проектирования и разработки приложения (операционная система Microsoft Windows 10, настольный компьютер с доступом в Интернет, браузер Google Chrome, Visual Studio Community).

2 Техническое задание на разработку сайта

3 Рекомендуемая литература

3.1 ГОСТ 2.105–2019. ЕСКД. Общие требования к текстовым документам [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru. – Загл. с экрана.

3.2 ГОСТ 19.002–90. ЕСТП. Схемы алгоритмов и программ. Правила выполнения [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru. – Загл. с экрана.

3.3 ГОСТ 19.201-78 Технические задание. Требования к содержанию и оформлению // Режим доступа: http://docs.cntd.ru/document/gost-19-201-78. – Загл. с экрана . – Дата доступа: 19.09.2022.

3.4 Гвоздева, В. А. Введение в специальность программиста : учеб. для сред. проф. образования / В. А. Гвоздева. – Москва : ИНФРА-М ; ФОРУМ, 2013. – 208 с. : ил. – (Профессиональное образование).

3.5 Гниденко, И. Г. Технология разработки программного обеспечения : учеб. пособие для сред. проф. образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2024. — 248 с. — (Профессиональное образование). — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/539215 .

3.6 Голицына, О. Л. Программное обеспечение : учеб. пособие для сред. проф. образования / О. Л. Голицына, Т. Л. Партыка, И. И. Попов. – Москва : Форум, 2013. – 448 с. : ил. – (Профессиональное образование).

3.7 Зубкова, Т. М. Технология разработки программного обеспечения / Т. М. Зубкова. — Санкт-Петербург : Лань, 2023. — 252 с. —Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/276419

3.8 Казанский, А. А. Программирование на Visual C# : учеб. пособие для сред. проф. образования / А. А. Казанский. — Москва : Издательство Юрайт, 2024. — 192 с. — (Профессиональное образование). — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/538155

Задание выполняется студентом в следующем объеме:

**СОДЕРЖАНИЕ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ**

Введение (Актуальность выбранной темы, практическая значимость, объект исследования, предмет исследования, постановка цели и задач, круг рассматриваемых проблем)

1 Анализ предметной области и формирование требований к разрабатываемому приложению

1.1 Назначение и цели разработки приложения

1.2 Формирование требований к приложению

1.2.1 Требования к дизайну приложения

1.2.2 Требования к структуре и функционированию приложения

1.2.3 Технологические требования

1.2.4 Требования к безопасности приложения

2 Выбор средств реализации

2.1 Обзор современных языков программирования для разработки (технологии разработки приложений)

2.2 Обзор современных СУБД

2.3 Выбор и обоснование средств реализации в соответствии с предъявленными требованиями

3 Состав и содержание работ по созданию системы

4 Разработка программного продукта

4.1 Разработка схемы базы данных приложения

4.2 Разработка пользовательского интерфейса

4.3 Разработка серверной части проекта

5. Тестирование программного продукта

6 Введение системы в эксплуатацию

7 Экологический раздел, охрана труда, техника безопасности

7.1 Охрана труда при работе на персональном компьютере

7.2 Техника безопасности при работе с персональным компьютером

Заключение

Библиография

Приложения:

Приложение А. Листинг программы

Презентация доклада для демонстрации в процессе защиты курсовой работы.

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.Е. Ганеев

Задание принял к исполнению студент \_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Темников

Срок сдачи студентом законченной работы: «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**Содержание**

[Введение 7](#_Toc182991187)

[1 Анализ предметной области и формирование требований к разрабатываемому приложению 9](#_Toc182991188)

[1.1 Назначение и цели разработки сайта 9](#_Toc182991189)

[1.2 Формирование требований к сайту 9](#_Toc182991190)

[2 Выбор средств реализации 14](#_Toc182991195)

[2.1 Обзор современных языков программирования для разработки 14](#_Toc182991196)

[2.2 Обзор современных СУБД 14](#_Toc182991197)

[2.4 Выбор и обоснование средств реализации в соответствии с предъявленными требованиями 15](#_Toc182991198)

[3 Состав и содержание работ по созданию системы 16](#_Toc182991199)

[4 Разработка программного продукта 17](#_Toc182991200)

[4.1 Разработка схемы базы данных сайта 17](#_Toc182991201)

[4.2 Разработка пользовательского интерфейса 20](#_Toc182991202)

[4.3 Разработка серверной части проекта 27](#_Toc182991203)

[5 Тестирование программного продукта 32](#_Toc182991204)

[6 Экологический раздел, охрана труда, техника безопасности 36](#_Toc182991205)

[6.1 Охрана труда при работе на персональном компьютере 36](#_Toc182991206)

[6.2 Техника безопасности при работе с персональным компьютером 36](#_Toc182991207)

[Заключение 37](#_Toc182991208)

[Список использованных источников 38](#_Toc182991209)

[Приложение 40](#_Toc182991210)

Введение

**Объектом** исследования являются текстовые RPG и их привлекательность для пользователей. В условиях современного мира, где люди стремятся к новым впечатлениям и приключениям, текстовые ролевые игры предоставляют уникальную возможность погрузиться в увлекательные истории и развивать собственные персонажи. Эти игры позволяют игрокам не только развлекаться, но и проявлять творческий подход, принимая решения, которые влияют на ход событий.

**Предметом** исследования является устройство текстовых RPG, включая механики взаимодействия, создание сюжетов и построение игровых миров. Текстовые RPG привлекают игроков благодаря своей доступности и возможности глубоко погрузиться в сюжет, что делает их интересными как для опытных геймеров, так и для новичков.

**Целью** данной работы является разработка текстовой RPG, которая будет сочетать в себе элементы приключений и исследования. В процессе работы был проведен анализ предметной области, спроектирована и разработана игра, учитывающая современные тенденции в геймдизайне.

Для достижение поставленной цели необходимо решить следующие задачи:

* проанализировать существующие текстовые RPG и требования к ним;
* сформировать требования к игре;
* выбрать средство для реализации игры;
* разработать функциональную модель;
* реализовать игру по модели;
* разработать экономическую часть;

В разделе разработка игры была построена функциональная модель текстовой RPG и осуществлена ее инициализация. Также в рамках проекта была рассчитана стоимость разработки игры, что позволяет оценить экономическую целесообразность данного начинания.

Курсовая работа состоит из введения, анализа предметной области, выбора средств реализации, разработки игры, введения системы в эксплуатацию и экономической части. В введении обоснована актуальность исследования, сформулированы цель и задачи работы. В разделе анализа предметной области определены цели разработки игры, составлены требования к дизайну и механике игры.

В разделе выбора средств реализации описаны технологии и инструменты, используемые для создания текстовой RPG. В заключение подведены основные итоги работы, а также представлены рекомендации по дальнейшему развитию проекта. Библиография включает 16 источников, а приложения дополняют текст курсовой работы.

1. Анализ предметной области и формирование требований к разрабатываемому приложению

1.1 Назначение и цели разработки сайта

«Экспедиция» — это увлекательная текстовая RPG, предлагающая игрокам уникальный опыт исследования морских глубин и взаимодействия с разнообразными персонажами. Игра позволяет пользователям погрузиться в мир приключений, где они могут принимать решения, влияющие на ход событий и развитие сюжета.

Текстовые RPG становятся все более популярными благодаря своей способности сочетать элементы литературы и интерактивности. Они привлекают игроков возможностью использовать воображение и творчески подходить к игровому процессу. В отличие от графических игр, текстовые RPG позволяют игрокам глубже погружаться в сюжет и развивать свои персонажи, принимая важные решения, которые влияют на развитие истории.

Согласно исследованиям, текстовые RPG привлекают внимание благодаря своей гибкости и возможности создания уникальных историй. Они предоставляют игрокам платформу для самовыражения и общения с другими участниками, что особенно важно в условиях современного мира, где многие ищут новые способы взаимодействия и развлечения. Эти игры позволяют пользователям не только развлекаться, но и развивать навыки критического мышления и креативности, что делает их привлекательными для широкой аудитории.

1.2 Формирование требований к игре

1.2.1 Требования к дизайну

Дизайн интерфейса должен быть понятен потенциальным пользователям. Дизайн основного окна должен быть выполнен в сочетающийся цветовой гамме. Дизайн должен обладать легко читаемым шрифтом.

Основной цветовой гаммой будут цвета (рисунок 1) и шрифт Feature Mono[1] (рисунок 2).

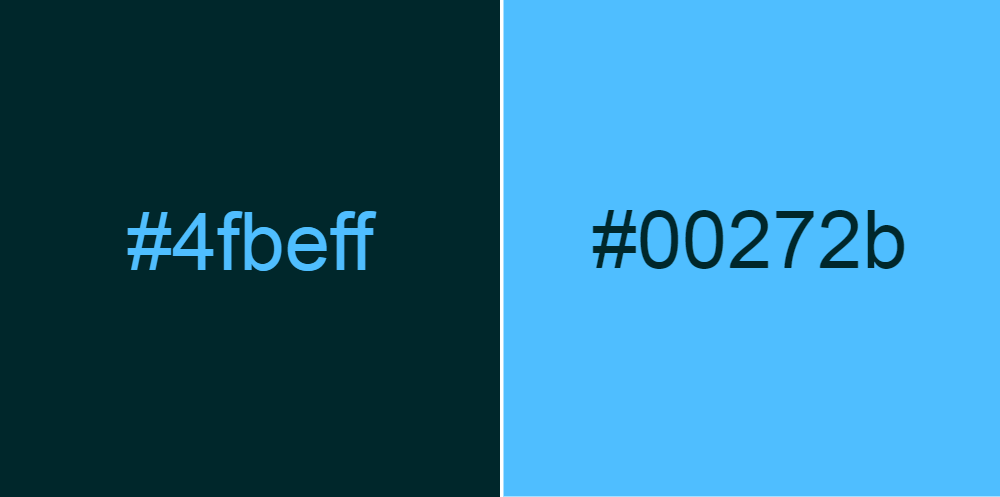


Рисунок 1 – Цвета сайта



Рисунок 2 – Шрифт сайта

* + 1. Требование к функционалу игры

Название игры:

* Отображение названия игры в верхней части окна для идентификации и создания атмосферы.

Кнопки для управления окном:

* Кнопки для минимизации, максимизации и закрытия окна, обеспечивающие удобство взаимодействия с приложением.

Вывод характеристик игрока:

* Здоровье;
* кислород;
* голод.

Кнопка сохранения игры:

* Возможность сохранения текущего прогресса игры для дальнейшего продолжения. Кнопка должна быть легко доступна и интуитивно понятна.

Поле вывода введённых команд и полученных на них ответов:

* Область, где отображаются все введённые игроком команды и соответствующие ответы системы, позволяющая игроку отслеживать ход игры и взаимодействие с окружающим миром.

Поле ввода команд:

* Текстовое поле для ввода команд, где игрок может вводить свои действия и запросы к игре.

Кнопка для отправки команды:

* Кнопка, позволяющая отправлять введённые команды на обработку. Она должна быть расположена рядом с полем ввода команд для удобства использования. Её функционал должен повторяться также и по нажатию на клавишу Enter.

