Частное учреждение образование

«Колледж бизнеса и права»

ОТЧЕТ ПО ПРАКТИКЕ

ПО РАЗРАБОТКЕ И СОПРОВОЖДЕНИЮ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ОП Т.693011.401

Выполнил К.В. Максимук

Руководители практики Н.В. Ржеутская

2019

Содержание

[1 Программа практики 3](#_Toc9847208)

[2 Объектно-ориентированный анализ и проектирование системы 3](#_Toc9847209)

[2.1 Сущность задачи 4](#_Toc9847210)

[2.2 Проектирование модели 5](#_Toc9847211)

[3 Вычислительная система 12](#_Toc9847212)

[3.1 Требования к аппаратным и операционным ресурсам 12](#_Toc9847213)

[3.2 Инструменты разработки 12](#_Toc9847214)

[4 Проектирование задачи 14](#_Toc9847215)

[4.1 Требование к приложению 14](#_Toc9847216)

[4.2 Концептуальный прототип 15](#_Toc9847217)

[4.3 Организация данных 16](#_Toc9847218)

[4.4 Функции и элементы управления 16](#_Toc9847219)

[4.5 Проектирование справочной системы приложения 25](#_Toc9847220)

[5 Описание программного средства 26](#_Toc9847221)

[5.1 Функциональное назначение 26](#_Toc9847222)

[5.2 Функциональное назначение 26](#_Toc9847223)

[5.3 Входные данные 27](#_Toc9847224)

[5.4 Выходные данные 27](#_Toc9847225)

[6 Методика испытаний 28](#_Toc9847226)

[6.2 Порядок проведения испытаний 28](#_Toc9847227)

[6.2.1 Функциональное тестирование 28](#_Toc9847228)

[6.2.2 Полное тестирование 28](#_Toc9847229)

[7 Применение 29](#_Toc9847230)

[7.1 Назначение программы 29](#_Toc9847231)

[7.2 Условия применения 29](#_Toc9847232)

[7.3 Справочная система 29](#_Toc9847233)

[Заключение 30](#_Toc9847234)

[Приложение А 32](#_Toc9847235)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ОП Т.693011.401

Разраб.

*К.В. Максимук*

Провер.

Н.В. Ржеутская

Т. Контр.

Н. Контр.

Утверд.

Отчет по практике

по разработке и сопровождению

Лит.

Листов

11

КБиП

# 1 Программа практики

Целями практики по разработке и сопровождению программного обеспечения являются:

- закрепление знаний, связанных с технологией обработки информации;

- приобретение навыков разработки программ с использованием современных средств обработки экономической и деловой информации;

- выработка умений применять средства стандартных библиотек в практических задачах.

Задачами практики по разработке и сопровождению ПО являются:

- углубленное изучение принципов организации программного обеспечения и технологии его проектирования;

- владение методами надежного программирования;

- умение разрабатывать программы в соответствии с промышленными требованиями, обеспечивая высокий уровень качества программного обеспечения и экономической эффективности;

- изучение способов определения экономической себестоимости и эффективности разработки программного обеспечения;

- оформление комплекта документации на созданное программное обеспечение.

Таблица 1 - Календарный график работы

|  |  |
| --- | --- |
| Период | Наименование и содержание работ |
| 09.02.19 | Изучение требований к рабочей дисциплине и технике безопасности.  Изучение нормативных документов. |
| 11.02.19-  17.02.19 | Ознакомление с различным программным обеспечением, используемым для разработки программ |
| 18.02.19-  24.02.19 | Получение индивидуальных заданий по созданию программ для обработки экономической и деловой информации. Обсуждение тем индивидуальных заданий. |
| 25.02.19-  03.03.19 | Изучение общих требований к программному обеспечению обработки экономической и деловой информации. Требования к интерфейсу пользователя, выходным документам (печатные, экранные формы, форматы результирующих файлов с результатами расчетов). |
| 04.03.19-  24.03.19 | Анализ предметной области индивидуального задания. Выбор и обоснование языка и среды программирования. Описание функциональных возможностей задачи. |
| 25.03.19-  07.04.19 | Технология создания программного обеспечения. Правила построения диаграмм UML. Проектирование ПО при помощи CASE-средства Rational Rose |
| 08.04.19-  21.04.19 | Конструирование алгоритма и его реализация в выбранной визуальной среде программирования. Разработка интерфейса .Проектирование классов, определение свойств и методов, определение связей между классами. |
| 22.04.19-  05.05.19 | Создание информационной базы задачи. Тестирование и отладка программы с использованием подготовленных тестовых примеров. Проектирование модулей, позволяющих осуществить защиту программного продукта от несанкционированного доступа.. |
| 06.05.19-  26.05.19 | Критерии оценки качества программного продукта. Изучение нормативных документов, регламентирующих оформление технического проекта. Создание документации по эксплуатации и сопровождению разработанного ПС. Проведение испытания ПС в нормальных и критических условиях.. |
| 27.05.19-  02.06.19 | Анализ возможности модернизации проекта с учетом альтернативных решений. Оформление аннотации на программное средство. |
| 03.06.19-  16.06.19 | Оформление отчета по практике. Подготовка листингов программ с дополнением комментариев. Распечатка экранных форм. |
| 17.06.19-  26.06.19 | Демонстрация программ по индивидуальным заданиям.  Защита отчета по практике. |

# 2 Объектно-ориентированный анализ и проектирование системы

## 2.1 Сущность задачи

Сущностью задачи является написание программы-игры «Герои и подземелья». Игра жанра пошаговой текстовой РПГ.

Компьютерная ролевая игра (англ. Computer Role-Playing Game, обозначается аббревиатурой CRPG или RPG) — жанр компьютерных игр, основанный на элементах игрового процесса традиционных настольных ролевых игр. В ролевой игре игрок управляет одним или несколькими персонажами, каждый из которых описан набором численных характеристик, списком способностей и умений; примерами таких характеристик могут быть хит-пойнты (англ. hit points, HP, ХП), показатели силы, ловкости, интеллекта, защиты, уклонения, уровень развития того или иного навыка.

У жанра RPG много общего с настольными ролевыми играми наподобие Dungeons and Dragons — жаргон, сеттинги, геймплейная механика. Обычно игрок управляет одним или несколькими главными героями («партией»), и добивается победы, выполняя задания («квесты»), участвуя в тактических боях и доходя до самого конца сюжета.

Примеры RPG: The Ender Scrolls 5, Ведьмак III: Дикая Охота, The World of Warcraft.

Пошаговая стратегия (англ. Turn-Based Strategy, TBS) — поджанр компьютерных стратегических игр, в которых игровой процесс состоит из последовательности фиксированных моментов времени, именуемых ходами (или шагами), во время которых игроки совершают свои действия.

Основной характеристикой пошаговых стратегических игр является дискретность игрового процесса. Игра состоит из фиксированных во времени моментов («шагов» или «ходов»), которые завершаются только по команде игрока. Во время этих ходов игрок совершает свои действия. Один ход может соответствовать промежутку во много лет в игровом мире, за которые игрок успевает управиться с событиями в каждом городе империи и отдать приказы сотням военных отрядов.

В большинстве пошаговых стратегий игроки совершают ходы по очереди, как в таких классических настольных играх, как шахматы и Риск.

Примерами пошаговых стратегий: Sid Meier's Civilization и Heroes of Might and Magic.

Игра Interactive fiction (дословно — интерактивная художественная литература; IF; текстовые квесты; adventure — приключенческая игра) — разновидность компьютерных игр, в которых взаимодействие с игроком осуществляется посредством текстовой информации.

Существуют два вида интерфейса:

* интерфейс с вводом текста с клавиатуры;
* интерфейс в виде меню, где игрок выбирает действие из нескольких предложенных (CYOA — Choose Your Own Adventure).

Choose Your Own Adventure (рус. Выбери себе приключение) — серия детских книг-игр, впервые опубликованная Bantam Books между 1979—1998 годами, и ныне выходящая в издательстве Chooseco. Каждая история написана от второго лица, где игрок выступает в роли протагониста и должен выбирать нужное со своей точки зрения действие, влияющее на дальнейшее прохождение.

После предисловия читателю даётся возможность выбрать первые шаги для главного героя книги. Например, в «Пещере времени» в начале предстоит выбор из двух пунктов:

* если вы хотите вернуться домой, идите на страницу 4.
* если вы хотите подождать, идите на страницу 5.

После того, как читатель сделал свой выбор, начинается сама история, в течение которой необходимо множество раз определяться с действиями, и заканчивающаяся множеством различных концовок.

Существует несколько типов возможных концовок книги:

* обычно одно, иногда несколько, наиболее выгодное окончание, полностью раскрывающее сюжет.
* концовки, заканчивающиеся смертью персонажа, его друзей или всех вместе, а также очень плохая концовка, в результате ошибочного выбора читателя.
* прочие концовки, не полностью раскрывающие сюжет (могут быть как относительно положительные, так и относительно отрицательные)
* иногда читатель попадает в так называемую петлю, возвращаясь по кругу к одной и той же странице. Выход в такой ситуации — это начать читать заново.
* в одной из книг — Inside UFO 54-40, по сюжету необходимо было достичь рая, но ни один из возможных переходов в книге не приводил к нужному параграфу, хотя в самом тексте таковой имелся. Пройти книгу-игру можно было лишь обманным путём.

Ранние книги позволяли читателю бо́льшую свободу действий, так, например, выбирать, будет ли стучащий в дверь незнакомец добрым или злым, однако, позднее в книгах оставили выбор действий только самого игрока.

По мере расширения серии длина историй увеличивалась, а выбор концовок, наоборот, уменьшался. Самые первые книги содержали до 40 возможных финалов, в то время как последние не более восьми.

Примеры текстовых RPG: Tale, Galatea, Zork, Curses.

## 2.2 Проектирование модели

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями. Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD), которые предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают некоторый набор стандартных обозначений для определения данных и отношений между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними, имеющих важное значение для разрабатываемой системы. Основными понятиями данной нотации являются понятия сущности и связи. При этом под сущностью понимается произвольное множество реальных или абстрактных объектов, каждый из которых обладает одинаковыми свойствами и характеристиками. В этом случае каждый рассматриваемый объект может являться экземпляром одной и только одной сущности, должен иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности. Связь определяется как отношение или некоторая ассоциация между отдельными сущностями. Примерами связей могут являться родственные отношения типа «отец-сын» или производственные отношения типа «начальник-подчиненный». Другой тип связей задается отношениями «иметь в собственности» или «обладать свойством».

Графическая модель данных строится таким образом, чтобы связи между отдельными сущностями отражали не только семантический характер соответствующего отношения, но и дополнительные аспекты обязательности связей, а также кратность участвующих в данных отношениях экземпляров сущностей. Информационная модель базы представлена на диаграмме «Сущность-связь». Данная диаграмма представлена на рисунке 2.1.

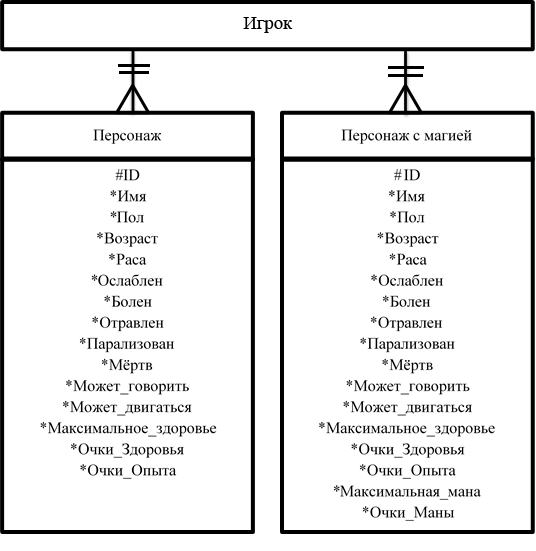


Рисунок 2.1- Диаграмма «сущность-связь» в нотации Баркера

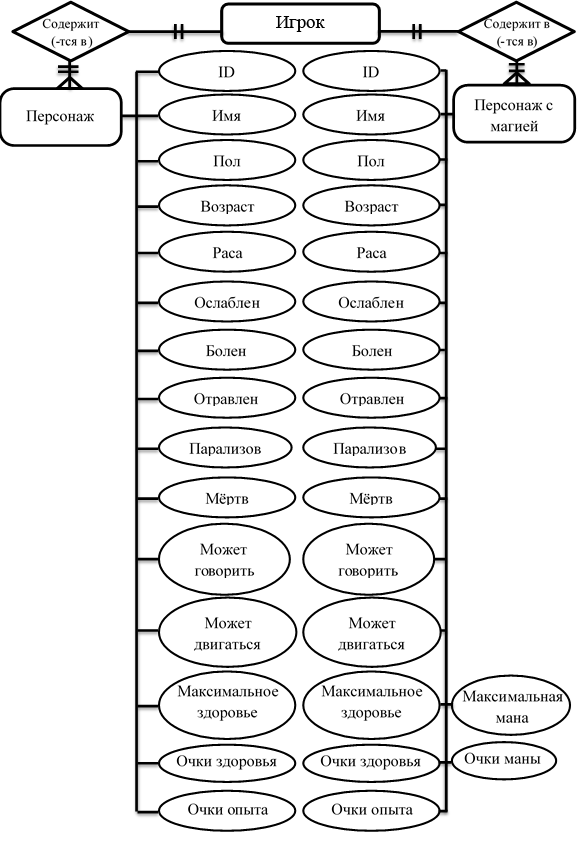


Рисунок 2.2 - Диаграмма ERD в нотации Ричарда Чена

Исходя из исследования предметной области, можно выделить следующие сущности разработки: игрок, персонаж, персонаж с магией.

У сущности «Игрок» нет выделяемых атрибутов

Для сущности «Персонаж» можно выделить следующие атрибуты:

- id;

- имя;

- пол;

- возраст;

- раса;

- ослаблен;

- болен;

- отравлен;

- парализован;

- мёртв;

- может говорить;

- может двигаться;

- максимальное здоровье;

- очки здоровья;

- очки опыта.

Для сущности «Персонаж с магией» можно выделить следующие атрибуты:

- id;

- имя;

- пол;

- возраст;

- раса;

- ослаблен;

- болен;

- отравлен;

- парализован;

- мёртв;

- может говорить;

- может двигаться;

- максимальное здоровье;

- очки здоровья;

- очки опыта;

- максимальная мана;

- очки маны.