Информация о текущем окружении:

* Название окружения;
* описание окружения;
* список доступных предметов и NPC в окружении.

Система квестов:

* Возможность легкой масштабируемости квестов;
* возможность создания нелинейных квестов;
* информация о выполненных заданиях и доступных действиях.

Инвентарь игрока:

* Возможность открытия инвентаря для просмотра собранных предметов;
* отображение информации о каждом предмете (Название и количество).

1.2.3 Технические требования

Аппаратные требования:

* Процессор: Минимум 1 ГГц, двухъядерный процессор.
* Оперативная память: Не менее 2 ГБ RAM.
* Жесткий диск: Минимум 500 МБ свободного места для установки игры и хранения сохранений.
* Графическая карта: Поддержка OpenGL 2.0 или выше (для графического интерфейса, если он используется).

Программные требования:

* Операционная система: Windows 10 или новее;
* .NET Framework: Не менее версии 4.7.2 или .NET Core 3.1 и выше для запуска приложения.

3. Зависимости

* Библиотеки: Newtonsoft.Json для работы с JSON (для сохранения и загрузки данных).

1. Выбор средств реализации

2.1 Обзор современных языков программирования для разработки

C++ — это высокоуровневый, компилируемый язык программирования, который широко используется для разработки системного программного обеспечения, игр, драйверов и приложений с высокими требованиями к производительности. Он сочетает в себе как низкоуровневые, так и высокоуровневые элементы, что делает его мощным инструментом для разработчиков. C++ поддерживает объектно-ориентированное программирование, что позволяет создавать сложные и масштабируемые приложения.

Плюсы:

* Высокая производительность благодаря компиляции в машинный код;
* поддержка объектно-ориентированного программирования для создания модульных и переиспользуемых компонентов;
* широкие возможности для работы с памятью и низкоуровневыми системными ресурсами;
* большое сообщество и множество библиотек для различных задач.

Минусы:

* Сложный синтаксис, который может быть труден для изучения новичками;
* отсутствие автоматического управления памятью, что увеличивает риск утечек памяти;
* долгое время компиляции по сравнению с интерпретируемыми языками.

Java — это высокоуровневый, объектно-ориентированный язык программирования, который используется для разработки кроссплатформенных приложений. Он работает на принципе "один раз написано — везде работает", что достигается благодаря виртуальной машине Java (JVM). Java широко применяется в веб-разработке, мобильных приложениях (особенно на Android) и корпоративных системах.

Плюсы:

* Портативность благодаря работе на JVM, что позволяет запускать приложения на любых платформах;
* сильная типизация и управление памятью через сборщик мусора, что снижает риск утечек памяти;
* обширная стандартная библиотека и поддержка многопоточности;
* широкое сообщество и множество ресурсов для обучения.

Минусы:

* Более низкая производительность по сравнению с языками, компилируемыми в машинный код (например, C++);
* сложность настройки окружения для разработки и выполнения приложений;
* зачастую более громоздкий синтаксис по сравнению с другими языками.

C# — это высокоуровневый, объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он используется для создания приложений на платформе .NET и подходит как для веб-разработки, так и для разработки настольных приложений. C# предлагает богатый набор функций и инструментов для создания современных приложений.

Плюсы:

* Интуитивно понятный синтаксис, похожий на Java и C++, что облегчает изучение;
* поддержка современных парадигм программирования, включая функциональное программирование;
* интеграция с платформой .NET и доступ к мощным библиотекам;
* отличные инструменты разработки в Visual Studio.

Минусы:

* Привязанность к экосистеме Microsoft может ограничивать кроссплатформенность (хотя .NET Core улучшает эту ситуацию);
* менее распространен в некоторых областях по сравнению с Java или Python;
* потребность в установке .NET Framework или .NET Core для запуска приложений.

Python — это высокоуровневый, интерпретируемый язык программирования, известный своей простотой и читаемостью. Он широко используется в веб-разработке, научных вычислениях, анализе данных и машинном обучении. Python имеет большое сообщество и множество библиотек, что делает его популярным выбором как для начинающих программистов, так и для профессионалов.

Плюсы:

* Простой и понятный синтаксис, который облегчает обучение новичкам;
* широкий выбор библиотек и фреймворков для различных задач (например, Django для веб-разработки, NumPy для научных вычислений);
* поддержка множества парадигм программирования (объектно-ориентированное, функциональное);
* активное сообщество и множество ресурсов для обучения.

Минусы:

* Более низкая производительность по сравнению с компилируемыми языками (например, C++ или Java);
* отсутствие статической типизации может привести к ошибкам во время выполнения;
* меньшая поддержка многопоточности из-за глобальной блокировки интерпретатора (GIL).

2.2 Выбор и обоснование средств реализации в соответствии с предъявленными требованиями

Выбор языка программирования C# для реализации проекта текстовой RPG «Экспедиция» обусловлен несколькими ключевыми факторами, которые делают его особенно подходящим для данной задачи:

* Интуитивно понятный синтаксис:

C# обладает ясным и структурированным синтаксисом, что упрощает процесс разработки и делает его доступным для разработчиков различного уровня. Это позволяет быстро реализовать идеи и сосредоточиться на логике игры, а не на сложностях языка.

* Поддержка объектно-ориентированного программирования:

C# является объектно-ориентированным языком, что идеально подходит для создания сложных игровых систем. Возможность создавать классы и объекты позволяет эффективно организовывать код, что особенно важно для управления игровыми элементами, такими как персонажи, предметы и окружение. Например, в вашем проекте используются классы для описания окружения и предметов (например, Environment, IInventoryItem), что способствует лучшей модульности и переиспользованию кода.

* Широкие возможности платформы .NET:

C# работает в рамках платформы .NET, которая предоставляет множество библиотек и инструментов для разработки. Это позволяет легко интегрировать функциональность, такую как работа с файлами (для сохранения и загрузки игры) и обработка данных в формате JSON (что используется в вашем коде для сериализации состояния игры). Платформа также поддерживает многопоточность и асинхронное программирование, что может быть полезно для улучшения производительности.

* Развитое сообщество и ресурсы:

C# имеет большое сообщество разработчиков и обширную документацию, что облегчает поиск решений возникающих проблем. Наличие множества обучающих материалов позволяет быстро освоить язык и его особенности, что особенно важно при разработке нового проекта.

* Инструменты разработки:

C# поддерживается мощными инструментами разработки, такими как Visual Studio, которые предлагают удобные функции отладки, автозаполнения кода и управления проектами. Это значительно ускоряет процесс разработки и упрощает тестирование игры.

1. Состав и содержание работ по созданию системы

Разработка текстовой RPG «Экспедиция» осуществляется на основе утвержденного технического задания, которое определяет ключевые характеристики и функциональные требования к системе. На этом этапе важно сформулировать основные положения игры, принципы ее функционирования и взаимодействия с другими компонентами, а также определить структуру и конструкцию основных элементов системы.

В состав и содержание работы по созданию системы входят следующие этапы:

* Предварительное обследование и сбор необходимой информации для определения целей игры и задач, которые необходимо решить. Это включает в себя исследование потребностей целевой аудитории и анализ существующих текстовых RPG;
* обследование технологических процессов и детальный анализ системы, включая внедрение организационных требований для достижения поставленных целей. На этом этапе разрабатываются концепции игровых механик и взаимодействия между персонажами, окружением и предметами;
* разработка модели программы, включая описание спецификаций данных, идентификацию отношений между сущностями (например, игрок, NPC, предметы) и построение концептуальных и логических моделей базы данных. Это позволит создать структуру для хранения информации о состоянии игры;
* написание программы, в ходе которого реализуются основные игровые механики, интерфейс пользователя и взаимодействие с базой данных. Программирование включает создание классов для описания окружения, персонажей и предметов;
* отладка программы, на этом этапе выявляются и исправляются ошибки в коде, что обеспечивает стабильную работу игры;
* корректировка программы на основе результатов отладки и обратной связи от тестировщиков. Это может включать изменения в игровом процессе или интерфейсе для улучшения пользовательского опыта;
* проведение тестирования и доработка информационного программного обеспечения по замечаниям и предложениям. Тестирование охватывает функциональные возможности игры, производительность и удобство использования;

Таким образом, процесс разработки текстовой RPG «Экспедиция» включает в себя тщательный анализ целей и задач системы с последующей разработкой модели программы и созданием рабочего проекта, который проходит тестирование и настройку перед сдачей в эксплуатацию.

1. Разработка программного продукта

4.1 Разработка пользовательского интерфейса

В процессе разработки было принято решение об использовании цветов c кодовым HEX номером «#4fbeff» и «#00272B». В качестве основного шрифта используется Feature Mono[1]. Дизайн главной страницы предоставлен на рисунке 3. Пользователю предлагается ввести первую команду



Рисунок 3 – Основное окно

После ввода команды о начале игры, появляются дополнительные элементы интерфейса, представленные на рисунке 4.

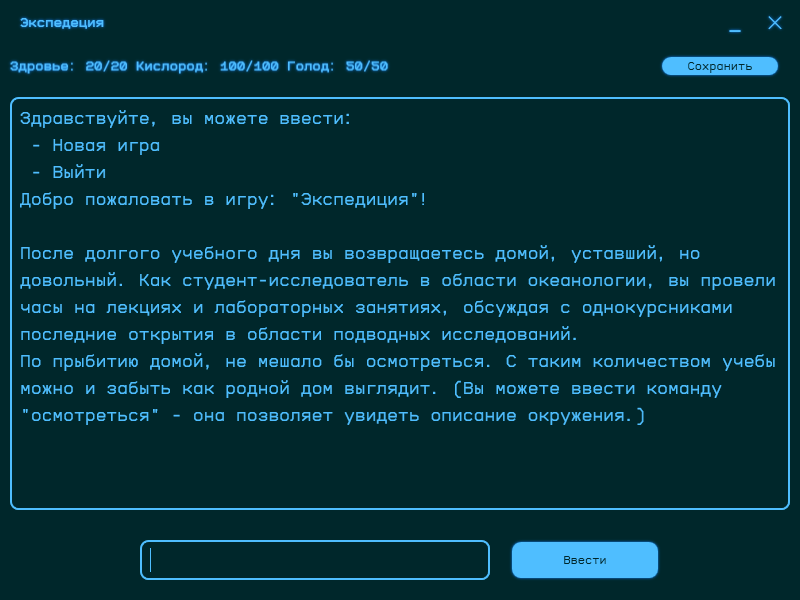


Рисунок 4 – Основное окно с отображением игры

4.2 Разработка функциональной модели игры

Проект строиться на базе C# WPF. Модель игры строится на базовом понимании архитектуры текстовых RPG, но разрабатывалась самостоятельно на собственном понимании структуры игры, для более полного понимания строения текстовых RPG в процессе разработки.

Список моделей игры «Экспедиция»:

* GameModel;
* Player:
* Environment:
* ExitsFromEnvironment;
* IInventoryItem;
* Food:
* HealthKit;
* QuestItem;
* ItemEffect;
* NPC;
* Phrase;
* Quest;
* QuestState;
* QuestStep;
* QuestResults;
* QuestEnding;
* MoralChoice;
* GameSaveData;
* InventoryItemConverter;

GameModel – основная модель игры, содержащая все нужные поля и методы для управления игрой.

Player – модель игрока, отвечающая за хранение данных об игроке и управление ими.

Environment – модель окружения, требуется для создания и управления окружением игрока.

ExitsFromEnvironment – модель, хранящая данные о доступных переходах из окружения.

IInventoryItem – интерфейс для реализации предметов, которые могут храниться в инвентаре у игрока.

Food – модель наследуемая от интерфейса IInventoryItem, реализует свойства еды для игры.

HealthKit – модель наследуемая от интерфейса IInventoryItem, реализует свойства медикаментов для игры.

QuestItem – модель наследуемая от интерфейса IInventoryItem, реализует свойства медикаментов для игры.

ItemEffect – модель, позволяющая реализовывать эффекты у предметов.

NPC – модель, которая реализует возможность создания неигровых персонажей и взаимодействие с ними.

Phrase – модель для создания фраз NPC.

Quest – модель для реализации квестов.

QuestState – список констант состояния квеста.

QuestStep – модель для реализации шагов квеста.

QuestResults – модель для передачи данных на модель игры.

QuestEnding – модель, реализующая возможность квеста к использованию нескольких концовок, в зависимости от действий игрока.

MoralChoice – модель, позволяющая добавлять в квест моральный выбор.

GameSaveData – модель для сохранения данных.

InventoryItemConverter – модель, наследуемая от JsonConverter. Позволяет сериализовать классы, наследуемые от интерфейса IInventoryItem.

Список окон игры:

* MainWindow.

MainWindow – основное окно, реализующее взаимодействие с игроком.

Классы инициализации игры:

* InitializeGame.

InitializeGame – класс для реализации и инициализации игры и ее основного сюжета

Структура папок предоставлена на рисунке 5.

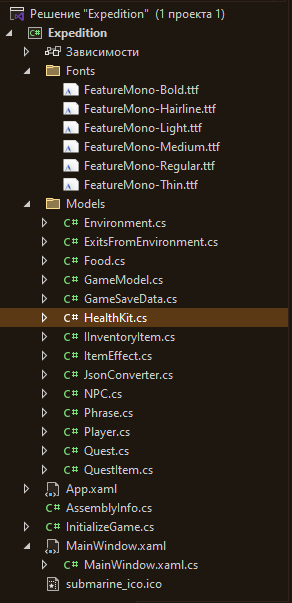


Рисунок 5 – Структура папок

В рамках разработки текстовой RPG «Экспедиция» было выполнено множество задач, среди которых — проектирование и реализация игры, создание модели данных и её интеграция в среду разработки, а также проведение анализа игровых механик. В результате использования программных средств был разработан интуитивно понятный интерфейс и реализованы различные функции, такие как управление инвентарем, взаимодействие с окружением, выполнение квестов и сохранение игрового прогресса.

В целом, все задачи были выполнены успешно, что позволило достичь цели — создания полноценной текстовой RPG «Экспедиция», предлагающей игрокам увлекательный опыт исследования и приключений.

1. Тестирование программного продукта

Тестирование программного продукта играет ключевую роль в процессе разработки. Этот этап необходим для того, чтобы проект мог быть успешно запущен в эксплуатацию. В ходе тестирования могут быть выявлены ошибки, допущенные на этапе разработки. Игра была протестирована с использованием тест-кейсов (таблица 1).

Таблица 1 – Тестирование программного продукта

|  |  |  |
| --- | --- | --- |
| №1: Проверка запуска новой игры. | | |
| Запуск игры | | |
| Действия | Ожидаемый результат | Результат теста |
| 1. Ввести команду «Новая игра» | Инициализация новой игры и отображение начала игры. | Тест успешен |
| №2: Проверка команды прохождения квеста и основных команд | | |
| Запуск игры и инициализация новой игры | | |
| Действия | Ожидаемый результат | Результат теста |
| 1. Ввести команду «Осмотреться» | Получение описания текущего окружения и продвижение квеста на шаг вперед. | Тест успешен |
| 1. Ввести команду «Посмотреть на ноутбук» | Получить описание ноутбука и продвижение квеста на шаг вперед. |
| 1. Ввести команду «Посмотреть уведомление» | Получить описание уведомления и продвижение квеста на шаг вперед. |
| 1. Ввести команду «Взять рюкзак» | Добавление рюкзака в инвентарь и продвижение квеста на шаг вперед. |
| 1. Ввести команду «Перейти в институт» | Смена текущего окружения и смена квеста на следующий. |
| №3: Проверка сохранения игры | | |
| Действия | Ожидаемый результат | Результат теста |
| 1. Нажать кнопку сохранение игры | Создание файла JSON с сохраненной игрой в корневой папке игры | Тест успешен |
| №4: Проверка загрузки игры | | |
| Запуск игры | | |
| Действия | Ожидаемый результат | Результат теста |
| 1 Ввести команду «Загрузить игру» | Десериализация файла JSON из корневой папки игры и вывод игры из файла на экран | Тест успешен |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| №5: Проверка возможности разговоров с NPC | | |
| Запуск игры и загрузка игры | | |
| Действия | Ожидаемый результат | Результат теста |
| 1. Ввести команду «Исследовать» | Получение информации о близлежащих активностях | Тест успешен |
| 1. Ввести команду «Поговорить с Др. Смит» | Получить фразу Др. Смита |
| №6: Проверка на работоспособность морального выбора в концовке игры | | |
| Запуск игры и нахождение на финальной стадии игры | | |
| Действия | Ожидаемый результат | Результат теста |
| * Ввести команду «Взять непонятное устройство» | Получить концовку в соответствии с выбранной командой | Тест успешен |
| * Загрузить игру на состояние до ввода команды | Загрузка последнего сохранения |
| * Ввести команду «Перейти в правый коридор» | Получить концовку в соответствии с выбранной командой |

В результате тестирования был проверен основной функционал игры и сделан вывод о том, что игру можно допускать до выпуска в общий доступ, игра не имеет явных багов и предоставляет полный игровой опыт, задуманный в ней. Функционал игры не имеет критических или каких-либо еще ошибок.

1. Экологический раздел, охрана труда, техника безопасности

6.1 Охрана труда при работе на персональном компьютере

Охрана труда при работе на персональном компьютере включает в себя набор мер и рекомендаций, направленных на обеспечение безопасности и здоровья пользователей. Важно соблюдать правила ergonomics, чтобы предотвратить профессиональные заболевания и обеспечить комфортные условия работы. К основным аспектам охраны труда относятся:

* Правильная организация рабочего места: Стол, стул и монитор должны быть расположены так, чтобы минимизировать нагрузку на позвоночник и глаза. Рекомендуется использовать регулируемые по высоте столы и стулья с поддержкой поясницы.
* Регулярные перерывы: Необходимо делать паузы каждые 30-60 минут для отдыха глаз и тела. Это помогает снизить риск утомления и улучшить концентрацию.
* Освещение: Рабочее место должно быть хорошо освещено, чтобы избежать перенапряжения глаз. Избегайте бликов на экране от окон или ламп.
* Использование защитных средств: При необходимости следует использовать специальные очки для работы за компьютером, которые уменьшают нагрузку на зрение.

6.2 Техника безопасности при работе с персональным компьютером

Техника безопасности при работе с персональным компьютером включает в себя набор правил, которые помогают предотвратить несчастные случаи и повреждение оборудования. К основным рекомендациям относятся:

* Правильное подключение оборудования: Убедитесь, что все кабели подключены правильно и надежно, чтобы избежать короткого замыкания или электрического удара.
* Осторожность с жидкостями: Избегайте размещения напитков рядом с компьютером, чтобы предотвратить случайное проливание и повреждение оборудования.
* Регулярное обновление программного обеспечения: Установка обновлений операционной системы и антивирусного ПО помогает защитить компьютер от вирусов и других угроз.
* Соблюдение правил эксплуатации: Не перегружайте компьютер, следите за его температурой и состоянием компонентов. При возникновении подозрительных звуков или запахов немедленно отключите устройство от сети.

Заключение

Потребность человека в приключениях и исследованиях является естественной и актуальной в условиях современного мира. В условиях быстрого темпа жизни и постоянной занятости у людей часто не остается времени для полноценного отдыха и развлечений. Текстовые RPG предоставляют уникальную возможность погрузиться в увлекательные истории, не выходя из дома, и позволяют игрокам исследовать новые миры, развивать персонажей и принимать важные решения.Целью данного проекта была разработка текстовой RPG «Экспедиция». В ходе работы были выполнены следующие задачи: анализ предметной области, проектирование игровых механик, создание модели данных и реализация ее в среде разработки. Также был проведен сравнительный анализ существующих текстовых RPG для выявления лучших практик и подходов.Разработанная игра включает в себя такие функции, как:

* Управление персонажем;
* исследование окружения и взаимодействие с NPC;
* выполнение квестов и сбор предметов;
* управление инвентарем и использование предметов.

Таким образом, текстовая RPG «Экспедиция» представляет собой полноценный проект, который предлагает игрокам увлекательный опыт исследования подводных глубин и взаимодействия с разнообразными персонажами. Игра не только развлекает, но и способствует развитию критического мышления и творческого подхода к решению задач.

Список использованных источников

1. Feature Mono разработан Анастасией Врублевской в Школе дизайна НИУ ВШЭ.
2. StopGame: История текстовых ролевых игр. <https://stopgame.ru/blogs/topic/106500/istoriya_tekstovyh_rolevyh_igr>
3. Habr: <https://habr.com/ru/articles/647251/>
4. Дзен: <https://dzen.ru/a/Z1vQoHSRngIzTKJg>
5. Дзен: https://dzen.ru/a/ZJGdgo24o2AUuR86