Суть диаграммы вариантов использования состоит в том, что проектируемая система представляется в виде множества сущностей или актёров, взаимодействующих с системой с помощью, так называемых, вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

Актёр представляет собой внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует её функциональные возможности для достижения определённых целей или решения частных задач. При этом актёры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актёр может рассматриваться как некоторая отдельная роль относительно конкретного варианта использования.

Данное программное средство имеет следующие основные (Include) функции:

* Файл сохранения;
* Список предметов инвентаря;
* Сценарии развития событий;
* Текущий сценарий развития событий;
* Список наград;
* Текущее количество рун;
* Список заклинаний;
* Количество маны для выбранного персонажа;
* Список существующих героев.

К вспомогательным (Extend) функциям, расширяющим возможности системы, относятся следующие функции:

* Взаимодействие с программой
* Сохранить игру;
* Загрузить игру;
* Взаимодействовать с инвентарем;
* Изменение текущего героя на следующего по списку;
* Создать персонажа (-жей);
* Посмотреть информацию о текущем персонаже;
* Отправиться на приключения;
* Изменить состояние персонажа(-жей);
* Использовать заклинание;
* Получать награды;
* Сражаться с врагами.

Диаграмма вариантов использования представлена в графической части на листе 1.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

В данном курсовом проекте будут реализованы классы и их методы, представлены в таблице 2.

Таблица 2 - Классы

|  |  |  |
| --- | --- | --- |
| Класс | Поля | Методы |
| MainWindow | AddHeroWindow, adventureScripts, AdventureScriptsNonRepeat, AttackDialog, ConsoleBuffer, InventoryWindow, NextAdventureIs, МножительСилыРун, Обладает\_магией, ,ПереченьПредметов, Персонажи, ПорядокАтаки, СписокГероевВЗащите, СписокТекущихВрагов, ТекущийПерсонаж, IsBattleMode | Block\_Button\_Choice\_1, Block\_Button\_Choice\_2, Block\_Button\_Choice\_3, Button\_Choice\_1\_Click, Button\_Choice\_2\_Click, Button\_Choice\_3\_Click, Button\_GiveUp\_Click, Button\_Inventory\_Click, Button\_Load\_Click, Button\_Save\_Click, Change\_Button\_Choice\_1, Change\_Button\_Choice\_2, Change\_Button\_Choice\_3, ConsoleWriteLine, DoLoadFile, Get\_Button\_Choice\_1, Get\_Button\_Choice\_2, Get\_Button\_Choice\_3, |

Продолжение таблицы 2 - Классы

|  |  |  |
| --- | --- | --- |
| Класс | Поля | Методы |
| MainWindow |  | MainWindow, MainWindow\_Button\_AddHero\_Click, MainWindow\_Button\_GoForAdventure\_Click, MainWindow\_Button\_ShowInfo\_Click, MainWindow\_Button\_SwitchHero\_Click, SetRuneRank, UnBlock\_Button\_Choice\_1, UnBlock\_Button\_Choice\_2, UnBlock\_Button\_Choice\_3 |
| AddHero | IsAgeAcceptable, IsMagicChecked, IsNameAcceptable, RecievedID, МножительСилыРун, Новый\_персонаж, Новый\_персонаж\_с\_магией, Возраст, Имя, Пол, Раса | AddHero, Button\_Accept\_Click, Button\_Accept\_MouseEnter, CheckAllRequaredFields, Hero\_DoHaveMagic\_Click, Hero\_DoNotHaveMagic\_Click, RadioButton\_Hero\_Female\_Click, RadioButton\_Hero\_Female\_MouseEnter, RadioButton\_Hero\_Male\_Click, RadioButton\_Hero\_Male\_MouseEnter, RadioButton\_Race\_Dwarf\_Checked, RadioButton\_Race\_Dwarf\_MouseEnter, RadioButton\_Race\_Elf\_Checked, RadioButton\_Race\_Elf\_MouseEnter, RadioButton\_Race\_Goblin\_Checked, RadioButton\_Race\_Goblin\_MouseEnter, RadioButton\_Race\_Human\_Checked, RadioButton\_Race\_Human\_MouseEnter, RadioButton\_Race\_Ork\_Checked, RadioButton\_Race\_Ork\_MouseEnter, StackPanel\_MouseEnter, TextBlock\_Change, TextBlock\_Clear, TextBox\_MouseEnter, TextBox\_NameOfHero\_TextChanged, TextBox\_TextChanged, Получить\_Персонажа, |
| Attack | АтакующийПерсонаж, СписокТекущихВрагов, IsEnemyKilled, SelectedEnemy, ЗначениеАтаки | Attack, Button\_Attack\_Click, ComboBox\_EnemyList\_SelectionChanged, GetСписокТекущихВрагов |
| Персонаж | МножительСилыРун, Состояние, ID, Возраст, Имя, Максимальное\_здоровье, Максимальное\_здоровье\_Начальное, Может\_говорить, Может\_двигаться, Очки\_Здоровья, Очки\_Опыта, Пол, Раса | RuneUpdate, ToString, Дебаффы, Добавить\_ОчкиОпыта, ИзменениеСостоянияЗдоровья, ИзменениеСостоянияЭффектов, Лет\_Лета\_Год, ПолучитьЗначениеАтаки Установить\_константы |
| Персонаж\_с\_магией | Максимальная\_мана, Очки\_Маны | ToString, ИзменениеСостоянияМаны, Персонаж\_с\_магией, Персонаж\_с\_магией, Установить\_константы |
| AdventureScripts | Choices, MainWindow, Choices\_GetLast, OrderOf | AdventureScripts, AreEnemiesAlive, AreHeroesAlive, Battle, Choices\_Add, GameOver, IncreaseOrderOf, Regen, Воспроизведение\_Шагов, Начало\_приключения |

Продолжение таблицы 2 - Классы

|  |  |  |
| --- | --- | --- |
| Класс | Поля | Методы |
| Заклинание | Название, Стоимость | – |
| Воскрешение | Название, Стоимость | Воскрешение |
| Заморозка | Название, Стоимость | Заморозка |
| Лечение | Название, Стоимость, Сила\_Воздействия | Лечение |
| Огненный\_Шар | Название, Стоимость | Огненный\_Шар |
| Magic | OrderOf, МножительСилыРун, ПорядокАтаки, СписокВрагов, СписокПерсонажей, ConsoleOutput | Button\_Использовать\_Click, ComboBox\_ВыборЗаклинания\_SelectionChanged, ComboBox\_ЦельЗаклинания\_SelectionChanged, Magic |
| Враги | IsFrozen, ЗдоровьеВрага, ИмяВрага, МаксимальноеЗдоровье, СилаАтаки | ОтнятьЗдоровье |
| ГлаваСтражи | IsFrozen, ЗдоровьеВрага, ИмяВрага, МаксимальноеЗдоровье, СилаАтаки | – |
| Inventory | НазванияПредметов, ПереченьПредметов, Персонажи | Button\_Use\_Click, GetInventory, Inventory, ReloadInventoryList, SearchForItem, UpdateFormWindow, Выпадающий\_Список\_Персонажей\_SelectionChanged, Список\_Предметов\_SelectionChanged |
| Банка\_маны | Количество\_Восполняемой\_Маны, Название | Банка\_маны |
| Большое\_Зелье\_Лечения | Количество\_Восполняемого\_Здоровья, Название | Большое\_Зелье\_Лечения |
| Бутылек\_Маны | Количество\_Восполняемой\_Маны, Название | Бутылек\_Маны |
| Зелье | Название | – |
| Зелье\_Лечения | Количество\_Восполняемого\_Здоровья | – |
| Зелье\_маны | Количество\_Восполняемой\_Маны | – |
| Малое\_Зелье\_Лечения | Количество\_Восполняемого\_Здоровья, Название | Малое\_Зелье\_Лечения |
| Среднее\_Зелье\_Лечения | Количество\_Восполняемого\_Здоровья, Название | Среднее\_Зелье\_Лечения |
| Фласка\_маны | Количество\_Восполняемой\_Маны, Название | Фласка\_маны |

Диаграмма классов для проектируемой системы представлена в графической части на листе 2.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому.

Основная цель использования диаграмм деятельности - визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения.

Расписать - Диаграмма деятельности построена для конкретной функции или для всей программы полностью

Диаграмма деятельности представлена в графической части на листе 3.

Для моделирования взаимодействия объектов в UML используются соответствующие диаграммы взаимодействия. Если рассматривать взаимодействия объектов во времени, тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности.

Временной аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения: одно - слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии; второе - вертикальная временная ось, направленная сверху вниз, на которой начальному моменту времени соответствует самая верхняя часть диаграммы.

Диаграмма последовательности для проектируемой системы представлена в графической части на листе 4.

Рассмотренные ранее диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления. Особенность логического представления заключается в том, что оно оперирует понятиями, которые не имеют самостоятельного материального воплощения. Другими словами, различные элементы логического представления, такие как классы, ассоциации, состояния, сообщения, не существуют материально или физически. Они лишь отражают наше понимание структуры физической системы или аспекты ее поведения.

Основное назначение логического представления состоит в анализе структурных и функциональных отношений между элементами модели системы. Однако для создания конкретной физической системы необходимо некоторым образом реализовать все элементы логического представления в конкретные материальные сущности. Для описания таких реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели.

Диаграмма компонентов описывает объекты реального мира - компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами.

Вид диаграммы компонентов для данной проектируемой системы представлен в графической части на листе 5 и содержит следующие компоненты:

* файл программы «Ролевая игра (WPF).exe»:
* файл сохранения прогресса игры «SaveFile.svfl»
* файл иконок для кнопок программы «\*.png»:
* файл форм «\*.resx»:
* файл форм «\*.cs»;
* файл справочной системы «HelpFile.сhm».

# 3 Вычислительная система

## 3.1 Требования к аппаратным и операционным ресурсам

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению персонального компьютера, являются:

* процессор 800 МГц и выше;
* оперативная память 128 Мбайт и более;
* свободное место на диске 5 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более;
* монитор;
* мышь, клавиатура;

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и выше. Наиболее удобной операционной системой для проведения испытаний является Windows 10, так как она ориентированна на максимальное использование всех возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

## 3.2 Инструменты разработки

Инструментами разработки для будущего программного приложения будут являться:

* операционная система Widows 10;
* программная среда разработки Microsoft Visual Studio 2019;
* язык программирования C#;
* MS Visio 2019;
* MS Word 2019;
* Dr.Explain.

Операционная система — это набор управляющих программ, предназначенных для управления ресурсами вычислительной системы как единого комплекса, другими словами, операционная система – это набор программного обеспечения, который обеспечивает работу компьютера.

При разработке программного средства использовалась операционная система Windows 10, так как на данный момент эта операционная система является самой распространённой операционной системой. В Windows 10 были исправлены практически все недостатки предыдущих операционных систем. Аппаратные требования Windows 10 скромнее, она способна работать даже на маломощных компьютерах и ещё добавлено множество функций, существенно облегчающих работу за компьютером.

Visual Studio меняет отношение к процессу разработки, делая его увлекательным и хорошо организованным. Новый продукт отличается повышенной скоростью загрузки рабочей среды и открывает разработчикам доступ к конкретным проектам буквально в считанные секунды. Кроме того, все длительные процессы выполняются в Visual Studio в фоновом режиме, что не замедляет скорость работы среды и не отвлекает разработчика от основных задач.

Также рабочая среда Visual Studio имеет новый контекстно-зависимый интерфейс. Главная его особенность заключается в том, что он предлагает разработчику только те функции и инструменты, которые ему нужны на данном этапе работы. Таким образом, панель инструментов не содержит ничего лишнего и не затрудняет поиск нужных функций.

Visual Studio позволяет эффективно управлять полным жизненным циклом приложения от этапа его разработки до стадии эксплуатации. Такой подход предполагает командную работу и участие в процессе большого количества специалистов разного профиля: от архитекторов и разработчиков до дизайнеров и заказчиков проекта.

Консолидацию всех циклов работы над приложением и взаимодействие рабочей группы в Visual Studio обеспечивает усовершенствованное решение Team Foundation Server. С его помощью все участники процесса разработки могут отслеживать состояние проекта, видеть его динамику, контролировать сроки и получать аналитические отчёты о каждом периоде работы.

Кроме того, Visual Studio содержит обновлённые инструменты проверки качества и работоспособности приложения, что позволяет тестировщикам моделировать поведение приложения в момент его использования, а также вовремя обнаруживать недочёты в разработке. А функция PowerPoint StoryBoarding позволяет техническим специалистам представлять макет будущего решения заказчику в понятном для него формате.

С# — это язык программирования, синтаксис которого очень похож на синтаксис Java (но не идентичен ему). Например, в С# (как в Java) определение класса состоит из одного файла (\*.cs), в отличие от C++, где определение класса разбито на заголовок (\*.h) и реализацию (\*.срр). Однако называть С# клоном Java было бы неверно. Как С#, так и Java основаны на синтаксических конструкциях C++. Если Java во многих отношениях можно назвать очищенной версией C++, то С# можно охарактеризовать как очищенную версию Java.

Синтаксические конструкции С# унаследованы не только от C++, но и от Visual Basic. Например, в С#, как и в Visual Basic, используются свойства классов как C++, С# позволяет производить перегрузку операторов для созданных вами типов (Java не поддерживает ни ту, ни другую возможность). С# — это фактически гибрид разных языков. При этом С# синтаксически не менее (если не более) чист, чем Java, так же прост, как Visual Basic, и обладает практически той же мощью и гибкостью, что и C++.

Microsoft Visio — векторный графический редактор, редактор диаграмм и блок-схем для Windows. Полнофункциональная версия Microsoft Visio Professional для создания и редактирования монограмм и диаграмм в пакеты MS Office не входит и распространяется отдельно.

Microsoft Word — текстовый процессор. Доступен под Windows, Android и macOS. Позволяет подготавливать документы различной сложности. Поддерживает OLE, подключаемые модули сторонних разработчиков, шаблоны и многое другое. Основным форматом в последней версии является позиционируемый как открытый Microsoft Office Open XML, который представляет собой ZIP-архив, содержащий текст в виде XML, а также всю необходимую графику. Наиболее распространенным остается двоичный формат файлов Microsoft Word 97—2003 с расширением DOC. Продукт занимает ведущее положение на рынке текстовых процессоров, и его форматы используются как стандарт де-факто в документообороте большинства предприятий.