Приложение

Приложение А. Листинг кода программы

|  |
| --- |
| MainWindow.xaml |
| <Window x:Class="Expedition.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:Expedition"  mc:Ignorable="d"  Title="MainWindow"  Height="600"  Width="800"  Background="#00272B"  WindowStyle="None"  WindowStartupLocation="CenterScreen"  MouseDown="Window\_MouseDown"  AllowsTransparency="True"  ResizeMode="NoResize">  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="Auto"/>  <RowDefinition/>  </Grid.RowDefinitions>  <Border Background="Transparent" CornerRadius="15">  <Grid Grid.Row="0">  <Grid.ColumnDefinitions>  <ColumnDefinition Width="20"/>  <ColumnDefinition/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <TextBlock Grid.Column="1" Text="Экспедеция" HorizontalAlignment="Left" VerticalAlignment="Center" FontSize="14" Style="{StaticResource MainText}"></TextBlock>  <StackPanel Grid.Column="2" Orientation="Horizontal" Margin="0 5 5 0">  <TextBlock FontSize="18" Style="{StaticResource MainText}"  Cursor="Hand" MouseDown="appMinimize"  TextAlignment="Center" VerticalAlignment="Center" HorizontalAlignment="Center"  Margin="5,5,5,5" Width="30" Height="30">\_</TextBlock>  <TextBlock FontSize="18" Style="{StaticResource MainText}"  Cursor="Hand" MouseDown="appExit"  TextAlignment="Center" VerticalAlignment="Center" HorizontalAlignment="Center"  Margin="5,5,5,5" Width="30" Height="30">✕</TextBlock>  </StackPanel>  </Grid>  </Border>  <Grid Grid.Row="1">  <Grid.RowDefinitions>  <RowDefinition Height="Auto"></RowDefinition>  <RowDefinition></RowDefinition>  <RowDefinition Height="Auto"></RowDefinition>  </Grid.RowDefinitions>  <Grid Grid.Row="0" HorizontalAlignment="Stretch" VerticalAlignment="Center" Margin="10, 5">  <TextBlock Name="txtPlayesStats" Style="{StaticResource MainText}" Text="" FontSize="14" HorizontalAlignment="Left" VerticalAlignment="Center"></TextBlock>  <Button Name="btnSave" Style="{StaticResource MainButton}" Width="120" Margin="10, 5" Content="Сохранить" HorizontalAlignment="Right" VerticalAlignment="Center" Click="btnSave\_Click" Visibility="Hidden"/>  </Grid>  <TextBox Name="OutputTextBox" Grid.Row="1" TextWrapping="Wrap" IsReadOnly="True" VerticalScrollBarVisibility="Auto"  Margin="10" AcceptsReturn="True" Style="{StaticResource CustomTextBoxStyle}"/>  <StackPanel Grid.Row="2" Orientation="Horizontal" Margin="10" HorizontalAlignment="Center" VerticalAlignment="Center">  <TextBox Name="InputTextBox" Style="{StaticResource CustomTextBoxStyle}" Height="40" Width="350" Margin="10" VerticalScrollBarVisibility="Disabled" KeyDown="InputTextBox\_KeyDown"/>  <Button Style="{StaticResource MainButton}" Width="150" Margin="10" Content="Ввести" Click="btnEnter\_Click"/>  </StackPanel>  </Grid>  </Grid>  </Window> |
| MainWindow.xaml.cs |
| using Expedition.Models;  using System.Text;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Navigation;  using System.Windows.Shapes;  using System.IO;  using Path = System.IO.Path;  namespace Expedition  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  private GameModel gameModel;  private bool IsGameUpload;  private string textHistory;  public MainWindow()  {  InitializeComponent();  StartApp();  OutputTextBox.ScrollToEnd();  }  private void StartApp()  {  string helloWords = "Здравствуйте, вы можете ввести:\n - Новая игра\n ";  string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "savegame.json");  if (File.Exists(filePath))  {  helloWords += "- Загрузить игру\n ";  }  helloWords += "- Выйти\n";  OutputTextBox.AppendText(helloWords);  }  private void UpdateOutput(string message)  {  if (IsGameUpload)  {  GameUploadOutput(message);  }  else  {  NonGameUploadOutput(message);  }  OutputTextBox.ScrollToEnd();  }  private void GameUploadOutput(string message)  {  string response = gameModel.ProcessCommand(message);  if (response.StartsWith("Вы умерли!"))  {  textHistory = response;  OutputTextBox.AppendText(response);  }  else  {  var textToPlayer = $"Вы: {message}\n\n{response}\n";  OutputTextBox.AppendText(textToPlayer);  textHistory += textToPlayer;  }  txtPlayesStats.Text = gameModel.Player.CurrentStats;  if (response.EndsWith("Игра окончена!"))  {  string helloWords = "Вы можете ввести:\n - Новая игра\n ";  string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "savegame.json");  if (File.Exists(filePath))  {  helloWords += "- Загрузить игру\n ";  }  helloWords += "- Выйти\n";  OutputTextBox.AppendText(helloWords);  IsGameUpload = false;  }  }  private void NonGameUploadOutput(string message)  {  string command = message.ToLower();  if (command == "новая игра")  {  gameModel = new InitializeGame().CreateExpeditionGame("Дэй");  OutputTextBox.AppendText($"Добро пожаловать в игру: \"Экспедиция\"!\n\n");  var response = gameModel.CurrentQuest.StartQuest() + "\n\n";  textHistory = "";  textHistory += response;  OutputTextBox.AppendText(response);  IsGameUpload = true;  txtPlayesStats.Text = gameModel.Player.CurrentStats;  btnSave.Visibility = Visibility.Visible;  }  else if (command == "загрузить игру")  {  gameModel = new GameModel();  textHistory = gameModel.LoadGame();  OutputTextBox.AppendText(textHistory + "\n\n");  OutputTextBox.AppendText(gameModel.CurrentQuest.GetCurrentStepDescription() + "\n\n");  IsGameUpload = true;  txtPlayesStats.Text = gameModel.Player.CurrentStats;  btnSave.Visibility = Visibility.Visible;  }  else if (command == "выйти")  {  Application.Current.Shutdown();  }  else  {  OutputTextBox.AppendText("Неизвестная команда.\n");  }  }  private void InputTextBox\_KeyDown(object sender, KeyEventArgs e)  {  if (e.Key == Key.Enter)  {  string input = InputTextBox.Text.Trim();  InputTextBox.Clear();  UpdateOutput(input);  }  }  private void Window\_MouseDown(object sender, MouseButtonEventArgs e)  {  if (e.LeftButton == MouseButtonState.Pressed)  {  DragMove();  }  }  private void appExit(object sender, MouseButtonEventArgs e)  {  Application.Current.Shutdown();  }  private void appMinimize(object sender, MouseButtonEventArgs e)  {  Application.Current.MainWindow.WindowState = WindowState.Minimized;  }  private void btnSave\_Click(object sender, RoutedEventArgs e)  {  gameModel.SaveGame(textHistory);  }  private void btnEnter\_Click(object sender, RoutedEventArgs e)  {  string input = InputTextBox.Text.Trim();  InputTextBox.Clear();  UpdateOutput(input);  }  }  } |
|  |
| GameModel.cs |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Numerics;  using System.Runtime.Serialization.Formatters.Binary;  using System.Text;  using Newtonsoft.Json;  using System.Threading.Tasks;  using System.Windows.Documents;  using System.Security.AccessControl;  namespace Expedition.Models  {  public class GameModel  {  public Player Player { get; private set; }  public List<Quest> Quests { get; private set; }  public Quest CurrentQuest { get; set; }  public List<Environment> Environments { get; private set; }  public Environment CurrentEnvironment { get; set; }  public List<NPC> NPCs { get; private set; }  public List<IInventoryItem> AllItems { get; private set; }  public List<ExitsFromEnvironment> AllTransfers { get; private set; }  public GameModel()  {  Player = new Player();  Quests = new List<Quest>();  Environments = new List<Environment>();  NPCs = new List<NPC>();  AllItems = new List<IInventoryItem>();  AllTransfers = new List<ExitsFromEnvironment>();  }  public void SetQuests(List<Quest> quests)  {  Quests = quests;  }  public void SetEnviromnets(List<Environment> environments)  {  Environments = environments;  }  public void SetNPCS(List<NPC> npcs)  {  NPCs = npcs;  }  public void SetItems(List<IInventoryItem> items)  {  AllItems = items;  }  public void SetTransfers(List<ExitsFromEnvironment> transfers)  {  AllTransfers = transfers;  }  public void SetCurrentQuest(int id)  {  CurrentQuest = Quests[id];  }  public void SetCurrentEnvironment(int id)  {  CurrentEnvironment = Environments[id];  }  public string ProcessCommand(string userCommand)  {  var response = "";  var command = userCommand.ToLower();  if(command.StartsWith("осмотреться"))  {  response += CurrentEnvironment.GetDescription();  }  if (command.StartsWith("исследовать"))  {  response += CurrentEnvironment.GetInteractives();  }  if(command.StartsWith("выходы"))  {  response += GetExits();  }  if (command.StartsWith("посмотреть на"))  {  var item = GetItem(command, "посмотреть на ");  if(item != null)  {  if(CurrentEnvironment.HasItem(item)) response += item.Description;  }  else  {  response += "Такого предмета не найдено";  }  }  if (command.StartsWith("взять"))  {  var item = GetItem(command, "взять ");  if (item == null)  {  response += "Такого предмета не найдено.";  }  else  {  if (!CurrentEnvironment.HasItem(item))  {  response += "Такого предмета нет в окружении.";  }  else  {  if (!Player.AddToInventory(item))  {  response += "Вы не можете это взять! Максимальное количество уже в инвентаре.";  }  else  {  CurrentEnvironment.RemoveItem(item);  response += $"Вы взяли {item.Name}.";  }  }  }  }  if (command.StartsWith("открыть инвентарь"))  {  response += Player.GetInventory();  }  if(command.StartsWith("использовать "))  {  var item = GetItem(command, "использовать");  if (item != null)  {  if(Player.HasItem(item))  {  var effect = item.UseItem();  if (effect.EffectName == "saturation")  {  Player.CurrentHunger = Player.CurrentHunger + effect.EffectValue <= Player.MaximumHunger ? Player.CurrentHunger + effect.EffectValue : Player.MaximumHunger;  if(!Player.RemoveFromInventory(item))  {  response += $"Не удалось использовать {item.Name}";  }  }  if (effect.EffectName == "healing")  {  Player.CurrentHealthPoint = Player.CurrentHealthPoint + effect.EffectValue <= Player.MaximumHealthPoint ? Player.CurrentHealthPoint + effect.EffectValue : Player.MaximumHealthPoint;  if (!Player.RemoveFromInventory(item))  {  response += $"Не удалось использовать {item.Name}";  }  }  response += effect.UseResponse;  }  }  else  {  response += "Такого предмета не найдено";  }  }  if (command.StartsWith("перейти в"))  {  if (!CurrentQuest.Steps[CurrentQuest.currentStepIndex].IsMovementAllowed)  {  return "Вы не можете перемещаться сейчас.";  }  var envName = command.Replace("перейти в ", String.Empty).Trim();  var env = Environments.FirstOrDefault(x => x.Name.ToLower() == envName.ToLower() || x.SecondName.ToLower() == envName.ToLower());  if (env != default)  {  var exits = AllTransfers.Where(x => x.EnvironmentName == CurrentEnvironment.Name).FirstOrDefault();  if (exits != default)  {  var thisEnv = exits.Exits.FirstOrDefault(x => x.Name == env.Name);  if (thisEnv == default) response += "Такого перехода не существует";  else  {  CurrentEnvironment = env;  response += $"Вы перешли в {env.Name}";  }  }  }  else  {  response += "Подобная локация не найдена";  }  }  if(command.StartsWith("поговорить с"))  {  var npcName = command.Replace("поговорить с ", String.Empty).Trim();  var npc = NPCs.Where(x => x.Name.ToLower() == npcName.ToLower()).FirstOrDefault();  if (npc != default)  {  response += npc.GetPhrase(CurrentEnvironment.Name);  }  }  if(command == "загрузить игру")  {  LoadGame();  return CurrentQuest.GetCurrentStepDescription();  }  var effectStr = EnvironmentEffect();  if (effectStr.StartsWith("Вы умерли!"))  {  return effectStr;  }  if(response == "")  {  response += "Неизвестная команда.";  return response;  }  return response + "\n" + CheckQuest(command);  }  public IInventoryItem GetItem(string command, string commandStart)  {  var objectName = command.Replace(commandStart, string.Empty).Trim();  var item = AllItems.FirstOrDefault(x => x.Name.ToLower() == objectName.ToLower() || x.SecondName.ToLower() == objectName.ToLower());  return item;  }  public string GetExits()  {  string exitStr = "";  var exits = AllTransfers.Where(x => x.EnvironmentName == CurrentEnvironment.Name).FirstOrDefault().Exits;  if (exits != default)  {  foreach (var ex in exits)  {  exitStr += $" - {ex.Name}\n";  }  }  if (exitStr == "") return "Нет переходов";  return exitStr;  }  public string EnvironmentEffect()  {  if (CurrentEnvironment?.IsDangerous == true)  {  int effect = (int)CurrentEnvironment.NegativeEffect;  Player.CurrentOxygen = Player.CurrentOxygen - effect > 0 ? Player.CurrentOxygen - effect : 0;  if(Player.CurrentOxygen == 0)  {  Player.CurrentHealthPoint -= effect/2;  }  }  else  {  Player.CurrentOxygen = Player.MaximumOxygen;  }  int damage = 2;  Player.CurrentHunger = Player.CurrentHunger - damage > 0 ? Player.CurrentHunger - damage : 0;  if(Player.CurrentHunger == 0)  {  Player.CurrentHealthPoint -= damage/2;  }    if (Player.CurrentHealthPoint <= 0)  {  return Death();  }  return "";  }  public string Death()  {  string loadMessage = LoadGame();  if (string.IsNullOrEmpty(loadMessage))  {  return "Вы умерли! Игра загружена с последнего сохранения.";  }  else  {  return $"Вы умерли! {loadMessage}";  }  }  public string CheckQuest(string command)  {  var response = CurrentQuest.ProcessCommand(command, Player.Inventory);  if(response.ResultText == "NonQuest")  {  return "";  }  if(response.IsEnd == true)  {  var questIndex = Quests.FindIndex(a => a.Title == CurrentQuest.Title);  Quests[questIndex] = CurrentQuest;  if (CheckAllQuestsCompleted() != "")  {  return response.ResultText + "\n" + CheckAllQuestsCompleted();  }  CurrentQuest = Quests[questIndex + 1];  if (response.ItemsToRemove != null && response.ItemsToRemove.Count > 0)  {  foreach (var item in response.ItemsToRemove)  {  Player.RemoveFromInventory(item);  }  }  return CurrentQuest.GetStartText() + "\n" + CurrentQuest.Steps[CurrentQuest.currentStepIndex].Description;  }  return response.ResultText;  }  public string CheckAllQuestsCompleted()  {  bool allQuestsCompleted = Quests.All(quest => quest.State == QuestState.Completed);  if (allQuestsCompleted)  {  return "Поздравляем! Все квесты завершены. Игра окончена!";  }  return "";  }  public void SaveGame(string textHistory)  {  string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "savegame.json");  GameSaveData saveData = new GameSaveData  {  Player = Player,  Quests = Quests,  CurrentQuestIndex = Quests.IndexOf(CurrentQuest),  Environments = Environments,  CurrentEnvironmentIndex = Environments.IndexOf(CurrentEnvironment),  TextHistory = textHistory,  NPCs = NPCs,  AllItems = AllItems,  AllTransfers = AllTransfers  };  string json = JsonConvert.SerializeObject(saveData, Formatting.Indented, new JsonSerializerSettings  {  Converters = { new InventoryItemConverter() }  });  File.WriteAllText(filePath, json);  }  public string LoadGame()  {  string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "savegame.json");  if (File.Exists(filePath))  {  try  {  string json = File.ReadAllText(filePath);  GameSaveData saveData = JsonConvert.DeserializeObject<GameSaveData>(json, new JsonSerializerSettings  {  Converters = { new InventoryItemConverter() }  });  if (saveData == null)  {  return "Ошибка загрузки: данные сохранения пусты.";  }  Player = saveData.Player ?? new Player();  Quests = saveData.Quests ?? new List<Quest>();  var currentQuestIndex = saveData.CurrentQuestIndex;  if (currentQuestIndex >= 0 && currentQuestIndex < Quests.Count)  {  CurrentQuest = Quests[currentQuestIndex];  }  else  {  return "Ошибка загрузки: некорректный индекс текущего квеста.";  }  Environments = saveData.Environments ?? new List<Environment>();  var currentEnvironmentIndex = saveData.CurrentEnvironmentIndex;  if (currentEnvironmentIndex >= 0 && currentEnvironmentIndex < Environments.Count)  {  CurrentEnvironment = Environments[currentEnvironmentIndex];  }  else  {  return "Ошибка загрузки: некорректный индекс текущей локации.";  }  NPCs = saveData.NPCs ?? new List<NPC>();  AllItems = saveData.AllItems ?? new List<IInventoryItem>();  AllTransfers = saveData.AllTransfers ?? new List<ExitsFromEnvironment>();  return saveData.TextHistory ?? "История текста отсутствует.";  }  catch (JsonReaderException ex)  {  return $"Ошибка чтения сохраненной игры: {ex.Message}";  }  catch (Exception ex)  {  return $"Не удалось загрузить игру: {ex.Message}";  }  }  return "Не получилось загрузить игру! Файл не найден.";  }  }  } |
| Player.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class Player  {  public string Name { get; set; }  public List<IInventoryItem> Inventory { get; set; }  public int CurrentHealthPoint { get; set; }  public int MaximumHealthPoint { get; set; }  public int CurrentOxygen { get; set; }  public int MaximumOxygen { get; set; }  public int CurrentHunger { get; set; }  public int MaximumHunger { get; set; }  public string CurrentStats  {  get  {  return $"Здровье: {CurrentHealthPoint}/{MaximumHealthPoint} Кислород: {CurrentOxygen}/{MaximumOxygen} Голод: {CurrentHunger}/{MaximumHunger}";  }  }  public Player()  {  Inventory = new List<IInventoryItem>();  Name = "Исследователь";  MaximumHealthPoint = 20;  MaximumOxygen = 100;  MaximumHunger = 50;  CurrentHealthPoint = MaximumHealthPoint;  CurrentOxygen = MaximumOxygen;  CurrentHunger = MaximumHunger;  }  public bool AddToInventory(IInventoryItem item)  {  if (Inventory.Count(x => x.Name == item.Name) < item.MaximumCount)  {  Inventory.Add(item);  return true;  }  return false;  }  public bool RemoveFromInventory(IInventoryItem item)  {  var existingItem = Inventory.FirstOrDefault(x => x.Name == item.Name);  if (existingItem != null)  {  Inventory.Remove(existingItem);  return true;  }  return false;  }  public bool HasItem(IInventoryItem item)  {  var thisItem = Inventory.FirstOrDefault(x => x.Name == item.Name);  if (thisItem == default) return false;  return true;  }  public string GetInventory()  {  var inventoryString = "\n";  foreach (var item in Inventory.GroupBy(x => x.Name))  {  inventoryString += $"- {item.Key} (Количество: {item.Count()})\n";  }  if (inventoryString == "\n")  return "В инвентаре пусто!";  return inventoryString;  }  }  } |
| Enivornment.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class Environment  {  public string Name { get; private set; }  public string SecondName { get; private set; }  public string Description { get; private set; }  public List<IInventoryItem> ItemsInEnvironment { get; private set; }  public List<NPC> CharactersInEnvironment { get; private set; }  public bool IsDangerous { get; set; }  public int? NegativeEffect { get; set; }  public Environment(string name, string description, string secondName = "")  {  Name = name;  Description = description;  ItemsInEnvironment = new List<IInventoryItem>();  CharactersInEnvironment = new List<NPC>();  IsDangerous = false;  if(secondName == "")  {  SecondName = name;  }  else  {  SecondName = secondName;  }  }  public void AddItem(IInventoryItem item)  {  ItemsInEnvironment.Add(item);  }  public void RemoveItem(IInventoryItem item)  {  ItemsInEnvironment.Remove(item);  }  public void AddCharacter(NPC character)  {  CharactersInEnvironment.Add(character);  }  public void SetDangerous(int effect)  {  IsDangerous = true;  NegativeEffect = effect;  }  public string GetDescription()  {  return Description;  }  public string GetInteractives()  {  string interactives = "";  interactives += GetCharacters();  interactives += GetItems();  if (interactives == "") return "Рядом нет активностей";  return interactives;  }  public string GetCharacters()  {  var characters = "";  if (CharactersInEnvironment.Count > 0)  {  characters += "Люди рядом:\n";  foreach (var character in CharactersInEnvironment)  {  characters += $"- {character.Name}\n";  }  }  return characters;  }  public string GetItems()  {  var items = "";  if (ItemsInEnvironment.Count > 0)  {  items += "Предметы рядом:\n";  foreach (var item in ItemsInEnvironment.GroupBy(x => x.Name))  {  items += $"- {item.Key}\n";  }  }  if (items == "\n")  return "В комнате пусто";  return items;  }  public bool HasItem(IInventoryItem item)  {  var thisItem = ItemsInEnvironment.FirstOrDefault(x => x.Name == item.Name);  if(thisItem == default) return false;  return true;  }  }  } |
| ExitsFromEnvironment.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class ExitsFromEnvironment  {  public string EnvironmentName { get; set; }  public List<Environment> Exits { get; set; }  public ExitsFromEnvironment(string environmentName)  {  EnvironmentName = environmentName;  Exits = new List<Environment>();  }  public void AddExit(Environment environment)  {  Exits.Add(environment);  }  static public Environment Transfer(List<ExitsFromEnvironment> allExits, string currentEnvironment, string nextEnvironment)  {  var exits = allExits.Where(x => x.EnvironmentName == currentEnvironment).FirstOrDefault();  if (exits != default)  {  var exit = exits.Exits.Where(x => x.Name == nextEnvironment).FirstOrDefault();  if (exit != default)  {  return exit;  }  }  return null;  }  }  } |
| IInventoryItem.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public interface IInventoryItem  {  public string Name { get; set; }  public string Description { get; set; }  public int MaximumCount { get; set; }  public bool IsVisible { get; set; }  public string SecondName { get; set; }  public string InspectItem();  public ItemEffect UseItem();  }  } |
| Food.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Xml.Linq;  namespace Expedition.Models  {  public class Food : IInventoryItem  {  public string Name { get; set; }  public string Description { get; set; }  public int Saturation { get; set; }  public int MaximumCount { get; set; }  public bool IsVisible { get; set; } = true;  private string? \_secondName;  public string? SecondName  {  get  {  return \_secondName ?? Name;  }  set  {  \_secondName = value;  }  }  public string InspectItem()  {  return Description;  }  public ItemEffect UseItem()  {  return new ItemEffect("saturation", Saturation, $"Вы съели {Name}.");  }  }  } |