Dr. Explain – программа для быстрого создания файлов справки (help-файлов), справочных систем, online руководств пользователя, пособий и технической документации к программному обеспечению и техническим системам. Уникальность Dr.Explain заключается в принципиально новом подходе к созданию пользовательской документации, который значительно ускоряет этот трудоемкий процесс по сравнению с другими инструментами.

**Программа способна анализировать пользовательский интерфейс приложений и создавать скриншоты (копии экранов) окон, автоматически расставляя на них пояснительные выноски для элементов интерфейса.**

Процесс практически полностью автоматизирован, что позволяет достаточно быстро аннотировать экраны приложений и веб-сайтов для иллюстрирования пользовательской документации на ПО.

# 4 Проектирование задачи

## 4.1 Требование к приложению

Данное приложение не требует никаких специальных средств защиты, либо ограничений прав доступа к данным.

Функциональные требования представлены на диаграмме SADT, которая отображает вначале всю систему в виде простейшей компоненты – одного блока и дуг, изображающих интерфейсы с функциями вне системы. Единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг – они также представляют полный набор внешних интерфейсов системы в целом. Диаграмма нулевого уровня представлена на рисунке 4.1.

Рисунок 4.1 – Функциональная SADT диаграмма нулевого уровня

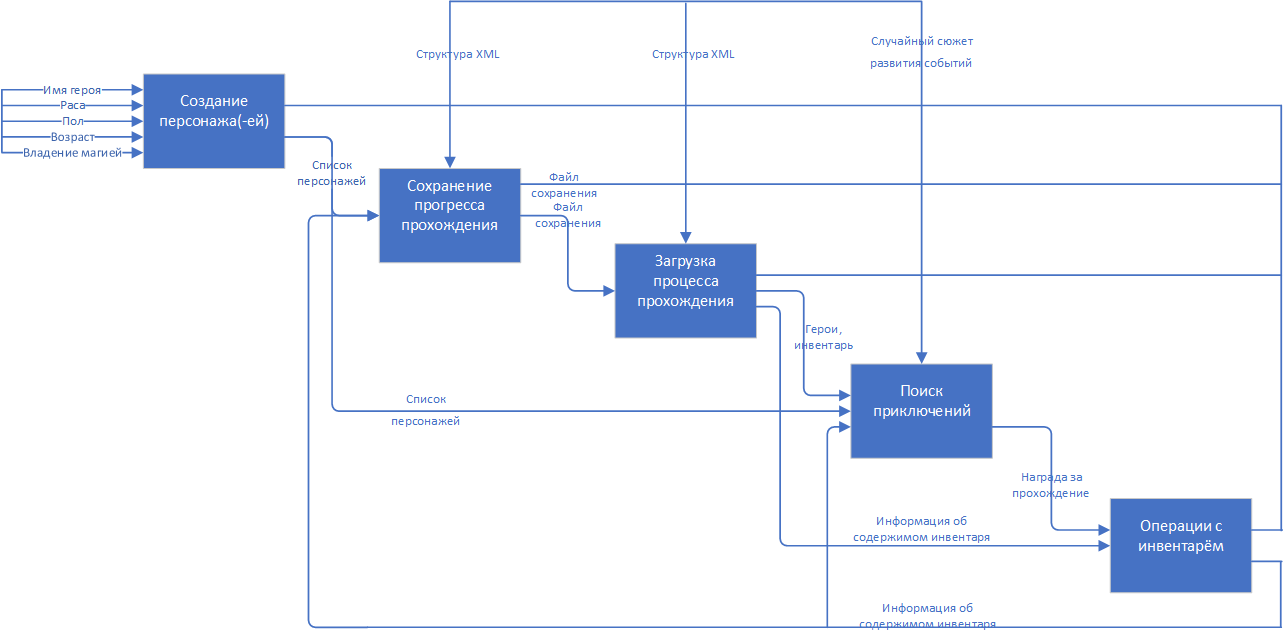
Далее блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления. Диаграмма первого уровня детализации представлена на рисунке 4.2.

Рисунок 4.2 – Функциональная SADT диаграмма первого уровня детализации

## 4.2 Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – систему меню и форм.

Все действия можно будет выполнить с помощью главной формы, которая будет появляться после запуска программы. Главная форма будет иметь кнопочное меню программы позволяющее вызывать остальные формы программы, осуществлять различные манипуляции с данными, а также получать справочную информацию о работе с приложением (по нажатию на «F1» из любого места программы). Используя меню, пользователь имеет возможность завершить работу всего приложения с сохранением сделанных изменений (при нажатии на соответствующую кнопку на главной форме).

Главное меню программы будет представлено формой Main на которой находятся основные элементы управления программой. С помощью кнопок на главной форме можно получить доступ к другим формам: создания персонажа(-ей), просмотра инвентаря (текущих предметов в наличии у героев), оправления на приключения, боя, справке и возможность выйти из приложения по завершению приключения. Структура меню представлена в таблице 3.

Таблица 3 – Структура меню

|  |  |
| --- | --- |
| Главная форма |  |
|  | Консоль – Главный инструмент повествования |
|  | Создать персонажа |
|  | Загрузить прогресс прохождения |
|  | Сохранить прогресс прохождения |
|  | Взаимодействовать с инвентарем |
|  | Узнать состояние текущего героя |
|  | Смена текущего героя |
| Форма создания персонажа |  |
| 2.1 | Имя |
| 2.2 | Пол |
| 2.3 | Возраст |
| 2.4 | Раса |
| 2.5 | Обладает магией |
| 2.6 | Создать (Закрывает окно, возвращает на главную форму) |
| Справка |  |
| 3.1 | О программе |
| 3.2 | Справка |

Для создания нового героя будет разработана специальная форма, представленная на рисунке 4.2.

Имя

Поле ввода имени

Создать

* Герой
* Героиня

Пол

Подсказки для пользователя (при наведении на пункт формы)

* Человек
* Гном
* Эльф(-ийка)
* Орк
* Гоблин

Раса

Возраст

Поле для возраста

* Да
* Нет

Обладает магией

Рисунок 4.2 – Проект формы для добавления нового героя

## 4.3 Организация данных

Организация данных будет представлена файлом с расширением .svfl и представляет собой файл сохранения прогресса игрока в формате XML. В нем будет храниться информация о герое(-ях) игрока, его инвентаре (текущих предметов, имеющихся у игрока), текст повествования, а также информация о текущем прохождении, а именно: какие сюжетные пути уже были изучены в процессе повествования, номер следующего приключения.

## 4.4 Функции и элементы управления

За кнопкой выбора 1 и, по совместительству, кнопкой атаки, которая располагается на главной форме в «окне приключений», выполняется следующий код:

private void Button\_Choice\_1\_Click(object sender, RoutedEventArgs e)

{

if (Button\_Choice\_1.Content.ToString() == "[Закрыть \nигру]")

{

Application.Current.Shutdown();

}

else if (!IsBattleMode)

{

adventureScripts.Choices\_Add(1);

ConsoleWriteLine(1);

adventureScripts.Воспроизведение\_Шагов();

}

else

{

int OrderOf = adventureScripts.OrderOf;

AttackDialog = new Attack(СписокТекущихВрагов, Персонажи[OrderOf]);

AttackDialog.ShowDialog();

if (AttackDialog.DialogResult == true)

{

СписокТекущихВрагов = AttackDialog.GetСписокТекущихВрагов();

if (AttackDialog.IsEnemyKilled == false)

{

if (Персонажи[OrderOf].Пол == true)

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} нанёс {AttackDialog.ЗначениеАтаки} единиц урона здоровью цели: " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага}. Текущие значение здоровья врага стало " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ЗдоровьеВрага} единиц здоровья.");

}

else

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} нанесла {AttackDialog.ЗначениеАтаки} единиц урона здоровью цели: " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага}. Текущие значение здоровья врага стало " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ЗдоровьеВрага} единиц здоровья.");

}

}

else

{

if (Персонажи[OrderOf].Пол == true)

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} нанёс {AttackDialog.ЗначениеАтаки} единиц урона здоровью цели: " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага}. {СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага} убит.");

}

else

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} нанесла {AttackDialog.ЗначениеАтаки} единиц урона здоровью цели: " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага}. {СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага} убит.");

}

}

Random random = new Random();

Персонажи[OrderOf].Добавить\_ОчкиОпыта((uint)random.Next(100, 150));

adventureScripts.IncreaseOrderOf();

}

adventureScripts.Battle(СписокТекущихВрагов, ПорядокАтаки, СписокГероевВЗащите);

}

}

За кнопкой выбора 2, и по совместительству, кнопкой использования заклинания, которая располагается на главной форме в «окне приключений», выполняется следующий код:

private void Button\_Choice\_2\_Click(object sender, RoutedEventArgs e)

{

if (!IsBattleMode)

{

adventureScripts.Choices\_Add(2);

ConsoleWriteLine(2);

adventureScripts.Воспроизведение\_Шагов();

}

else

{

int OrderOf = adventureScripts.OrderOf;

Magic magic = new Magic(Персонажи, СписокТекущихВрагов, ПорядокАтаки, OrderOf, МножительСилыРун);

magic.ShowDialog();

if (magic.DialogResult == true)

{

ConsoleWriteLine(magic.ConsoleOutput);

adventureScripts.IncreaseOrderOf();

adventureScripts.Battle(СписокТекущихВрагов, ПорядокАтаки, СписокГероевВЗащите);

}

}

}

За кнопкой выбора 3, и по совместительству, кнопкой защиты, которая располагается на главной форме в «окне приключений» выполняется следующий код:

private void Button\_Choice\_3\_Click(object sender, RoutedEventArgs e)

{

if (!IsBattleMode)

{

adventureScripts.Choices\_Add(3);

ConsoleWriteLine(3);

adventureScripts.Воспроизведение\_Шагов();

}

else

{

int OrderOf = adventureScripts.OrderOf;

СписокГероевВЗащите[OrderOf] = true;

if (Персонажи[OrderOf].Пол == true)

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} принял защитную стойку. Входящий урон снижен в 2 раза.");

}

else

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} приняла защитную стойку. Входящий урон снижен в 2 раза.");

}

adventureScripts.IncreaseOrderOf();

adventureScripts.Battle(СписокТекущихВрагов, ПорядокАтаки, СписокГероевВЗащите);

}

}

При сохранении игры выполняется следующий код:

private void Button\_Save\_Click(object sender, RoutedEventArgs e)

{

//Сохранение информации о персонажах

try

{

DataSet ds = new DataSet();

DataTable dt = new DataTable

{

TableName = "Персонажи"

};

dt.Columns.Add("ID");

dt.Columns.Add("Имя");

dt.Columns.Add("Раса");

dt.Columns.Add("Возраст");

dt.Columns.Add("Пол");

dt.Columns.Add("Обладает\_магией");

dt.Columns.Add("Ослаблен");

dt.Columns.Add("Болен");

dt.Columns.Add("Отравлен");

dt.Columns.Add("Парализован");

dt.Columns.Add("Мёртв");

dt.Columns.Add("Может\_говорить");

dt.Columns.Add("Может\_двигаться");

dt.Columns.Add("Максимальное\_здоровье");

dt.Columns.Add("Очки\_здоровья");

dt.Columns.Add("Очки\_опыта");

dt.Columns.Add("Максимальная\_мана");

dt.Columns.Add("Очки\_маны");

ds.Tables.Add(dt);

for (int i = 0; i < Персонажи.Count; i++)

{

if (Обладает\_магией[i] == false)

{

Персонаж персонаж = Персонажи[i] as Персонаж;

DataRow строка = ds.Tables["Персонажи"].NewRow();

строка["ID"] = персонаж.ID;

строка["Имя"] = персонаж.Имя;

строка["Раса"] = персонаж.Раса;

строка["Возраст"] = персонаж.Возраст;

строка["Пол"] = персонаж.Пол;

строка["Обладает\_магией"] = "False";

строка["Ослаблен"] = персонаж.Состояние[0];

строка["Болен"] = персонаж.Состояние[1];

строка["Отравлен"] = персонаж.Состояние[2];

строка["Парализован"] = персонаж.Состояние[3];

строка["Мёртв"] = персонаж.Состояние[4];

строка["Может\_говорить"] = персонаж.Может\_говорить;

строка["Может\_двигаться"] = персонаж.Может\_двигаться;

строка["Максимальное\_здоровье"] = персонаж.Максимальное\_здоровье;

строка["Очки\_здоровья"] = персонаж.Очки\_Здоровья;

строка["Очки\_опыта"] = персонаж.Очки\_Опыта;

ds.Tables["Персонажи"].Rows.Add(строка);

}

else

{

Персонаж\_с\_магией персонаж = Персонажи[i] as Персонаж\_с\_магией;

DataRow строка = ds.Tables["Персонажи"].NewRow();

строка["ID"] = персонаж.ID;

строка["Имя"] = персонаж.Имя;

строка["Раса"] = персонаж.Раса;

строка["Возраст"] = персонаж.Возраст;

строка["Пол"] = персонаж.Пол;

строка["Обладает\_магией"] = "True";

строка["Ослаблен"] = персонаж.Состояние[0];

строка["Болен"] = персонаж.Состояние[1];

строка["Отравлен"] = персонаж.Состояние[2];

строка["Парализован"] = персонаж.Состояние[3];

строка["Мёртв"] = персонаж.Состояние[4];

строка["Может\_говорить"] = персонаж.Может\_говорить;

строка["Может\_двигаться"] = персонаж.Может\_двигаться;

строка["Максимальное\_здоровье"] = персонаж.Максимальное\_здоровье;

строка["Очки\_здоровья"] = персонаж.Очки\_Здоровья;

строка["Очки\_опыта"] = персонаж.Очки\_Опыта;

строка["Максимальная\_мана"] = персонаж.Максимальная\_мана;

строка["Очки\_маны"] = персонаж.Очки\_Маны;

ds.Tables["Персонажи"].Rows.Add(строка);

}

}

//сохранение информации о тексте в консоли

dt = new DataTable

{

TableName = "Консоль"

};

dt.Columns.Add("Консоль\_Текст");

ds.Tables.Add(dt);

DataRow СтрокаКонсоли = ds.Tables["Консоль"].NewRow();