| HealthKit.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class HealthKit : IInventoryItem  {  public string Name { get; set; }  public string Description { get; set; }  public int RecoverableHealth { get; set; }  public int MaximumCount { get; set; }  public bool IsVisible { get; set; } = true;  private string? \_secondName;  public string? SecondName  {  get  {  return \_secondName ?? Name;  }  set  {  \_secondName = value;  }  }  public string InspectItem()  {  return Description;  }  public ItemEffect UseItem()  {  return new ItemEffect("healing", RecoverableHealth, $"Вы использовали: {Name}");  }  }  } |
| QuestItem.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class QuestItem : IInventoryItem  {  public string Name { get; set; }  public string Description { get; set; }  public string Action { get; set; }  public int MaximumCount { get; set; }  public bool IsVisible { get; set; } = true;  private string? \_secondName;  public string? SecondName  {  get  {  return \_secondName ?? Name;  }  set  {  \_secondName = value;  }  }  public string InspectItem()  {  return Description;  }  public ItemEffect UseItem()  {  return new ItemEffect("quest", 1, $"Вы использовали {Name}.");  }  }  } |
| ItemEffect.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class ItemEffect  {  public string EffectName { get; set; }  public int EffectValue { get; set; }  public string UseResponse { get; set; }  public ItemEffect(string effectName, int effectValue, string useResponse)  {  EffectName = effectName;  EffectValue = effectValue;  UseResponse = useResponse;  }  }  } |
| NPC.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class NPC  {  public string Name { get; set; }  public List<Phrase> Phrases { get; set; }  public NPC(string name) {  Name = name;  Phrases = new List<Phrase>();  }  public void AddPhrase(string text, string environmentName)  {  Phrases.Add(new Phrase(text, environmentName));  }  public string GetPhrase(string environmentName)  {  var phrase = Phrases.Where(x => x.EnvironmentName == environmentName).FirstOrDefault();  return phrase != default ? phrase.GetPhrase(Name) : $"Сейчас вы не можете поговорить с {Name}";  }  }  } |
| Phrase.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class Phrase  {  public string Text { get; set; }  public string EnvironmentName { get; set; }  public bool IsCheck { get; set; }  public Phrase(string text, string environmentName)  {  Text = text;  EnvironmentName = environmentName;  IsCheck = false;  }  public string GetPhrase(string characterName)  {  if(!IsCheck)  {  return Text;  }  else  {  return $"Вы уже разговаривали {characterName}.";  }  }  }  } |
| Quest.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Expedition.Models  {  public class Quest  {  public string Title { get; private set; }  public string StartText { get; private set; }  public List<QuestStep> Steps { get; private set; }  public QuestState State { get; set; }  public int currentStepIndex;  public List<IInventoryItem> ItemsToRemove { get; private set; }  public List<QuestEnding> QuestEndings { get; set; }  public Quest(string title, string startText, List<IInventoryItem> items = null)  {  Title = title;  StartText = startText;  Steps = new List<QuestStep>();  State = QuestState.NotStarted;  currentStepIndex = 0;  ItemsToRemove = new List<IInventoryItem>();  QuestEndings = new List<QuestEnding>();  if(items != null)  {  ItemsToRemove = items;  }  }  public string StartQuest()  {  State = QuestState.InProgress;  return StartText + "\n" + Steps[currentStepIndex].Description;  }  public string GetStartText()  {  return StartText;  }  public void SetSteps(List<QuestStep> steps)  {  Steps = steps;  }  public void AddStep(QuestStep step)  {  Steps.Add(step);  }  public void AddItemToRemove(QuestItem item)  {  ItemsToRemove.Add(item);  }  public void SetEndings(List<QuestEnding> endings)  {  QuestEndings = endings;  }  public void AddEnding(string text, List<QuestStep> steps)  {  QuestEndings.Add(new QuestEnding()  {  EndingText = text,  NecessarySteps = steps  });  }  public string GetCurrentStepDescription()  {  if (currentStepIndex < Steps.Count)  {  return Steps[currentStepIndex].Description;  }  return "Квест завершен!";  }  public QuestResults ProcessCommand(string command, List<IInventoryItem> playerItems)  {  if (currentStepIndex < Steps.Count)  {  var currentStep = Steps[currentStepIndex];  currentStep.CheckRequiredItems(playerItems);  var check = currentStep.Complete(command);  if (currentStep.MoralChoice != null)  {  return HandleMoralChoice(currentStep.MoralChoice, command);  }  if (check)  {  return CompleteCurrentStep();  }  }  return new QuestResults("NonQuest", null);  }  private QuestResults HandleMoralChoice(MoralChoice moralChoice, string playerDecision)  {  if (playerDecision.ToLower() == moralChoice.ChoicePositiveText)  {  State = QuestState.Completed;  return new QuestResults(moralChoice.PositiveOutcome, null, true);  }  else if (playerDecision.ToLower() == moralChoice.ChoiceNegativeText)  {  State = QuestState.Completed;  return new QuestResults(moralChoice.NegativeOutcome, null, true);  }  return new QuestResults("Неизвестный выбор.", null);  }  private QuestResults CompleteCurrentStep()  {  currentStepIndex++;  if (currentStepIndex >= Steps.Count)  {  State = QuestState.Completed;  return new QuestResults(GetEnding(), ItemsToRemove, true);  }  return new QuestResults(Steps[currentStepIndex].Description, null);  }  public string GetEnding()  {  foreach (var ending in QuestEndings)  {  bool allStepsCompleted = true;  foreach (var step in ending.NecessarySteps)  {  var questStep = Steps.FirstOrDefault(x => x.Description == step.Description);  if(questStep != default)  {  if (!questStep.IsCompleted)  {  allStepsCompleted = false;  break;  }  }  }  if (allStepsCompleted)  {  return ending.EndingText;  }  }  return "К сожалению, вы не смогли выполнить все задачи квеста.";  }  }  public enum QuestState  {  NotStarted,  InProgress,  Completed  }  public class QuestStep  {  public List<string> NecessaryActions { get; private set; }  public string Description { get; private set; }  public bool IsCompleted { get; set; }  public bool IsMovementAllowed { get; set; }  public List<IInventoryItem> RequiredItems { get; private set; }  public MoralChoice MoralChoice { get; set; }  public QuestStep(List<string> necessaryActions, string description, List<IInventoryItem> requiredItems = null, bool isMovement = true)  {  NecessaryActions = necessaryActions;  Description = description;  IsCompleted = false;  RequiredItems = requiredItems ?? new List<IInventoryItem>();  IsMovementAllowed = isMovement;  }  public void SetMovementTrue()  {  IsMovementAllowed = true;  }  public void SetMovementFalse()  {  IsMovementAllowed = false;  }  public void SetMoralChoice(MoralChoice moralChoice)  {  MoralChoice = moralChoice;  }  public bool Complete(string action)  {  if (NecessaryActions.Contains(action))  {  NecessaryActions.Remove(action);  if (NecessaryActions.Count == 0 && CanComplete())  {  IsCompleted = true;  return true;  }  return false;  }  return false;  }  public bool CanComplete()  {  return IsCompleted || RequiredItems.Count == 0;  }  public void CheckRequiredItems(List<IInventoryItem> playerItems)  {  foreach (var item in RequiredItems.ToList())  {  var thisItem = playerItems.FirstOrDefault(x => x.Name == item.Name);  if (thisItem != default)  {  NecessaryActions.Remove($"взять {item.Name.ToLower()}");  RequiredItems.Remove(item);  }  }  }  }  public class QuestResults  {  public string ResultText { get; set; }  public List<IInventoryItem> ItemsToRemove { get; set; }  public bool IsEnd { get; set; }  public QuestResults(string result, List<IInventoryItem> items, bool isEnd = false)  {  ResultText = result;  ItemsToRemove = items;  IsEnd = isEnd;  }  public void SetItemsToRemove(List<IInventoryItem> items)  {  ItemsToRemove = items;  }  public void AddItem(IInventoryItem item)  {  ItemsToRemove.Add(item);  }  }  public class QuestEnding  {  public string EndingText { get; set; }  public List<QuestStep> NecessarySteps { get; set; }  }  public class MoralChoice  {  public string ChoicePositiveText { get; set; }  public string ChoiceNegativeText { get; set; }  public string PositiveOutcome { get; set; }  public string NegativeOutcome { get; set; }  public MoralChoice(string choicePositiveText, string choiceNegativeText, string positiveOutcome, string negativeOutcome)  {  ChoicePositiveText = choicePositiveText;  ChoiceNegativeText = choiceNegativeText;  PositiveOutcome = positiveOutcome;  NegativeOutcome = negativeOutcome;  }  }  } |
| GameSaveData.cs |
| using System;  using System.Collections.Generic;  namespace Expedition.Models  {  [Serializable]  public class GameSaveData  {  public Player Player { get; set; }  public List<Quest> Quests { get; set; }  public int CurrentQuestIndex { get; set; }  public List<Environment> Environments { get; set; }  public int CurrentEnvironmentIndex { get; set; }  public string TextHistory { get; set; }  public List<NPC> NPCs { get; set; }  public List<IInventoryItem> AllItems { get; set; }  public List<ExitsFromEnvironment> AllTransfers { get; set; }  public GameSaveData()  {  Player = new Player();  Quests = new List<Quest>();  Environments = new List<Environment>();  NPCs = new List<NPC>();  AllItems = new List<IInventoryItem>();  TextHistory = string.Empty;  AllTransfers = new List<ExitsFromEnvironment>();  }  }  } |
| JsonConverter.cs |
| using Expedition.Models;  using Newtonsoft.Json;  using Newtonsoft.Json.Linq;  using System;  public class InventoryItemConverter : JsonConverter  {  public override bool CanConvert(Type objectType)  {    return typeof(IInventoryItem).IsAssignableFrom(objectType);  }  public override void WriteJson(JsonWriter writer, object value, JsonSerializer serializer)  {  var item = (IInventoryItem)value;  JObject jo = new JObject  {  { "Type", item.GetType().Name },  { "Name", item.Name },  { "Description", item.Description },  { "MaximumCount", item.MaximumCount },  { "IsVisible", item.IsVisible },  { "SecondName", item.SecondName }  };  if (item is Food food)  {  jo.Add("Saturation", food.Saturation);  }  else if (item is HealthKit healthKit)  {  jo.Add("RecoverableHealth", healthKit.RecoverableHealth);  }  else if (item is QuestItem questItem)  {  jo.Add("Action", questItem.Action);  }  jo.WriteTo(writer);  }  public override object ReadJson(JsonReader reader, Type objectType, object existingValue, JsonSerializer serializer)  {  JObject jo = JObject.Load(reader);  string type = jo["Type"].ToString();  IInventoryItem item = type switch  {  nameof(Food) => new Food(),  nameof(HealthKit) => new HealthKit(),  nameof(QuestItem) => new QuestItem(),  \_ => throw new NotSupportedException($"Тип '{type}' не поддерживается.")  };  serializer.Populate(jo.CreateReader(), item);  return item;  }  } |
| InitializeGame.cs |
| using Expedition.Models;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using Environment = Expedition.Models.Environment;  namespace Expedition  {  public class InitializeGame  {  public GameModel CreateExpeditionGame(string playerName)  {  \_model = new GameModel();  \_model.Player.Name = playerName;  \_model.SetItems(CreateGameItems());  \_model.SetNPCS(CreateGameNPC());  \_model.SetEnviromnets(CreateGameEnvironments());  \_model.SetTransfers(CreateGameExits());  \_model.SetQuests(CreateGameQuests());  \_model.SetCurrentQuest(0);  \_model.SetCurrentEnvironment(0);  return \_model;  }  GameModel \_model;  List<IInventoryItem> \_items;  List<NPC> \_characters;  List<Environment> \_environments;  List<ExitsFromEnvironment> \_exits;  List<Quest> \_quests;  private List<IInventoryItem> CreateGameItems()  {  \_items = new List<IInventoryItem>();  //Еда  \_items.Add(new Food()  {  Name = "Маленькая булочка",  Description = "Восстанавливает мало сытости(15 ед.), зато легко помещается в карман.",  Saturation = 15,  MaximumCount = 4,  SecondName = "Маленькую булочку"  });  \_items.Add(new Food()  {  Name = "Сытный пирожок",  Description = "Восстанавливает прилично сытости(30 ед.), но для носки не помешает рюкзак.",  Saturation = 30,  MaximumCount = 2  });  //Лечение  \_items.Add(new HealthKit()  {  Name = "Бинт",  Description = "Восстанавливает немного здоровья(4 ед.), незаменимая вещь в путешествии.",  RecoverableHealth = 4,  MaximumCount = 5  });  \_items.Add(new HealthKit()  {  Name = "Аптечка",  Description = "Восстанавливает половину здоровья(10 ед.), самая важная вешь в приключениях, никто же не хочет тут умереть.",  RecoverableHealth = 10,  MaximumCount = 2,  SecondName = "Аптечку"  });  //Квестовые предметы  \_items.Add(new QuestItem()  {  Name = "Рюкзак",  Description = "Это ваш рюкзак, благодаря нему вы можете таскать с собой веши.",  Action = "Теперь вам доступен инвентарь.",  MaximumCount = 1,  IsVisible = false  });  \_items.Add(new QuestItem()  {  Name = "Ноутбук",  Description = "Ваш рабочий ноутбук, вы любите сидеть за ним после учебы. Учеба и развлечения в одном месте, что может быть лучше. Похоже на почте новое <уведомление>",  Action = "Кажется, вам пришло новое уведомление.",  MaximumCount = 0  });  \_items.Add(new QuestItem()  {  Name = "Уведомление",  Description = $"Уведомление из университета. Стоит прочитать, что там написано. Уведомление:\nУважаемый {\_model.Player.Name}, Мы рады сообщить вам о возможности присоединиться к нашей экспедиции по исследованию неизведанных глубин океана на планете 4546B. Ваша страсть к океанологии и ваш академический успех сделали вас идеальным кандидатом для участия в этой экспедиции. Мы ищем смелых и любознательных людей, готовых столкнуться с неизведанным и раскрыть тайны подводного мира. Экспедиция начнется через две недели. Вам предстоит работать в команде с учеными и инженерами, которые разделяют вашу любовь к океану. Это шанс не только проверить свои силы, но и внести свой вклад в научные исследования. С нетерпением ждем вас через две недели в пункте отправки университета!\n",  Action = "Это уведомление мечты, лучше просто и быть не может.",  MaximumCount = 0  });  \_items.Add(new QuestItem()  {  Name = "Оборудование",  Description = "Оборудование необходимое для экспедиции.",  Action = "С этим оборудованием можно отправляться в экспедицию.",  MaximumCount = 1  });  \_items.Add(new QuestItem()  {  Name = "Оранжевый коралл",  Description = "Оранжевый коралл полукруглой формы. Неплохой сувенир.",  Action = "Сувенир.",  MaximumCount = 1  });  \_items.Add(new QuestItem()  {  Name = "Плод зеленых водорослей",  Description = "Большой желтый плод, по форме напоминает каплю воды. Светится.",  Action = "Это, можно сказать, переносная лампочка.",  MaximumCount = 1  });  \_items.Add(new QuestItem()  {  Name = "Непонятное устройство",  Description = "Небольшое устройство, по размерам можно сравнить с буханкой хлеба, по форме же это прямоугольник с зелеными светящимися полосами по периметру.",  Action = "Требуется изучение",  MaximumCount = 1  });  return \_items;  }  private List<NPC> CreateGameNPC()  {  \_characters = new List<NPC>();  var drSmith = new NPC("Др. Смит");  drSmith.AddPhrase("Я занимаюсь анализом данных о морской жизни.", "Институт");  drSmith.AddPhrase("Рад, что тебе получилось попасть сюда. Это ценный опыт для тебя. Надеюсь тебе понравиться экспедиция.", "Спальня");  var Jane = new NPC("Джейн");  Jane.AddPhrase("Я отвечаю за техническое обеспечение нашей команды.", "Институт");  Jane.AddPhrase("Интересный кабинет, правда? Благодаря современным технологиям врачи нам уже не нужны, в подобных кабинетах все полностью автоматизировано. Обычные медикаменты тоже есть, но ими обучают пользоваться в институте, надеюсь ты не пропускал эти занятия.", "Мед. кабинет");  var Alex = new NPC("Алекс");  Alex.AddPhrase("Я изучаю экосистемы океана и помогаю команде с анализом.", "Институт");  Alex.AddPhrase("Тоже ищешь поесть?", "Кухня");  \_characters.Add(drSmith);  \_characters.Add(Jane);  \_characters.Add(Alex);  return \_characters;  }  private List<Environment> CreateGameEnvironments()  {  \_environments = new List<Environment>();  var homeRoom = new Environment("Твоя комната", "Главный герой живет в небольшой, но уютной однокомнатной квартире, где царит атмосфера спокойствия и уюта. В комнате стоит удобная кровать, рядом с которой находится письменный стол с <ноутбуком>, на котором герой работает над своими проектами. На стенах висят фотографии из путешествий и картины с изображениями подводного мира, отражающие его увлечение океанологией. В кухонной зоне расположены необходимые бытовые приборы, а на столе — несколько книг и чашка с горячим чаем. Большие окна пропускают много света и открывают вид на город, создавая ощущение простора и тепла.");  homeRoom.AddItem(GetItem("Рюкзак"));  homeRoom.AddItem(GetItem("Ноутбук"));  homeRoom.AddItem(GetItem("Уведомление"));  \_environments.Add(homeRoom);  var institute = new Environment("Институт", "Институт, где собираются участники экспедиции, представляет собой современное здание с большими стеклянными окнами, через которые проникает яркий солнечный свет. Внутри царит атмосфера активности и ожидания: студенты и ученые обсуждают свои исследования, а в воздухе витает запах свежезаваренного кофе. Помещение для сбора участников — просторный конференц-зал с удобными креслами и большим экраном для презентаций.На стенах висят карты океанских глубин и фотографии прошлых экспедиций, вдохновляющие на новые открытия. Участники собираются за длинным столом, где разложены документы и оборудование для предстоящего погружения. Здесь царит дух сотрудничества и энтузиазма, ведь каждый из них готов внести свой вклад в изучение загадок подводного мира.");  institute.AddItem(GetItem("Оборудование"));  institute.AddCharacter(GetCharacter("Др. Смит"));  institute.AddCharacter(GetCharacter("Джейн"));  institute.AddCharacter(GetCharacter("Алекс"));  \_environments.Add(institute);  var submarine = new Environment("Основной шлюз", "Субмарина, на борту которой участники экспедиции отправляются исследовать глубины океана, представляет собой высокотехнологичное подводное судно, оснащенное всем необходимым для комфортного и безопасного погружения. Внутри субмарины царит атмосфера научного исследования: стены обшиты прочным металлом, а большие иллюминаторы позволяют наблюдать за подводным миром. Основное помещение — это просторный салон с удобными креслами и современным оборудованием для мониторинга состояния окружающей среды.Здесь находятся консоли управления, экраны с данными о глубине и температуре воды, а также карты исследуемых районов.Кухонный уголок предлагает простые, но питательные блюда, которые готовятся на борту. В отдельной каюте участники могут отдохнуть и подготовиться к предстоящим погружениям. Субмарина оборудована всем необходимым для длительных исследований, создавая идеальные условия для изучения загадок океанских глубин.", secondName: "субмарину");  \_environments.Add(submarine);  var bedroom = new Environment("Спальня", "Это спальня, здесь отдыхают участники экспедиции", "спальню");  bedroom.AddCharacter(GetCharacter("Др. Смит"));  \_environments.Add(bedroom);  var kitchen = new Environment("Кухня", "Это кухня, здесь можно запастись едой.", "кухню");  for(int i = 0; i < 500; i++)  {  kitchen.AddItem(GetItem("Маленькая булочка"));  kitchen.AddItem(GetItem("Сытный пирожок"));  }  kitchen.AddCharacter(GetCharacter("Алекс"));  \_environments.Add(kitchen);  var healthRoom = new Environment("Мед. кабинет", "Это мед. кабинет, здесь можно запастись медикаментами");  for (int i = 0; i < 500; i++)  {  healthRoom.AddItem(GetItem("Бинт"));  healthRoom.AddItem(GetItem("Аптечка"));  }  healthRoom.AddCharacter(GetCharacter("Джейн"));  \_environments.Add(healthRoom);  var safeEnv = new Environment("Безопасные отмели", "Это безопасные отмели, место вашей остановки и основная зона в данном океане.");  safeEnv.AddItem(GetItem("Оранжевый коралл"));  safeEnv.SetDangerous(2);  \_environments.Add(safeEnv);    var seaweedZone = new Environment("Зона водорослей", "Ближайшая к безопансым отмелям зона, она заполнена зелеными водорослями с желтыми светящимеся плодами.", "зону водорослей");  seaweedZone.AddItem(GetItem("Плод зеленых водорослей"));  seaweedZone.SetDangerous(2);  \_environments.Add(seaweedZone);  var redDesert = new Environment("Красные пески", "Зона заполненная песками, получила свое нозвание из-за красных растений растущих из песка, в следствие чего вся зона кажется красной. По зоне также растилается множество гор, что делает ее трудной к передвижению на крупном транспорте.");  redDesert.SetDangerous(2);  \_environments.Add(redDesert);  var oldHouse = new Environment("Древний храм", "Древний подводных храм. Судя по всему его построили прошлые обитатели планеты, они были весьма разумны, чтобы построить такое. Перед вам открывается большой холл и вы видите развилку из двух коридоров.");  oldHouse.SetDangerous(2);  \_environments.Add(oldHouse);    var leftCorridor = new Environment("Левая комната", "Вы видите перед собой большую комнату. По периметру стен стоят каменные плиты и небольшие коробки из того же камня. Похоже это кровати с тумбочками, отсюда можно сделать вывод, что это общежитие, но, правда, больше походит на военный