СтрокаКонсоли["Консоль\_Текст"] = Console.Text;

ds.Tables["Консоль"].Rows.Add(СтрокаКонсоли);

//сохранение информации о предметах в инвентаре

dt = new DataTable

{

TableName = "Инвентарь"

};

dt.Columns.Add("Название");

dt.Columns.Add("Числовое\_значение"); //кол-во предметов либо значение восстановления (HP/MP)

ds.Tables.Add(dt);

foreach (Предметы.Зелье Предмет in ПереченьПредметов)

{

DataRow Строка = ds.Tables["Инвентарь"].NewRow();

Строка["Название"] = Предмет.Название;

if (Предмет is Предметы.Зелье\_Лечения)

{

if (Предмет is Предметы.Малое\_Зелье\_Лечения)

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Малое\_Зелье\_Лечения).Количество\_Восполняемого\_Здоровья;

}

else if (Предмет is Предметы.Среднее\_Зелье\_Лечения)

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Среднее\_Зелье\_Лечения).Количество\_Восполняемого\_Здоровья;

}

else

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Большое\_Зелье\_Лечения).Количество\_Восполняемого\_Здоровья;

}

}

else

{

if (Предмет is Предметы.Бутылек\_Маны)

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Бутылек\_Маны).Количество\_Восполняемой\_Маны;

}

else if (Предмет is Предметы.Фласка\_маны)

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Фласка\_маны).Количество\_Восполняемой\_Маны;

}

else

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Банка\_маны).Количество\_Восполняемой\_Маны;

}

}

ds.Tables["Инвентарь"].Rows.Add(Строка);

}

ds.WriteXml("SaveFile.svfl");

MessageBox.Show($"Игра сохранена", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Asterisk);

}

catch (Exception ex)

{

StreamWriter streamWriter = new StreamWriter("ErrorLog.log");

streamWriter.Write(ex);

MessageBox.Show($"Произошла ошибка при сохранении. В директории с игрой был создан лог файл для отладки", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

streamWriter.Close();

}

}

При загрузке игры выполняется следующий код:

private void Button\_Load\_Click(object sender, RoutedEventArgs e)

{

if (Персонажи.Count == 0)

{

DoLoadFile();

}

else

{

MessageBoxResult DialogResult = MessageBox.Show("Вы уверены что хотите загрузить игру? Текущая игра будет безвозвратно утеряна", "Подтвердить действие?", MessageBoxButton.YesNo, MessageBoxImage.Warning);

if (DialogResult == MessageBoxResult.Yes)

{

DoLoadFile();

}

}

if (File.Exists("SaveFile.svfl"))

{

Button\_Inventory.IsEnabled = true;

MainWindow\_Button\_SwitchHero.IsEnabled = true;

MainWindow\_Button\_GoForAdventure.IsEnabled = true;

}

}

private void DoLoadFile()

{

if (File.Exists("SaveFile.svfl"))

{

Персонажи.Clear();

ПереченьПредметов.Clear();

//загрузка персонажей

DataSet ds = new DataSet();

ds.ReadXml("SaveFile.svfl");

foreach (DataRow строка in ds.Tables["Персонажи"].Rows)

{

if (строка[5] as string == "True")

{

Персонажи.Add(new Персонаж\_с\_магией(

Convert.ToUInt32(строка["ID"]),

Convert.ToString(строка["Имя"]),

Convert.ToBoolean(строка["Пол"]),

Convert.ToUInt32(строка["Возраст"]),

Convert.ToString(строка["Раса"]),

Convert.ToBoolean(строка["Ослаблен"]),

Convert.ToBoolean(строка["Болен"]),

Convert.ToBoolean(строка["Отравлен"]),

Convert.ToBoolean(строка["Парализован"]),

Convert.ToBoolean(строка["Мёртв"]),

Convert.ToBoolean(строка["Может\_говорить"]),

Convert.ToBoolean(строка["Может\_двигаться"]),

Convert.ToUInt32(строка["Максимальное\_здоровье"]),

Convert.ToUInt32(строка["Очки\_Здоровья"]),

Convert.ToUInt32(строка["Очки\_Опыта"]),

Convert.ToUInt32(строка["Максимальная\_мана"]),

Convert.ToUInt32(строка["Очки\_Маны"]),

МножительСилыРун

));

}

else

{

Персонажи.Add(new Персонаж(

Convert.ToUInt32(строка["ID"]),

Convert.ToString(строка["Имя"]),

Convert.ToBoolean(строка["Пол"]),

Convert.ToUInt32(строка["Возраст"]),

Convert.ToString(строка["Раса"]),

Convert.ToBoolean(строка["Ослаблен"]),

Convert.ToBoolean(строка["Болен"]),

Convert.ToBoolean(строка["Отравлен"]),

Convert.ToBoolean(строка["Парализован"]),

Convert.ToBoolean(строка["Мёртв"]),

Convert.ToBoolean(строка["Может\_говорить"]),

Convert.ToBoolean(строка["Может\_двигаться"]),

Convert.ToUInt32(строка["Максимальное\_здоровье"]),

Convert.ToUInt32(строка["Очки\_Здоровья"]),

Convert.ToUInt32(строка["Очки\_Опыта"]),

МножительСилыРун

));

}

}

//загрузка консоли

DataRow СтрокиКонсоли = ds.Tables["Консоль"].Rows[0];

Console.Text = Convert.ToString(СтрокиКонсоли["Консоль\_Текст"]);

ТекущийПерсонаж = Персонажи.Count - 1;

foreach (Персонаж Перс in Персонажи)

{

if (Перс is Персонаж\_с\_магией)

{

Обладает\_магией.Add(true);

}

else

{

Обладает\_магией.Add(false);

}

}

//загрузка предметов

if (ds.Tables["Инвентарь"] != null)

{

foreach (DataRow строка in ds.Tables["Инвентарь"].Rows)

{

if (строка[0] as string == "[+25 ОЗ] Малое зелье лечения")

{

ПереченьПредметов.Add(new Предметы.Малое\_Зелье\_Лечения());

}

else if (строка[0] as string == "[+50 ОЗ] Среднее зелье лечения")

{

ПереченьПредметов.Add(new Предметы.Среднее\_Зелье\_Лечения());

}

else if (строка[0] as string == "[+75 ОЗ] Большое зелье лечения")

{

ПереченьПредметов.Add(new Предметы.Большое\_Зелье\_Лечения());

}

else if (строка[0] as string == "[+25 ОМ] Бутылек Маны")

{

ПереченьПредметов.Add(new Предметы.Бутылек\_Маны());

}

else if (строка[0] as string == "[+50 ОМ] Фласка маны")

{

ПереченьПредметов.Add(new Предметы.Фласка\_маны());

}

else if (строка[0] as string == "[+75 ОМ] Банка маны")

{

ПереченьПредметов.Add(new Предметы.Банка\_маны());

}

}

}

InventoryWindow = new Inventory(Персонажи, ПереченьПредметов);

if (Персонажи.Count == 3)

{

MainWindow\_Button\_AddHero.IsEnabled = false;

}

MainWindow\_Button\_ShowInfo.IsEnabled = true;

Button\_Save.IsEnabled = true;

}

else

{

MessageBox.Show("Файл сохранения не найден. Убедитесь что файл сохранения располагается в директории с игрой", "Ошибка.", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

## 4.5 Проектирование справочной системы приложения

В данной программе присутствует справочная система для ознакомления пользователя с программой и помощи в навигации между разделами меню. Для доступа к справке из любого места программы используйте клавишу «F1».

Справочная система необходима для ознакомления с программой.

Система справки данного программного средства будет содержать следующие разделы:

* справка о главном меню;
* справка о меню создания персонажа;
* справка о меню инвентаря персонажей;
* справка о сохранении/загрузке текущего прохождения
* о программе.

Справочная система будет создана в программе Dr.Explain.

# 5 Описание программного средства

## 5.1 Функциональное назначение

Для хранения и использования программы потребуется от 100 кб до 1 мб свободного места на диске. Во время тестов, объем памяти занимаемой программой составил 109 кб.

Процесс инсталляции не требуется. Программа запускается As-Is («Как есть») из .exe файла.

Ролевая игра «Герои подземелья» предназначено для досуга.

Характерными особенностями разработанной программы являются:

* + Практическое использование Windows Presentation Foundation (WPF) с использованием языка программирования C#.
  + Практическое применение экстремального варианта написания приложения в наикратчайшие сроки.

Для применения данного программного средства необходимы следующие технически требования:

* процессор 800 МГц и выше;
* оперативная память 128 Мбайт и более;
* свободное место на диске 100 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более;
* монитор;
* мышь, клавиатура;

Программное средство создано в среде разработки Visual Studio 2019 на языке программирования С# и XAML в операционной системе Windows 10. Программное средство может работать в средах операционных систем семейства Microsoft Windows начиная с Windows 7. Программа не требовательна к системным ресурсам, также проста в использовании и не требует специальных навыков при работе. Для работы данного программного средства требуется предварительная установка и настройка следующих программных продуктов:

* Microsoft Net Framework 4.6.1

Инсталляция программного средства не требуется, достаточно только скопировать готовый проект на ПК и запустить.

## 5.2 Функциональное назначение

Основной задачей приложения является досуг.

Данное программное средство предназначено для занимательного времяпрепровождения, занимающейся вопросами отсутствия задач для выполнения в настоящем времени, либо наличием свободного времени. Целью программного средства является заполнение свободного времени.

Программа использует стандартные элементы управления, такие как кнопки, меню, списка, поля ввода, что обеспечивает единство интерфейса системы и программного средства.

В работе программного средства предусмотрены некоторые ситуации, которые должны предупреждать пользователя, чтобы он выполнял все необходимые требования по эксплуатации программы. Для этого существуют сообщения системы, например если пользователь попытался загрузить сохраненное прохождение, но файл не найден либо поврежден выведется сообщение об ошибке «Файл не найден, удостоверьтесь что он находится в директории с игрой».

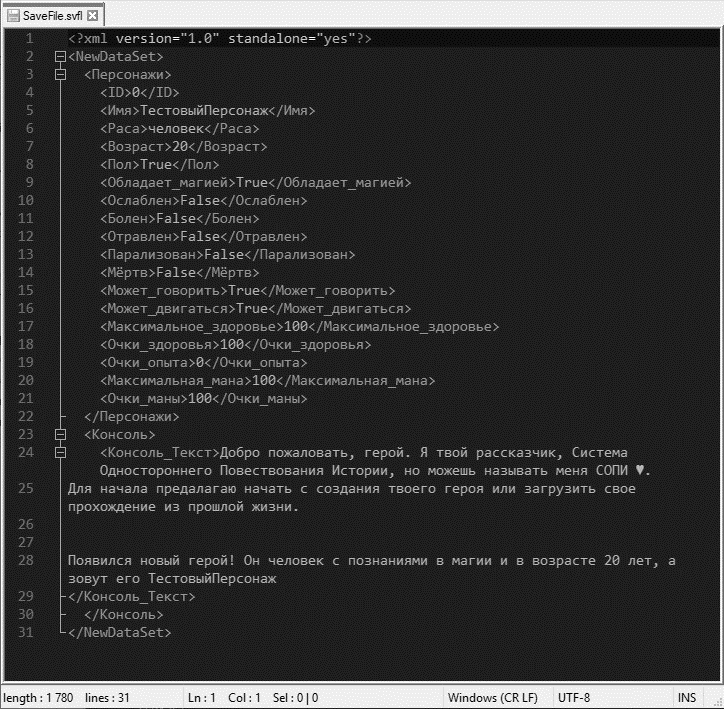
Таким образом, программа может применяться в реальных условиях, представляя собой приятный способ досуга.

## 5.3 Входные данные

Входными данными для приложения являются либо текстовые данные, вводимые с клавиатуры (такие как имя героя, возраст), либо файл сохранения (SaveFile.svfl).

## 5.4 Выходные данные

Выходными данными разработанного приложения является SaveFile.svfl, имеющий характерную внутреннюю структуру с файлами XML. Внутренняя структура типичного начального файла сохранения отображена в приложении Б на рисунке Б1.



# 6 Методика испытаний

6.1 Технические требования

Минимальными требованиями для оптимальной работы программного средства является персональный компьютер (ПК) со следующими характеристиками:

* процессор 800 МГц и выше;
* оперативная память 128 Мбайт и более;
* свободное место на диске 5 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более.

Компьютер должен работать под управлением операционной системы, начиная с Windows7 и выше. Наиболее удобной операционной системой для проведения испытаний является Windows 7, так как она ориентированна на максимальное использование всех возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

## 6.2 Порядок проведения испытаний

## 6.2.1 Функциональное тестирование

функциональное (проверка каждого пункта меню, каждой операции, каждой функции, которые выполняет приложение с представлением скриншотов результатов работы с соответствующим действием)

## 6.2.2 Полное тестирование

полное тестирование (описание тестового примера с конкретными входными данными и результатами, которые должны быть получены).

В полном тестировании необходимо проводить ссылки на результаты работы ПС, которые описаны в функциональном тестировании. Дублирование результатов работы ПС не допускается.

# 7 Применение

## 7.1 Назначение программы

описать сведения о назначении программного средства, области применения, классе решаемых задач, ограничениях, накладываемых на область применения.

## 7.2 Условия применения

Для применения данного программного средства необходимы следующие технически требования:

* процессор Intel Core 2 Duo или выше;
* минимальный объем оперативной памяти — 512 Мбайт;
* операционная система Windows 7 и выше;
* Framework v4.6.1
* рекомендуется монитор типа VGA или с лучшей разрешающей способностью;
* клавиатура;
* мышь.

## 7.3 Справочная система

Справочная система программного средства представляет собой отдельный файл «HelpFile.chm» с полным описанием основных функций программы в формате \*.chm. В справочной системе даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Справка имеет следующие разделы, которые отображены на рисунке 7.1:

* начальная страница;
* главное окно;
* окно приключений;
* экран битвы;
* создание персонажа;
* заклинания;
* инвентарь;
* руны.



Рисунок 7.1 – Внутренняя структура разделов справки

# Заключение

необходимо отразить:

* как выполнена поставленная задача;
* степень соответствия проектных решений поставленной задаче;
* причины несоответствия (если таковы имеются).

Программное средство можно модифицировать, как того пожелает заказчик.