барак.", "левую комнату");  leftCorridor.SetDangerous(2);  \_environments.Add(leftCorridor);  var rightCorridor = new Environment("Правый коридор", "Длинный коридор с тусклым зеденым свеченим из под потолка, в конце вы видите переход в другую комнату, но из-за света не можете разглядеть что там.");  rightCorridor.SetDangerous(2);  \_environments.Add(rightCorridor);  var strangeRoom = new Environment("Странная комната", "Вы видите перед собой непонятную комнату, размера она примерно 3 метра в каждом параметре, зеленый свет из под потолка здесь уже достаточно ярко освещает комнату, в центре стоит подобите стола и на нем дежит <непонятное устройство>", "странную комнату");  strangeRoom.AddItem(GetItem("Непонятное устройство"));  strangeRoom.SetDangerous(2);  \_environments.Add(strangeRoom);  return \_environments;  }  private IInventoryItem GetItem(string itemName)  {  return \_items.Where(x => x.Name == itemName).FirstOrDefault();  }  private NPC GetCharacter(string characterName)  {  return \_characters.Where(x => x.Name == characterName).FirstOrDefault();  }  private List<ExitsFromEnvironment> CreateGameExits()  {  \_exits = new List<ExitsFromEnvironment>();  var homeRoomExits = new ExitsFromEnvironment("Твоя комната");  homeRoomExits.AddExit(GetEnvironment("Институт"));  \_exits.Add(homeRoomExits);  var institutExits = new ExitsFromEnvironment("Институт");  institutExits.AddExit(GetEnvironment("Основной шлюз"));  \_exits.Add(institutExits);  var submarineExits = new ExitsFromEnvironment("Основной шлюз");  submarineExits.AddExit(GetEnvironment("Спальня"));  submarineExits.AddExit(GetEnvironment("Кухня"));  submarineExits.AddExit(GetEnvironment("Мед. кабинет"));  submarineExits.AddExit(GetEnvironment("Безопасные отмели"));  \_exits.Add(submarineExits);  var bedroomExits = new ExitsFromEnvironment("Спальня");  bedroomExits.AddExit(GetEnvironment("Основной шлюз"));  \_exits.Add(bedroomExits);  var kitchenExits = new ExitsFromEnvironment("Кухня");  kitchenExits.AddExit(GetEnvironment("Основной шлюз"));  \_exits.Add(kitchenExits);  var healthRoomExit = new ExitsFromEnvironment("Мед. кабинет");  healthRoomExit.AddExit(GetEnvironment("Основной шлюз"));  \_exits.Add(healthRoomExit);  var safeZoneExits = new ExitsFromEnvironment("Безопасные отмели");  safeZoneExits.AddExit(GetEnvironment("Основной шлюз"));  safeZoneExits.AddExit(GetEnvironment("Зона водорослей"));  safeZoneExits.AddExit(GetEnvironment("Красные пески"));  \_exits.Add(safeZoneExits);  var seaweedZoneExits = new ExitsFromEnvironment("Зона водорослей");  seaweedZoneExits.AddExit(GetEnvironment("Безопасные отмели"));  \_exits.Add(seaweedZoneExits);  var redDesertExits = new ExitsFromEnvironment("Красные пески");  redDesertExits.AddExit(GetEnvironment("Безопасные отмели"));  redDesertExits.AddExit(GetEnvironment("Древний храм"));  \_exits.Add(redDesertExits);  var oldHouseExits = new ExitsFromEnvironment("Древний храм");  oldHouseExits.AddExit(GetEnvironment("Красные пески"));  oldHouseExits.AddExit(GetEnvironment("Левая комната"));  oldHouseExits.AddExit(GetEnvironment("Правый коридор"));  \_exits.Add(oldHouseExits);  var leftRoomExits = new ExitsFromEnvironment("Левая комната");  leftRoomExits.AddExit(GetEnvironment("Древний храм"));  \_exits.Add(leftRoomExits);  var rightCorridorExits = new ExitsFromEnvironment("Правый коридор");  rightCorridorExits.AddExit(GetEnvironment("Древний храм"));  rightCorridorExits.AddExit(GetEnvironment("Странная комната"));  \_exits.Add(rightCorridorExits);  var strangeRoomExits = new ExitsFromEnvironment("Странная комната");  strangeRoomExits.AddExit(GetEnvironment("Правый коридор"));  \_exits.Add(strangeRoomExits);  return \_exits;  }  private Environment GetEnvironment(string environmentName)  {  return \_environments.Where(x => x.Name == environmentName).FirstOrDefault();  }  private List<Quest> CreateGameQuests()  {  \_quests = new List<Quest>();  var firstQuest = new Quest("Прибытие домой", "После долгого учебного дня вы возвращаетесь домой, уставший, но довольный. Как студент-исследователь в области океанологии, вы провели часы на лекциях и лабораторных занятиях, обсуждая с однокурсниками последние открытия в области подводных исследований.");  var firstQuestSteps = new List<QuestStep>()  {  new QuestStep(new List<string>() { "осмотреться" }, "По прыбитию домой, не мешало бы осмотреться. С таким количеством учебы можно и забыть как родной дом выглядит. (Вы можете ввести команду \"осмотреться\" - она позволяет увидеть описание окружения.)", isMovement: false),  new QuestStep(new List<string>() { "посмотреть на ноутбук", "посмотреть на уведомление" }, "Целый день без <ноутбука>, стоит проверить что там, да как.(Команда \"посмотреть на\" позволяет увидеть описание объекта; интерактивные объекты выделяются фигурными скобками: <объект>)", isMovement: false),  new QuestStep(new List<string>() { "взять рюкзак" }, "Надеюсь время быстро пролетит. \n\nСпустя две недели...\n\nУже прошли две недели, пора собираться на выход. Нужно взять <рюкзак>.(Команада \"взять\" позволяет брать предметы в руки)", isMovement: false),  new QuestStep(new List<string>() { "перейти в институт" }, "Вещи собраны, пора идти в <институт>.(Для перемешения используйте команду \"перейти в\")", new List<IInventoryItem>() { GetItem("Рюкзак")})  };  firstQuest.SetSteps(firstQuestSteps);  var neccessaryFirstSteps = new List<QuestStep>()  {  firstQuestSteps[2],  firstQuestSteps[3]  };  firstQuest.AddEnding("Ежедневыне поездки на общественном транспорте ваше любимое занятие по утрам.", neccessaryFirstSteps);  \_quests.Add(firstQuest);  var instituteQuest = new Quest("Сбор в институте", "Небольшая поедка на общественном транспорте и вот вы уже в институте, место уже столь знакомое, но каждый раз вы любите осматриваеть его и наслаждаетесь местной красотой, сделаете ли вы это вновь?", items: new List<IInventoryItem>() { GetItem("Оборудование") });  var instutueQuestSteps = new List<QuestStep>()  {  new QuestStep(new List<string>() { "исследовать" }, "Стоит понять, что я могу здесь сделать.(Команда \"исследовать\" покажет все предметы взаимодействия в округе)", isMovement: false),  new QuestStep(new List<string>() { "поговорить с др. смит", "поговорить с джейн", "поговорить с алекс" }, "Не помешало бы познакомиться со всеми участниками экспедиции.(Команда \"поговорить с\" позволит поговорить с выбранным персонажем)", isMovement: false),  new QuestStep(new List<string>() { "взять оборудование" }, "Теперь вы со всеми знакомы, необходимо взять <оборудование> для экспедиции.", isMovement: false),  new QuestStep(new List<string>() { "перейти в субмарину" }, "С этим оборудованием можно спокойно садиться в <субмарину> и готовиться к эскпедиции")  };  instituteQuest.SetSteps(instutueQuestSteps);  var neccessaryInstituteSteps = new List<QuestStep>()  {  instutueQuestSteps[1],  instutueQuestSteps[2]  };  instituteQuest.AddEnding("Вы отправились в экспедицию", neccessaryInstituteSteps);  \_quests.Add(instituteQuest);  var submarineQuest = new Quest("Осмотр субмарины", "Ваше приключение начинается и теперь вы остановились посреди океана, чтобы исследовать окрестности.");  var submarineQuestSteps = new List<QuestStep>()  {  new QuestStep(new List<string>() { "перейти в спальню", "перейти в кухню", "перейти в мед. кабинет" }, "Субмарина удивительное средство передвижения, обустроенное для полного удобства при экспедиции. Стоит пройтись посмотреть все комнаты.(Чтобы узнать все переходы с локации можно воспользоваться командой \"выходы\")"),  new QuestStep(new List<string>() { "взять маленькую булочку", "взять сытный пирожок", "взять бинт", "взять аптечку" }, "Перед исследованием ближайшей месности стоит запастись едой и медикаментами."),  new QuestStep(new List<string>() { "перейти в безопасные отмели" }, "Теперь вы полностью снаряжены, можно отправляться на исследование местности. (Чтобы использовать собранные предметы используйте команду \"использовать\")")  };  submarineQuest.SetSteps(submarineQuestSteps);  var neccessarySubmarineSteps = new List<QuestStep>()  {  submarineQuestSteps[0],  submarineQuestSteps[1]  };  submarineQuest.AddEnding("Время опробовать себя в исследовании местности.", neccessarySubmarineSteps);  \_quests.Add(submarineQuest);  var explorationQuest = new Quest("Ислледование местности", "Вы впервые спустились в воду для исследования, вам переполняют чувства. Давайте же узнаем, что скрывают эти воды.");  var explorationQuestSteps = new List<QuestStep>()  {  new QuestStep(new List<string>() { "перейти в красные пески" }, "Стоит осмотреть близлежайшую местность, может получится найти что-то интересное."),  new QuestStep(new List<string>() { "перейти в древний храм" }, $"Вам на КПК пришло сообщение: {\_model.Player.Name}, послушай, это Др. Смит, неподалеку от тебя обнаружили странную активность, что-то похожее на древний храм. Раз ты ближе всех, посмотри, пожалуйста, что там происходит."),  };  explorationQuest.SetSteps(explorationQuestSteps);  var neccessaryExplorationSteps = new List<QuestStep>()  {  explorationQuestSteps[1]  };  explorationQuest.AddEnding("Вы попали в древний храм, нужно исслеовать его, чтобы разобраться в ситуации.", neccessaryExplorationSteps);  \_quests.Add(explorationQuest);  var endQuest = new Quest("Осмотреть древний храм", "По просьбе Др. Смита вы вошли в древний храм. Найдите источник странного сигнала и выясните что это.");  var endQuestSteps = new List<QuestStep>()  {  new QuestStep(new List<string>() { "перейти в правый коридор" }, "Стоит осмотреть здесь все."),  new QuestStep(new List<string>() { "перейти в странную комнату" }, "Впереди видно комнату, ее обязательно следует проверить."),  new QuestStep(new List<string>() { "посмотреть на непонятное устройство" }, "Стоит проверить это устройство."),  new QuestStep(new List<string>() { "" }, "Тебе стоит выбрать <взять> устройство и отнести его Др. Смиту или уйти отсюда и промолчать о находке. Что ты выберешь?")  };  endQuestSteps[3].SetMoralChoice(new MoralChoice("взять непонятное устройство", "перейти в правый коридор", "Вы взяли устройство и вернулись на субмарину. Др. Смит решил сразу же обследовать устройство и обнаружил, что в нем заключен вирус, а оболочка нестабильна. Если бы не ваши действия вирус мог бы выбраться наружу, а теперь вы можете спокойно исследовать океан дальше.", "Вы решили не трогать устройство и вернуться на базу. Др. Смиту вы сказали, что ничего не нашли. Через некоторое время вы начали замечать мертвых существ, а после и сами по очереди замечаете у себя признаки неизвестной болезни. Вы не можете вернуться в город за помощью и теперь вам придется понять, что же вы такое проигнорировали в том храме."));  endQuest.SetSteps(endQuestSteps);  \_quests.Add(endQuest);  return \_quests;  }  }  } |