В процессе разработки программы использовался в большом объеме материал по программированию и алгоритмизации, что способствовало закреплению наработанных навыков и умений в этих областях знаний.

При разработке приложения наибольшее внимание уделялось максимальному созданию лаконичного интерфейса.

Проект был разработан в среде Microsoft Visual Studio 2012 на языке С#.

Список используемых источников

1. Грабер, М. SQL. Описание SQL92, SQL99 и SQLJ/ Грабер М.: Лори, 2003 г.
2. Дейт, К. Дж. Введение в системы баз данных/ Дейт К.Дж.: Вильямс, 2006 г.
3. Мартин, К.Р. Чистый Код. Создание анализ и рефакторинг. Библиотека программистов./ К.Р. Мартин.: Питер, 2010 г.
4. Орлов, С.А. Технология разработки программного обеспечения. Учебник для вузов. 4-е изд./ С.А. Орлов, Б.Я. Цилькер – СПБ.: Питер, 2012 г.
5. Ржеутская, Н.В. Методические указания по оформлению курсовых проектов/ Н.В. Ржеутская – Минск: КБП, 2019.
6. Рудаков, А.В. Технология разработки программных продуктов. Практикум: учебн. пособие для студ. учреждений сред. проф. образования/А.В.Рудаков, Г.Н. Федорова М. :Издательский центр «Академия»; 2014 г.
7. Текст Программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск: Межгос. Совет по стандартизации, методологии и сертификации, 2000.
8. metanit [Электронный ресурс]/Сайт о программировании, - metanit.com, 2012-2019. – Режим доступа: <http://www.metanit.com>. – Дата доступа: 25.03.2019.
9. Microsoft [Электронный ресурс]/ Сайт, в котором хранятся инструменты разработки, - microsoft.com, 2019. – Режим доступа: <http://www.microsoft.com> – Дата доступа: 25.03.2019.
10. msdn [Электронный ресурс]/ Сайт о программировании, - msdn.com, 2019. – Режим доступа: <http://www.msdn.microsoft.com>. – Дата доступа: 25.03.2019.

# Приложение А

Текст программы

AddHero.xaml:

<Window x:Class="Ролевая\_игра\_\_WPF\_.AddHero" Name="NewHeroWindow" Title="Новый герой" Height="300" Width="600" ResizeMode="NoResize" WindowStartupLocation="CenterScreen" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<Grid Margin="15">

<Label Content="Имя" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="37" />

<TextBox Name="TextBox\_NameOfHero" HorizontalAlignment="Left" Height="23" Margin="47,10,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="426" MouseEnter="TextBox\_MouseEnter" MouseLeave="TextBlock\_Clear" TextChanged="TextBox\_NameOfHero\_TextChanged" />

<GroupBox Header="Пол" HorizontalAlignment="Left" Height="78" Margin="10,41,0,0" VerticalAlignment="Top" Width="100">

<StackPanel>

<RadioButton Name="RadioButton\_Hero\_Male" Content="Герой" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" MouseEnter="RadioButton\_Hero\_Male\_MouseEnter" Click="RadioButton\_Hero\_Male\_Click" />

<RadioButton Name="RadioButton\_Hero\_Female" Content="Героиня" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" MouseEnter="RadioButton\_Hero\_Female\_MouseEnter" Click="RadioButton\_Hero\_Female\_Click" />

</StackPanel>

</GroupBox>

<GroupBox Name="GroupBox\_Race" Header="Раса" HorizontalAlignment="Left" Height="156" Margin="126,41,0,0" VerticalAlignment="Top" Width="107">

<StackPanel Margin="0,0,0.4,0.2">

<RadioButton Name="RadioButton\_Race\_Human" Content="Человек" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Checked="RadioButton\_Race\_Human\_Checked" MouseEnter="RadioButton\_Race\_Human\_MouseEnter" />

<RadioButton Name="RadioButton\_Race\_Dwarf" Content="Гном" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Checked="RadioButton\_Race\_Dwarf\_Checked" MouseEnter="RadioButton\_Race\_Dwarf\_MouseEnter" />

<RadioButton Name="RadioButton\_Race\_Elf" Content="Эльф" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Checked="RadioButton\_Race\_Elf\_Checked" MouseEnter="RadioButton\_Race\_Elf\_MouseEnter" />

<RadioButton Name="RadioButton\_Race\_Ork" Content="Орк" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Checked="RadioButton\_Race\_Ork\_Checked" MouseEnter="RadioButton\_Race\_Ork\_MouseEnter" />

<RadioButton Name="RadioButton\_Race\_Goblin" Content="Гоблин" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Checked="RadioButton\_Race\_Goblin\_Checked" MouseEnter="RadioButton\_Race\_Goblin\_MouseEnter" />

</StackPanel>

</GroupBox>

<TextBlock Name="TextBlock\_Description" HorizontalAlignment="Left" TextWrapping="Wrap" VerticalAlignment="Top" Width="543" Height="38" Margin="10,202,0,0" />

<Button Name="Button\_Accept" Content="Создать" HorizontalAlignment="Left" Margin="478,10,0,0" VerticalAlignment="Top" Width="75" Height="23" IsDefault="True" IsEnabled="False" Click="Button\_Accept\_Click" MouseEnter="Button\_Accept\_MouseEnter" />

<GroupBox Header="Возраст" HorizontalAlignment="Left" Height="73" Margin="10,124,0,0" VerticalAlignment="Top" Width="100">

<TextBox Name="TextBox\_Age" HorizontalAlignment="Left" Height="23" Margin="18,15,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="54" TextChanged="TextBox\_TextChanged" />

</GroupBox>

<GroupBox Name="GroupBox\_HaveMagic" Header="Обладает магией" HorizontalAlignment="Left" Height="78" Margin="238,41,0,0" VerticalAlignment="Top" Width="128">

<StackPanel Margin="0,0,0.4,0.2" MouseEnter="StackPanel\_MouseEnter">

<RadioButton Name="Hero\_DoHaveMagic" Content="Да" Margin="10,10,0,0" HorizontalAlignment="Left" VerticalAlignment="Top" Click="Hero\_DoHaveMagic\_Click" />

<RadioButton Name="Hero\_DoNotHaveMagic" Content="Нет" Margin="10,10,0,52.6" HorizontalAlignment="Left" VerticalAlignment="Top" Click="Hero\_DoNotHaveMagic\_Click" />

</StackPanel>

</GroupBox>

</Grid>

</Window>

AddHero.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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.Shapes;

namespace Ролевая\_игра\_\_WPF\_

{

public partial class AddHero : Window

{

uint RecievedID;

bool IsAgeAcceptable = false;

bool IsNameAcceptable = false;

bool IsMagicChecked = false;

Персонаж Новый\_персонаж = null;

Персонаж\_с\_магией Новый\_персонаж\_с\_магией = null;

public string Имя { get; private set; }

public string Раса { get; private set; }

public uint Возраст { get; private set; }

public bool Пол { get; private set; }

private double[] МножительСилыРун;

public AddHero(int ID, double[] МножительСилыРун)

{

InitializeComponent();

RecievedID = (uint)ID;

GroupBox\_Race.Visibility = Visibility.Hidden;

GroupBox\_HaveMagic.Visibility = Visibility.Hidden;

this.МножительСилыРун = МножительСилыРун;

}

public Персонаж Получить\_Персонажа() => Новый\_персонаж;

public Персонаж\_с\_магией Получить\_Персонажа\_с\_магией() => Новый\_персонаж\_с\_магией;

private void Button\_Accept\_Click(object sender, RoutedEventArgs e)

{

if (Hero\_DoHaveMagic.IsChecked == true)

{

Новый\_персонаж\_с\_магией = new Персонаж\_с\_магией(RecievedID, Имя, Пол, Возраст, Раса, МножительСилыРун);

}

else

{

Новый\_персонаж = new Персонаж(RecievedID, Имя, Пол, Возраст, Раса, МножительСилыРун);

}

DialogResult = true;

Close();

}

void TextBlock\_Clear(object sender, MouseEventArgs e)

{

TextBlock\_Description.Text = "";

}

void TextBlock\_Change(string Text)

{

TextBlock\_Description.Text = Text;

}

private void RadioButton\_Hero\_Male\_Click(object sender, RoutedEventArgs e)

{

GroupBox\_Race.Visibility = Visibility.Visible;

RadioButton\_Race\_Elf.Content = "Эльф";

Пол = true;

CheckAllRequaredFields();

}

private void RadioButton\_Hero\_Female\_Click(object sender, RoutedEventArgs e)

{

GroupBox\_Race.Visibility = Visibility.Visible;

RadioButton\_Race\_Elf.Content = "Эльфийка";

Пол = false;

CheckAllRequaredFields();

}

//проверка ввода возраста

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

bool AgeHasInvalidInput = false;

if (TextBox\_Age.Text.Length == 0)

AgeHasInvalidInput = true;

foreach (char symbol in TextBox\_Age.Text)

{

if (!char.IsDigit(symbol))

{

AgeHasInvalidInput = true;

break;

}

}

if (AgeHasInvalidInput == false)

{

IsAgeAcceptable = true;

Возраст = Convert.ToUInt32(TextBox\_Age.Text);

}

else

IsAgeAcceptable = false;

CheckAllRequaredFields();

}

//проверка ввода имени

private void TextBox\_NameOfHero\_TextChanged(object sender, TextChangedEventArgs e)

{

bool NameHasInvalidInput = false;

char PreviousSymbol = 'a';

foreach (char symbol in TextBox\_NameOfHero.Text)

{

if (!char.IsLetter(symbol) || symbol == ' ' && PreviousSymbol == ' ')

{

NameHasInvalidInput = true;

break;

}

PreviousSymbol = symbol;

}

if (NameHasInvalidInput == false)

{

IsNameAcceptable = true;

Имя = TextBox\_NameOfHero.Text;

}

else

IsNameAcceptable = false;

CheckAllRequaredFields();

}

void CheckAllRequaredFields()

{

if (IsNameAcceptable && IsAgeAcceptable && IsMagicChecked)

Button\_Accept.IsEnabled = true;

else

Button\_Accept.IsEnabled = false;

}

private void RadioButton\_Race\_Human\_Checked(object sender, RoutedEventArgs e)

{

GroupBox\_HaveMagic.Visibility = Visibility.Visible;

Раса = "человек";

CheckAllRequaredFields();

}

private void RadioButton\_Race\_Dwarf\_Checked(object sender, RoutedEventArgs e)

{

GroupBox\_HaveMagic.Visibility = Visibility.Visible;

Раса = "гном";

CheckAllRequaredFields();

}

private void RadioButton\_Race\_Elf\_Checked(object sender, RoutedEventArgs e)

{

GroupBox\_HaveMagic.Visibility = Visibility.Visible;

if (Пол == true)

Раса = "эльф";

else

Раса = "эльфийка";

CheckAllRequaredFields();

}

private void RadioButton\_Race\_Ork\_Checked(object sender, RoutedEventArgs e)

{

GroupBox\_HaveMagic.Visibility = Visibility.Visible;

Раса = "орк";

CheckAllRequaredFields();

}

private void RadioButton\_Race\_Goblin\_Checked(object sender, RoutedEventArgs e)

{

GroupBox\_HaveMagic.Visibility = Visibility.Visible;

Раса = "гоблин";

CheckAllRequaredFields();

}

private void Hero\_DoHaveMagic\_Click(object sender, RoutedEventArgs e)

{

IsMagicChecked = true;

CheckAllRequaredFields();

}

private void Hero\_DoNotHaveMagic\_Click(object sender, RoutedEventArgs e)

{

IsMagicChecked = true;

CheckAllRequaredFields();

}

private void TextBox\_MouseEnter(object sender, MouseEventArgs e)

{

TextBlock\_Change("К истории о противостоянии присоединяется новый герой! \nО ком будут слагать эпические легенды?");

}

private void RadioButton\_Hero\_Male\_MouseEnter(object sender, MouseEventArgs e)

{

TextBlock\_Change($"Благородный сэр с гордым именем {TextBox\_NameOfHero.Text} и в сияющих доспехах");

}

private void RadioButton\_Hero\_Female\_MouseEnter(object sender, MouseEventArgs e)

{

TextBlock\_Change($"Благородная мэм с гордым именем {TextBox\_NameOfHero.Text} и в сияющих доспехах");

}

private void Button\_Accept\_MouseEnter(object sender, MouseEventArgs e)

{

TextBlock\_Change($"Готовы отправится на поиски приключений?");

}

private void RadioButton\_Race\_Human\_MouseEnter(object sender, MouseEventArgs e)

{

if (Пол)

{

TextBlock\_Change($"Храбрый герой, с благородным именем {Имя}, оказался человеком! \n" +

$"Самым средним и не выдающимся представителем мифического мира");

}

else

{

TextBlock\_Change($"Храбрая героиня, с благородным именем {Имя}, оказалась человеком! \n" +

$"Самым средним и не выдающимся представителем мифического мира");

}

}

private void RadioButton\_Race\_Dwarf\_MouseEnter(object sender, MouseEventArgs e)

{

if (Пол)

{

TextBlock\_Change($"Не позволяйте себе быть обманутым его ростом, он много кому сможет задать жару");

}

else

{

TextBlock\_Change($"Не позволяйте себе быть обманутым её ростом, она много кому сможет задать жару");

}

}

private void RadioButton\_Race\_Elf\_MouseEnter(object sender, MouseEventArgs e)

{

if (Пол)

{

TextBlock\_Change($"Высокие (и чаще высокомерные) мастера театра боевых действий с острыми ушками и не менее острым языком");

}

else

{

TextBlock\_Change($"Высокие (и чаще высокомерные) мастерицы театра боевых действий с острыми ушками и не менее острым языком");

}

}

private void RadioButton\_Race\_Ork\_MouseEnter(object sender, MouseEventArgs e)

{

if (Пол)

{

TextBlock\_Change($"Орки, могучие и свирепые берсерки с жаждой крови, но обделенные интеллектом");

}

else

{

TextBlock\_Change($"Орки, могучие и свирепые берсерки с жаждой крови, но обделенные интеллектом");

}

}

private void RadioButton\_Race\_Goblin\_MouseEnter(object sender, MouseEventArgs e)

{

if (Пол)

{

TextBlock\_Change($"Гоблины славятся своими умениями как инженеры (хоть их изобретения так и норовят развалиться " +

$"или взорваться), так и как торговцы, за счет своей хитрости");

}

else

{

TextBlock\_Change($"Гоблины славятся своими умениями как инженеры (хоть их изобретения так и норовят развалиться " +

$"или взорваться), так и как торговцы, за счет своей хитрости");

}

}

private void StackPanel\_MouseEnter(object sender, MouseEventArgs e)

{

TextBlock\_Change($"Будет ли {Имя} обладать магией?");

}

}

}

AdventureScripts.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace Ролевая\_игра\_\_WPF\_

{

public class AdventureScripts

{

private List<int> Choices = new List<int>();

private MainWindow MainWindow;

public int OrderOf { get; private set; } = 0;

public AdventureScripts(int CaseOfAdventure, MainWindow MainForm)

{

MainWindow = MainForm;

if (Choices.Count == 0)

{

Начало\_приключения();

}

else

{

Воспроизведение\_Шагов();

}

}

public void Choices\_Add(int value)

{

Choices.Add(value);

}

public int Choices\_GetLast => Choices.Last();

private void Начало\_приключения()

{

if (MainWindow.Персонажи.Count == 0 && MainWindow.Персонажи[0].Пол == true)

{

MainWindow.ConsoleWriteLine($"СОПИ: Герой-одиночка был призван в этот мир могущественными волшебниками королевства Мелромарк, " +

$"дабы спасти всех в королевстве от предсказанного уничтожения нашествиями волн монстров из другого измерения. " +

$"Вы были призваны в тёмной комнате отдающей сыростью, в окружении магов, внемлющих вашим первым словам. Что стоит им сказать?");

}

else if (MainWindow.Персонажи.Count == 0 && MainWindow.Персонажи[0].Пол == false)

{

MainWindow.ConsoleWriteLine($"СОПИ: Героиня-одиночка была призвана в этот мир могущественными волшебниками королества Мелромарк, " +

$"дабы спасти всех в королевстве от предсказанного уничтожения нашествиями волн монстров из другого измерения. " +

$"Вы были призваны в тёмной комнате отдающей сыростью, в окружении магов, внемлющих вашим первым словам. Что стоит им сказать?");

}

else if (MainWindow.Персонажи.Count == 1)

{

MainWindow.ConsoleWriteLine($"СОПИ: Двое героев были призваны в этот мир могущественными волшебниками королества Мелромарк, " +

$"дабы спасти всех в королевстве от предсказанного уничтожения нашествиями волн мостров из другого измерения. " +

$"Вы вдвоём были призваны в тёмной комнате отдающей сыростью, в окружении магов, внемлющих вашим первым словам. Что должны сказать герои?");

}

else

{

MainWindow.ConsoleWriteLine($"СОПИ: Трое героев были призваны в этот мир могущественными волшебниками королества Мелромарк, " +

$"дабы спасти всех в королевстве от предсказанного уничтожения нашествиями волн мостров из другого измерения. " +

$"Вы втроём были призваны в тёмной комнате отдающей сыростью, в окружении магов, внемлющих вашим первым словам. Что должны сказать герои?");

}

MainWindow.Change\_Button\_Choice\_1("Привет!");

MainWindow.Change\_Button\_Choice\_2("Как дела?");

MainWindow.Change\_Button\_Choice\_3("Кто вы?");

}

public void Воспроизведение\_Шагов()

{

if (Choices.Count == 1 && Choices[0] == 1)

{

if (MainWindow.Персонажи.Count == 0 && MainWindow.Персонажи[0].Пол == true)

{

MainWindow.ConsoleWriteLine($"- Герой, наше королество в опасности, на землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, герой, спаси нас!");

MainWindow.Change\_Button\_Choice\_1("Как я \nдолжен это \nсделать?");

}

else if (MainWindow.Персонажи.Count == 0 && MainWindow.Персонажи[0].Пол == false)

{

MainWindow.ConsoleWriteLine($"- Героиня, наше королество в опасности, на землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, героиня, спаси нас!");

MainWindow.Change\_Button\_Choice\_1("Как я \nдолжна это \nсделать?");

}

else if (MainWindow.Персонажи.Count >= 1)

{

MainWindow.ConsoleWriteLine($"- Герои, наше королество в опасности, на землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, герои, спасите нас!");

MainWindow.Change\_Button\_Choice\_1("Как мы \nдолжны это \nсделать?");

}

MainWindow.Block\_Button\_Choice\_2();

MainWindow.Block\_Button\_Choice\_3();

}

if (Choices.Count == 1 && Choices[0] == 2)

{

if (MainWindow.Персонажи.Count == 0 && MainWindow.Персонажи[0].Пол == true)

{

MainWindow.ConsoleWriteLine($"- Герой, наше королество в опасности, на землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, герой, спаси нас!");

MainWindow.Change\_Button\_Choice\_1("Как я \nдолжен это \nсделать?");

}

else if (MainWindow.Персонажи.Count == 0 && MainWindow.Персонажи[0].Пол == false)

{

MainWindow.ConsoleWriteLine($"- Героиня, наше королество в опасности, на землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, героиня, спаси нас!");

MainWindow.Change\_Button\_Choice\_1("Как я \nдолжна это \nсделать?");

}

else if (MainWindow.Персонажи.Count >= 1)

{

MainWindow.ConsoleWriteLine($"- Герои, наше королество в опасности, на землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, герои, спасите нас!");

MainWindow.Change\_Button\_Choice\_1("Как мы \nдолжны это \nсделать?");

}

MainWindow.Block\_Button\_Choice\_2();

MainWindow.Block\_Button\_Choice\_3();

}

if (Choices.Count == 1 && Choices[0] == 3)

{

if (MainWindow.Персонажи.Count == 0 && MainWindow.Персонажи[0].Пол == true)

{

MainWindow.ConsoleWriteLine($"- Мы - маги королевства Мелромарк и оно в опасности. На землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, герой, спаси нас!");

MainWindow.Change\_Button\_Choice\_1("Как я \nдолжен это \nсделать?");

}

else if (MainWindow.Персонажи.Count == 0 && MainWindow.Персонажи[0].Пол == false)

{

MainWindow.ConsoleWriteLine($"- Мы - маги королевства Мелромарк и оно в опасности. На землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, героиня, спаси нас!");

MainWindow.Change\_Button\_Choice\_1("Как я \nдолжна это \nсделать?");

}

else if (MainWindow.Персонажи.Count >= 1)

{

MainWindow.ConsoleWriteLine($"- Мы - маги королевства Мелромарк и оно в опасности. На землях Мелромарка лежит тяжелая ноша, сражение с волнами монстров. " +

$"Мы пытались отбить волны своими силами, но треть населения нашего королевства была жестоко убита. Пожалуйста, герои, спасите нас!");

MainWindow.Change\_Button\_Choice\_1("Как мы \nдолжны это \nсделать?");

}

MainWindow.Block\_Button\_Choice\_2();

MainWindow.Block\_Button\_Choice\_3();

}

if (Choices.Count == 2)

{

MainWindow.ConsoleWriteLine($"- Единственное что мы можем вам дать это несколько рун, " +

$"усиливающие ваши способности и немного целебных и магических снадобий. Используйте их с умом!");

MainWindow.ПереченьПредметов.Add(new Предметы.Малое\_Зелье\_Лечения());

MainWindow.ПереченьПредметов.Add(new Предметы.Среднее\_Зелье\_Лечения());

MainWindow.ПереченьПредметов.Add(new Предметы.Большое\_Зелье\_Лечения());

MainWindow.ПереченьПредметов.Add(new Предметы.Бутылек\_Маны());

MainWindow.ПереченьПредметов.Add(new Предметы.Фласка\_маны());

MainWindow.ПереченьПредметов.Add(new Предметы.Банка\_маны());

MainWindow.ConsoleWriteLine("СОПИ: Предметы получены: Руна Здоровья (Ур. 1 из 3). " +

"Описание: Увеличивает количество здоровья у персонажей. Обладает множителем \"x1\"");

MainWindow.ConsoleWriteLine("СОПИ: Предметы получены: Руна Эффективности использования заклинаний (Ур. 1 из 3). " +

"Описание: Уменьшает количество затрат маны на использование заклинаний у персонажей. Обладает множителем \"x1.2\"");

MainWindow.ConsoleWriteLine("СОПИ: Предметы получены: Руна Урона (Ур. 1 из 3). " +

"Описание: Увеличивает количество наносимого урона персонажами. Обладает множителем \"x1\"");

MainWindow.Change\_Button\_Choice\_1("Спасибо!");

}

if (Choices.Count == 3)

{

MainWindow.ConsoleWriteLine($"- А теперь, пожалуйста, чтобы мы могли оценить насколько вы сильны, сразитесь с одним из нашей стражи при короле. Пройдёмте...");

MainWindow.ConsoleWriteLine($"СОПИ: Вы следуете за магами в тренировочный зал. Выходя из подвала, " +

$"вы впервые видите как роскошно выглядит поместье 21 века, вместе с широкими красными " +

$"коврами и прекрасными ухоженными садами, среди не менее величественных картин и гобеленов.");

MainWindow.Change\_Button\_Choice\_1("Продолжить \nследовать");

}

if (Choices.Count == 4)

{

MainWindow.ConsoleWriteLine("СОПИ: Вы следуете за магами почти сквозь всё поместье к огромному " +

"тренировочному залу где вас уже поджидала стража. Увидев подходящих магов самый первый из их ряда " +

"вышел на один шаг вперед, предъявляя свою кандидатуру для вашей проверки.");

MainWindow.ConsoleWriteLine("Вы будете сражаться с главой нашей стражи, покажите что умеете и пожалуйста, не убейте его. Он нам ещё нужен.");

MainWindow.ConsoleWriteLine("СОПИ: Ваша задача - опустить здоровье противника до 50%");

MainWindow.Change\_Button\_Choice\_1("[Принять \nвызов]");

}

if (Choices.Count == 5)

{

MainWindow.IsBattleMode = true;

MainWindow.Change\_Button\_Choice\_1("Атака");

MainWindow.Change\_Button\_Choice\_2("Использовать \nзаклинание");

MainWindow.Change\_Button\_Choice\_3("Защита");

MainWindow.Button\_GiveUp.IsEnabled = true;

MainWindow.СписокТекущихВрагов.Add(new Враги.ГлаваСтражи());

for (int i = 0; i < MainWindow.Персонажи.Count; i++)

{

Персонаж Персонаж = MainWindow.Персонажи.ElementAt(i);

MainWindow.ПорядокАтаки.Add(Персонаж);

}

foreach (Враги Враг in MainWindow.СписокТекущихВрагов)

{

MainWindow.ПорядокАтаки.Add(Враг);

}

Battle(MainWindow.СписокТекущихВрагов, MainWindow.ПорядокАтаки, MainWindow.СписокГероевВЗащите);

}

if (Choices.Count == 6)

{

MainWindow.ConsoleWriteLine("- Достаточно, мы увидели что хотели. Мы можем не переживать за своё королевство. В награду " +

"наши лекари вылечат вас, восполнят запасы маны и лично мы выдаём вам улучшенные руны. А сейчас идите отдыхайте. Поговорим о делах завтра.");

foreach (Персонаж персонаж in MainWindow.Персонажи)

{

персонаж.ИзменениеСостоянияЗдоровья("лечение", 1000);

}

MainWindow.ConsoleWriteLine("СОПИ: Здоровье всех персонажей восстановлено.");

foreach (Персонаж персонаж in MainWindow.Персонажи)

{

if (персонаж is Персонаж\_с\_магией)

{

(персонаж as Персонаж\_с\_магией).ИзменениеСостоянияМаны("восполнение", 1000);

}

}

MainWindow.ConsoleWriteLine("СОПИ: Мана всех персонажей восполнена.");

MainWindow.SetRuneRank(2);

MainWindow.ConsoleWriteLine("СОПИ: Показатели рун улучшены. ");

MainWindow.ConsoleWriteLine("СОПИ: Предметы улучшены: Руна Здоровья (Ур. 2 из 3). " +

"Описание: Увеличивает количество здоровья у персонажей. Обладает множителем \"x2\"");

MainWindow.ConsoleWriteLine("СОПИ: Предметы улучшены: Руна Эффективности использования заклинаний (Ур. 2 из 3). " +

"Описание: Уменьшает количество затрат маны на использование заклинаний у персонажей. Обладает множителем \"x1\"");

MainWindow.ConsoleWriteLine("СОПИ: Предметы улучшены: Руна Урона (Ур. 2 из 3). " +

"Описание: Увеличивает количество наносимого урона персонажами. Обладает множителем \"x2\"");

MainWindow.Block\_Button\_Choice\_2();

MainWindow.Block\_Button\_Choice\_3();

MainWindow.Change\_Button\_Choice\_1("[Пойти отдыхать]");

MainWindow.Change\_Button\_Choice\_2("");

MainWindow.Change\_Button\_Choice\_3("");

}

if (Choices.Count == 7)

{

MainWindow.ConsoleWriteLine("\n");

MainWindow.ConsoleWriteLine("\n");

MainWindow.ConsoleWriteLine("Демоверсия окончена.");

GameOver();

}

if (Choices.Count == 8)

{

Application.Current.Shutdown();

}

}

public void Battle(List<Враги> СписокТекущихВрагов, List<object> ПорядокАтаки, bool[] СписокГероевВЗащите)

{

if (MainWindow.ПорядокАтаки[OrderOf] is Персонаж\_с\_магией)

{

MainWindow.UnBlock\_Button\_Choice\_2();

}

else

{

MainWindow.Block\_Button\_Choice\_2();

}

MainWindow.UnBlock\_Button\_Choice\_3();

MainWindow.ПорядокАтаки = ПорядокАтаки;

if (AreEnemiesAlive(СписокТекущихВрагов))

{

if (AreHeroesAlive(MainWindow.Персонажи))

{

if (ПорядокАтаки[OrderOf] is Персонаж && (ПорядокАтаки[OrderOf] as Персонаж).Состояние[4] == true)

{

IncreaseOrderOf();

}

else if (ПорядокАтаки[OrderOf] is Враги)

{

if ((ПорядокАтаки[OrderOf] as Враги).IsFrozen == false)

{

Random random = new Random();

int СлучайныйПерсонаж = random.Next(0, MainWindow.Персонажи.Count);

СлучайныйПерсонаж = Regen(СлучайныйПерсонаж, MainWindow.Персонажи, random);

if (СписокГероевВЗащите[СлучайныйПерсонаж] == true)

{

MainWindow.Персонажи[СлучайныйПерсонаж].ИзменениеСостоянияЗдоровья("урон", (uint)Math.Round(0.5 \* (ПорядокАтаки[OrderOf] as Враги).СилаАтаки));

MainWindow.ConsoleWriteLine($"CОПИ: {(ПорядокАтаки[OrderOf] as Враги).ИмяВрага} нанёс {MainWindow.Персонажи[СлучайныйПерсонаж].Имя} " +

$"{0.5 \* (ПорядокАтаки[OrderOf] as Враги).СилаАтаки} единиц урона (а должен был: {(ПорядокАтаки[OrderOf] as Враги).СилаАтаки} единиц урона). " +

$"У {MainWindow.Персонажи[СлучайныйПерсонаж].Имя} осталось {MainWindow.Персонажи[СлучайныйПерсонаж].Очки\_Здоровья} единиц здоровья.");

}

else

{

MainWindow.Персонажи[СлучайныйПерсонаж].ИзменениеСостоянияЗдоровья("урон", (uint)(ПорядокАтаки[OrderOf] as Враги).СилаАтаки);

MainWindow.ConsoleWriteLine($"CОПИ: {(ПорядокАтаки[OrderOf] as Враги).ИмяВрага} нанёс {MainWindow.Персонажи[СлучайныйПерсонаж].Имя} " +

$"{(ПорядокАтаки[OrderOf] as Враги).СилаАтаки} единиц урона. У {MainWindow.Персонажи[СлучайныйПерсонаж].Имя} осталось " +

$"{MainWindow.Персонажи[СлучайныйПерсонаж].Очки\_Здоровья} единиц здоровья.");

}

if (MainWindow.Персонажи[СлучайныйПерсонаж].Очки\_Здоровья == 0)

{

if (MainWindow.Персонажи[СлучайныйПерсонаж].Пол == true)

{

MainWindow.ConsoleWriteLine($"{MainWindow.Персонажи[СлучайныйПерсонаж].Имя} был повержен {(ПорядокАтаки[OrderOf] as Враги).ИмяВрага}.");

}

else

{

MainWindow.ConsoleWriteLine($"{MainWindow.Персонажи[СлучайныйПерсонаж].Имя} была повержена {(ПорядокАтаки[OrderOf] as Враги).ИмяВрага}.");

}

}

}

IncreaseOrderOf();

if (MainWindow.ПорядокАтаки[OrderOf] is Персонаж\_с\_магией)

{

MainWindow.UnBlock\_Button\_Choice\_2();

}

else

{

MainWindow.Block\_Button\_Choice\_2();

}

MainWindow.UnBlock\_Button\_Choice\_3();

bool AreAnyHeroesLeft = AreHeroesAlive(MainWindow.Персонажи);

if (!AreAnyHeroesLeft)

{

GameOver();

}

}

}

else

{

GameOver(); //все герои умерли

}

}

else

{

MainWindow.IsBattleMode = false;

Choices\_Add(1);

Воспроизведение\_Шагов();

}

}

public bool AreEnemiesAlive(List<Враги> СписокТекущихВрагов)

{

if (СписокТекущихВрагов[0].ЗдоровьеВрага < 0.5 \* СписокТекущихВрагов[0].МаксимальноеЗдоровье && СписокТекущихВрагов[0] is Враги.ГлаваСтражи)

{

MainWindow.IsBattleMode = false;

MainWindow.СписокТекущихВрагов.Clear();

}

if (СписокТекущихВрагов.Count == 0)

{

return false;

}

else

{

return true;

}

}

public bool AreHeroesAlive(List<Персонаж> Персонажи)

{

int КоличествоУмершихГероев = 0;

for (int i = 0; i < Персонажи.Count; i++)

{

Персонаж персонаж = Персонажи[i];

if (персонаж.Состояние[4] == true)

{

КоличествоУмершихГероев++;

}

}

if (КоличествоУмершихГероев == 0)

{

return true;

}

else if ((КоличествоУмершихГероев == 1 || КоличествоУмершихГероев == 2) && Персонажи.Count != КоличествоУмершихГероев)

{

return true;

}

else

{

return false;

}

}

public void IncreaseOrderOf()

{

if (OrderOf >= MainWindow.ПорядокАтаки.Count - 1)

{

OrderOf = 0;

for (int i = 0; i < MainWindow.СписокГероевВЗащите.Length; i++)

{

MainWindow.СписокГероевВЗащите[i] = false;

}

foreach (object Существо in MainWindow.ПорядокАтаки)

{

if (Существо is Враги)

{

(Существо as Враги).IsFrozen = false;

}

}

}

else

{

OrderOf++;

}

if (MainWindow.ПорядокАтаки[OrderOf] is Персонаж\_с\_магией)

{

MainWindow.UnBlock\_Button\_Choice\_2();

}

else

{

MainWindow.Block\_Button\_Choice\_2();

}

}

public void GameOver()

{

MainWindow.IsBattleMode = false;

MainWindow.Button\_Choice\_2.Visibility = Visibility.Hidden;

MainWindow.Button\_Choice\_3.Visibility = Visibility.Hidden;

MainWindow.Button\_GiveUp.Visibility = Visibility.Hidden;

MainWindow.Button\_Inventory.Visibility = Visibility.Hidden;

int КоличествоМёртвыхГероев = 0;

foreach (Персонаж персонаж in MainWindow.Персонажи)

{

if (персонаж.Состояние[4] == true)

{

КоличествоМёртвыхГероев++;

}

}

if (КоличествоМёртвыхГероев == MainWindow.Персонажи.Count)

{

MainWindow.ConsoleWriteLine("СОПИ: Игра окончена, все герои умерли");

}

else

{

MainWindow.ConsoleWriteLine("СОПИ: Игра окончена.");

}

int ОбщийСчет = 0;

foreach (Персонаж персонаж in MainWindow.Персонажи)

{

if (персонаж.Пол == true)

{

MainWindow.ConsoleWriteLine($"СОПИ: В течении прохождения игры герой {персонаж.Имя} получил {персонаж.Очки\_Опыта} очков опыта.");

}

else if (персонаж.Пол == false)

{

MainWindow.ConsoleWriteLine($"СОПИ: В течении прохождения игры героиня {персонаж.Имя} получила {персонаж.Очки\_Опыта} очков опыта.");

}

ОбщийСчет += (int)персонаж.Очки\_Опыта;

}

MainWindow.ConsoleWriteLine($"СОПИ: Общий счет игрока составил: {ОбщийСчет} очков.");

MainWindow.Change\_Button\_Choice\_1("[Закрыть \nигру]");

}

private int Regen(int value, List<Персонаж> Персонажи, Random random)

{

if (Персонажи[value].Состояние[4] != true)

{

return value;

}

else

{

return Regen(random.Next(0, MainWindow.Персонажи.Count), Персонажи, random);

}

}

}

}

Attack.xaml:

<Window x:Class="Ролевая\_игра\_\_WPF\_.Attack"

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:Ролевая\_игра\_\_WPF\_"

mc:Ignorable="d"

Title="Выбрать цель для атаки" Height="130" Width="380" WindowStartupLocation="CenterScreen" ResizeMode="NoResize">

<Grid>

<Button x:Name="Button\_Attack" Content="Атаковать" HorizontalAlignment="Center" Margin="130,45,130,0" VerticalAlignment="Top" Width="105" Height="40" IsDefault="True" IsEnabled="False" Click="Button\_Attack\_Click"/>

<ComboBox x:Name="ComboBox\_EnemyList" Margin="21,10,21,0" VerticalAlignment="Top" SelectionChanged="ComboBox\_EnemyList\_SelectionChanged"/>

</Grid>

</Window>

Attack.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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.Shapes;

namespace Ролевая\_игра\_\_WPF\_

{

/// <summary>

/// Логика взаимодействия для Attack.xaml

/// </summary>

public partial class Attack : Window

{

List<Враги> СписокТекущихВрагов;

Персонаж АтакующийПерсонаж;

public int ЗначениеАтаки { get; private set; }

public int SelectedEnemy { get; private set; }

public bool IsEnemyKilled { get; private set; }

public Attack(List <Враги> СписокНаследованныхВрагов, Персонаж НаследованныйАтакующийПерсонаж)

{

InitializeComponent();

СписокТекущихВрагов = СписокНаследованныхВрагов;

foreach (Враги Враг in СписокТекущихВрагов)

{

ComboBox\_EnemyList.Items.Add(Враг.ИмяВрага);

}

ComboBox\_EnemyList.SelectedIndex = 0;

SelectedEnemy = 0;

Button\_Attack.IsEnabled = true;

АтакующийПерсонаж = НаследованныйАтакующийПерсонаж;

ЗначениеАтаки = АтакующийПерсонаж.ПолучитьЗначениеАтаки();

}

private void ComboBox\_EnemyList\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

Button\_Attack.IsEnabled = true;

SelectedEnemy = ComboBox\_EnemyList.SelectedIndex;

}

private void Button\_Attack\_Click(object sender, RoutedEventArgs e)

{

СписокТекущихВрагов[SelectedEnemy].ОтнятьЗдоровье(ЗначениеАтаки);

DialogResult = true;

Close();

if (ЗначениеАтаки > СписокТекущихВрагов[SelectedEnemy].ЗдоровьеВрага)

IsEnemyKilled = true;

}

public List<Враги> GetСписокТекущихВрагов() => СписокТекущихВрагов;

}

}

Inventory.xaml:

<Window x:Class="Ролевая\_игра\_\_WPF\_.Inventory"

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:Ролевая\_игра\_\_WPF\_"

mc:Ignorable="d"

Title="Инвентарь персонажей" Height="450" Width="800" ResizeMode="NoResize" WindowStartupLocation="CenterScreen">

<Grid>

<ListBox x:Name="Список\_Предметов" HorizontalAlignment="Left" Height="400" Margin="10,10,0,0" VerticalAlignment="Top" Width="584" SelectionChanged="Список\_Предметов\_SelectionChanged"/>

<Button x:Name="Button\_Use" Content="Использовать" HorizontalAlignment="Left" Margin="641,67,0,0" VerticalAlignment="Top" Width="100" Height="35" IsEnabled="False" IsDefault="True" Click="Button\_Use\_Click"/>

<ComboBox x:Name="Выпадающий\_Список\_Персонажей" HorizontalAlignment="Left" Margin="599,40,0,0" VerticalAlignment="Top" Width="185" SelectionChanged="Выпадающий\_Список\_Персонажей\_SelectionChanged"/>

<Label Content="Использовать на:" HorizontalAlignment="Left" Margin="599,10,0,0" VerticalAlignment="Top" Width="185"/>

<GroupBox Header="Информация о герое" HorizontalAlignment="Left" Height="98" Margin="599,122,0,0" VerticalAlignment="Top" Width="185">

<StackPanel Margin="0,0,-3,-1.4" Orientation="Vertical">

<Label x:Name="Info\_HP" Content="Здоровье: N/A" Margin="10,10,0,0"/>

<Label x:Name="Info\_MP" Content="Мана: N/A" Margin="10,10,0,0"/>

</StackPanel>

</GroupBox>

</Grid>

</Window>

Inventory.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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.Shapes;

namespace Ролевая\_игра\_\_WPF\_

{

/// <summary>

/// Логика взаимодействия для Inventory.xaml

/// </summary>

public partial class Inventory : Window

{

private List<Персонаж> Персонажи;

private List<Предметы.Зелье> ПереченьПредметов;

private List<string> НазванияПредметов = new List<string>();

public Inventory(List<Персонаж> Текущие\_Персонажи, List<Предметы.Зелье> ПереченьПредметов)

{

InitializeComponent();

Персонажи = Текущие\_Персонажи;

foreach (Персонаж Текущий\_Персонаж in Текущие\_Персонажи)

{

if (Текущий\_Персонаж is Персонаж)

{

Выпадающий\_Список\_Персонажей.Items.Add((Текущий\_Персонаж as Персонаж).Имя);

}

else

{

Выпадающий\_Список\_Персонажей.Items.Add((Текущий\_Персонаж as Персонаж\_с\_магией).Имя);

}

}

this.ПереченьПредметов = ПереченьПредметов;

ReloadInventoryList();

Выпадающий\_Список\_Персонажей.SelectedIndex = 0;

}

private void Список\_Предметов\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (Список\_Предметов.SelectedItem != null && Выпадающий\_Список\_Персонажей.SelectedIndex != -1)

{

if (ПереченьПредметов[Список\_Предметов.SelectedIndex] is Предметы.Зелье\_маны && Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] is Персонаж\_с\_магией)

{

Button\_Use.IsEnabled = true;

}

else

if (Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex].Очки\_Здоровья != Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex].Максимальное\_здоровье

&& ПереченьПредметов[Список\_Предметов.SelectedIndex] is Предметы.Зелье\_Лечения)

{

Button\_Use.IsEnabled = true;

}

else

if (Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] is Персонаж\_с\_магией

&&

(Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] as Персонаж\_с\_магией).Очки\_Маны

!=

(Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] as Персонаж\_с\_магией).Максимальная\_мана

&&

ПереченьПредметов[Список\_Предметов.SelectedIndex] is Предметы.Зелье\_маны)

{

Button\_Use.IsEnabled = true;

}

else

{

Button\_Use.IsEnabled = false;

}

}

}

private void Выпадающий\_Список\_Персонажей\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (Список\_Предметов.SelectedItem != null && Выпадающий\_Список\_Персонажей.SelectedIndex != -1)

{

if (ПереченьПредметов[Список\_Предметов.SelectedIndex] is Предметы.Зелье\_маны && Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] is Персонаж\_с\_магией)

{

Button\_Use.IsEnabled = true;

}

else

if (Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex].Очки\_Здоровья != Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex].Максимальное\_здоровье

&& ПереченьПредметов[Список\_Предметов.SelectedIndex] is Предметы.Зелье\_Лечения)

{

Button\_Use.IsEnabled = true;

}

else

if (Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] is Персонаж\_с\_магией

&&

(Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] as Персонаж\_с\_магией).Очки\_Маны

!=

(Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] as Персонаж\_с\_магией).Максимальная\_мана

&&

ПереченьПредметов[Список\_Предметов.SelectedIndex] is Предметы.Зелье\_маны)

{

Button\_Use.IsEnabled = true;

}

else

{

Button\_Use.IsEnabled = false;

}

}

UpdateFormWindow();

}

private void Button\_Use\_Click(object sender, RoutedEventArgs e)

{

if ((string)Список\_Предметов.SelectedItem == "[+25 ОЗ] Малое зелье лечения")

{

foreach (Персонаж Текущий\_Персонаж in Персонажи)

{

if ((string)Выпадающий\_Список\_Персонажей.SelectedItem == (Текущий\_Персонаж as Персонаж).Имя)

{

Текущий\_Персонаж.ИзменениеСостоянияЗдоровья("лечение", 25);

}

}

ПереченьПредметов.RemoveAt(SearchForItem((string)Список\_Предметов.SelectedItem));

ReloadInventoryList();

}

else if ((string)Список\_Предметов.SelectedItem == "[+50 ОЗ] Среднее зелье лечения")

{

foreach (Персонаж Текущий\_Персонаж in Персонажи)

{

if ((string)Выпадающий\_Список\_Персонажей.SelectedItem == (Текущий\_Персонаж as Персонаж).Имя)

{

Текущий\_Персонаж.ИзменениеСостоянияЗдоровья("лечение", 50);

}

}

ПереченьПредметов.RemoveAt(SearchForItem((string)Список\_Предметов.SelectedItem));

ReloadInventoryList();

}

else if ((string)Список\_Предметов.SelectedItem == "[+75 ОЗ] Большое зелье лечения")

{

foreach (Персонаж Текущий\_Персонаж in Персонажи)

{

if ((string)Выпадающий\_Список\_Персонажей.SelectedItem == (Текущий\_Персонаж as Персонаж).Имя)

{

Текущий\_Персонаж.ИзменениеСостоянияЗдоровья("лечение", 75);

}

}

ПереченьПредметов.RemoveAt(SearchForItem((string)Список\_Предметов.SelectedItem));

ReloadInventoryList();

}

else if ((string)Список\_Предметов.SelectedItem == "[+25 ОМ] Бутылек Маны")

{

foreach (Персонаж Текущий\_Персонаж in Персонажи)

{

if ((string)Выпадающий\_Список\_Персонажей.SelectedItem == (Текущий\_Персонаж as Персонаж\_с\_магией).Имя)

{

(Текущий\_Персонаж as Персонаж\_с\_магией).ИзменениеСостоянияМаны("восполнение", 25);

}

}

ПереченьПредметов.RemoveAt(SearchForItem((string)Список\_Предметов.SelectedItem));

ReloadInventoryList();

}

else if ((string)Список\_Предметов.SelectedItem == "[+50 ОМ] Фласка маны")

{

foreach (Персонаж Текущий\_Персонаж in Персонажи)

{

if ((string)Выпадающий\_Список\_Персонажей.SelectedItem == (Текущий\_Персонаж as Персонаж\_с\_магией).Имя)

{

(Текущий\_Персонаж as Персонаж\_с\_магией).ИзменениеСостоянияМаны("восполнение", 50);

}

}

ПереченьПредметов.RemoveAt(SearchForItem((string)Список\_Предметов.SelectedItem));

ReloadInventoryList();

}

else if ((string)Список\_Предметов.SelectedItem == "[+75 ОМ] Банка маны")

{

foreach (Персонаж Текущий\_Персонаж in Персонажи)

{

if ((string)Выпадающий\_Список\_Персонажей.SelectedItem == (Текущий\_Персонаж as Персонаж\_с\_магией).Имя)

{

(Текущий\_Персонаж as Персонаж\_с\_магией).ИзменениеСостоянияМаны("восполнение", 75);

}

}

ПереченьПредметов.RemoveAt(SearchForItem((string)Список\_Предметов.SelectedItem));

ReloadInventoryList();

}

}

private void ReloadInventoryList()

{

Список\_Предметов.Items.Clear();

foreach (Предметы.Зелье Предмет in ПереченьПредметов.OrderBy(it => it.Название))

{

Список\_Предметов.Items.Add((Предмет as Предметы.Зелье).Название);

НазванияПредметов.Add(Предмет.Название);

}

UpdateFormWindow();

}

private int SearchForItem(string Название\_искомого\_объекта)

{

for (int i = 0; i < НазванияПредметов.Count; i++)

{

if (НазванияПредметов[i] == Название\_искомого\_объекта)

{

return i;

}

}

return -1;

}

public List<Предметы.Зелье> GetInventory()

{

return ПереченьПредметов;

}

private void UpdateFormWindow()

{

if (Выпадающий\_Список\_Персонажей.SelectedIndex != -1)

{

Info\_HP.Content = "Здоровье: " + Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex].Очки\_Здоровья + " ОЗ / " + Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex].Максимальное\_здоровье + " ОЗ";

if (Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] is Персонаж\_с\_магией)

{

Info\_MP.Content = "Мана: " + (Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] as Персонаж\_с\_магией).Очки\_Маны + " ОМ / " + (Персонажи[Выпадающий\_Список\_Персонажей.SelectedIndex] as Персонаж\_с\_магией).Максимальная\_мана + " ОМ";

}

else

{

Info\_MP.Content = "Мана: N/A";

}

}

}

}

}

Magic.xaml:

<Window x:Class="Ролевая\_игра\_\_WPF\_.Magic"

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:Ролевая\_игра\_\_WPF\_"

mc:Ignorable="d"

Title="Magic" Height="212" Width="350" ResizeMode="NoResize" WindowStartupLocation="CenterScreen">

<Grid>

<ComboBox x:Name="ComboBox\_ВыборЗаклинания" HorizontalAlignment="Left" Margin="10,41,0,0" VerticalAlignment="Top" Width="159" SelectionChanged="ComboBox\_ВыборЗаклинания\_SelectionChanged"/>

<Label Content="Выбор заклинания" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="159"/>

<Label Content="Цель заклинания" HorizontalAlignment="Left" Margin="174,10,0,0" VerticalAlignment="Top" Width="159"/>

<ComboBox x:Name="ComboBox\_ЦельЗаклинания" HorizontalAlignment="Left" Margin="174,41,0,0" VerticalAlignment="Top" Width="159" SelectionChanged="ComboBox\_ЦельЗаклинания\_SelectionChanged"/>

<Button x:Name="Button\_Использовать" Content="Использовать" HorizontalAlignment="Center" Margin="0,0,0,10" VerticalAlignment="Bottom" Width="150" Height="35" Click="Button\_Использовать\_Click"/>

<TextBlock x:Name="TextBlock\_Подсказки" HorizontalAlignment="Left" Margin="10,69,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="63" Width="323"/>

</Grid>

</Window>

Magic.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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.Shapes;

namespace Ролевая\_игра\_\_WPF\_

{

/// <summary>

/// Логика взаимодействия для Magic.xaml

/// </summary>

public partial class Magic : Window

{

private List<Персонаж> СписокПерсонажей;

private List<Враги> СписокВрагов;

private List<object> ПорядокАтаки;

private int OrderOf;

private double[] МножительСилыРун;

public string ConsoleOutput { get; private set; }

public Magic(List<Персонаж> СписокПерсонажей, List<Враги> СписокВрагов, List<object> ПорядокАтаки, int OrderOf, double[] МножительСилыРун)

{

InitializeComponent();

this.СписокПерсонажей = СписокПерсонажей;

this.СписокВрагов = СписокВрагов;

this.ПорядокАтаки = ПорядокАтаки;

this.OrderOf = OrderOf;

this.МножительСилыРун = МножительСилыРун;

ComboBox\_ВыборЗаклинания.Items.Add(Заклинания.Лечение.Название);

ComboBox\_ВыборЗаклинания.Items.Add(Заклинания.Воскрешение.Название);

ComboBox\_ВыборЗаклинания.Items.Add(Заклинания.Огненный\_Шар.Название);

ComboBox\_ВыборЗаклинания.Items.Add(Заклинания.Заморозка.Название);

ComboBox\_ВыборЗаклинания.SelectedIndex = 0;

TextBlock\_Подсказки.Text = "Лечит персонажа на 25 единиц здоровья";

}

private void ComboBox\_ВыборЗаклинания\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 0)

{

ComboBox\_ЦельЗаклинания.Items.Clear();

foreach (Персонаж персонаж in СписокПерсонажей)

{

if (персонаж.Состояние[4] == false)

{

ComboBox\_ЦельЗаклинания.Items.Add(персонаж.Имя);

}

}

TextBlock\_Подсказки.Text = "Лечит персонажа на 25 единиц здоровья";

Button\_Использовать.IsEnabled = false;

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 0 && ComboBox\_ЦельЗаклинания.SelectedIndex != -1)

{

TextBlock\_Подсказки.Text = $"Вылечит {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} на 25 единиц здоровья. " +

$"Сейчас {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Очки\_Здоровья} ОЗ / {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Максимальное\_здоровье} ОЗ.";

Button\_Использовать.IsEnabled = true;

}

}

else if (ComboBox\_ВыборЗаклинания.SelectedIndex == 1)

{

ComboBox\_ЦельЗаклинания.Items.Clear();

foreach (Персонаж персонаж in СписокПерсонажей)

{

if (персонаж.Состояние[4] == true)

{

ComboBox\_ЦельЗаклинания.Items.Add(персонаж.Имя);

}

}

TextBlock\_Подсказки.Text = "Воскресит персонажа и вылечит его на 50 единиц здоровья";

Button\_Использовать.IsEnabled = false;

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 0 && ComboBox\_ЦельЗаклинания.SelectedIndex != -1)

{

TextBlock\_Подсказки.Text = $"Воскресит {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} и вылечит его на 25 единиц здоровья.";

Button\_Использовать.IsEnabled = true;

}

}

else if(ComboBox\_ВыборЗаклинания.SelectedIndex == 2)

{

ComboBox\_ЦельЗаклинания.Items.Clear();

foreach (Враги Враг in СписокВрагов)

{

ComboBox\_ЦельЗаклинания.Items.Add(Враг.ИмяВрага);

}

TextBlock\_Подсказки.Text = "Создаст и запустит огненный шар который нанесет 50 единиц урона по врагу";

Button\_Использовать.IsEnabled = false;

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 0 && ComboBox\_ЦельЗаклинания.SelectedIndex != -1)

{

TextBlock\_Подсказки.Text = $"Создаст и запустит огненный шар который нанесет 50 единиц урона по {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага}." +

$"Сейчас {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ЗдоровьеВрага}/{СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].МаксимальноеЗдоровье}";

Button\_Использовать.IsEnabled = true;

}

}

else if(ComboBox\_ВыборЗаклинания.SelectedIndex == 3)

{

ComboBox\_ЦельЗаклинания.Items.Clear();

foreach (Враги Враг in СписокВрагов)

{

ComboBox\_ЦельЗаклинания.Items.Add(Враг.ИмяВрага);

}

TextBlock\_Подсказки.Text = "Заморозит врага на 1 ход. (Выбранный враг пропустит ход)";

Button\_Использовать.IsEnabled = false;

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 0 && ComboBox\_ЦельЗаклинания.SelectedIndex != -1)

{

TextBlock\_Подсказки.Text = $"Заморозит {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага} на 1 ход. (Выбранный {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага} пропустит ход)";

Button\_Использовать.IsEnabled = true;

}

}

}

private void ComboBox\_ЦельЗаклинания\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 0)

{

TextBlock\_Подсказки.Text = "Лечит персонажа на 25 единиц здоровья";

Button\_Использовать.IsEnabled = false;

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 0 && ComboBox\_ЦельЗаклинания.SelectedIndex != -1)

{

TextBlock\_Подсказки.Text = $"Вылечит {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} на 25 единиц здоровья. " +

$"Сейчас {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Очки\_Здоровья} ОЗ / {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Максимальное\_здоровье} ОЗ.";

Button\_Использовать.IsEnabled = true;

}

}

else if (ComboBox\_ВыборЗаклинания.SelectedIndex == 1)

{

TextBlock\_Подсказки.Text = "Воскресит персонажа и вылечит его на 50 единиц здоровья";

Button\_Использовать.IsEnabled = false;

if (ComboBox\_ЦельЗаклинания.SelectedIndex != -1)

{

TextBlock\_Подсказки.Text = $"Воскресит {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} и вылечит его на 25 единиц здоровья.";

Button\_Использовать.IsEnabled = true;

}

}

else if(ComboBox\_ВыборЗаклинания.SelectedIndex == 2)

{

TextBlock\_Подсказки.Text = "Создаст и запустит огненный шар который нанесет 50 единиц урона по врагу";

Button\_Использовать.IsEnabled = false;

if (ComboBox\_ЦельЗаклинания.SelectedIndex != -1)

{

TextBlock\_Подсказки.Text = $"Создаст и запустит огненный шар который нанесет 50 единиц урона по {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага}. " +

$"Сейчас {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ЗдоровьеВрага}/{СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].МаксимальноеЗдоровье}";

Button\_Использовать.IsEnabled = true;

}

}

else if(ComboBox\_ВыборЗаклинания.SelectedIndex == 3)

{

TextBlock\_Подсказки.Text = "Заморозит врага на 1 ход. (Выбранный враг пропустит ход)";

Button\_Использовать.IsEnabled = false;

if (ComboBox\_ЦельЗаклинания.SelectedIndex != -1)

{

TextBlock\_Подсказки.Text = $"Заморозит {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага} на 1 ход. (Выбранный {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага} пропустит ход)";

Button\_Использовать.IsEnabled = true;

}

}

}

private void Button\_Использовать\_Click(object sender, RoutedEventArgs e)

{

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 0)

{

new Заклинания.Лечение(ПорядокАтаки[OrderOf] as Персонаж\_с\_магией, СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex], МножительСилыРун[1]);

if ((ПорядокАтаки[OrderOf] as Персонаж).Пол == true)

{

ConsoleOutput = $"{(ПорядокАтаки[OrderOf] as Персонаж).Имя} вылечил {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} на 25 единиц здоровья. " +

$"Сейчас у {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} " +

$"{СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Очки\_Здоровья} ОЗ / {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Максимальное\_здоровье} ОЗ";

}

else

{

ConsoleOutput = $"{(ПорядокАтаки[OrderOf] as Персонаж).Имя} вылечила {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} на 25 единиц здоровья. " +

$"Сейчас у {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} " +

$"{СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Очки\_Здоровья} ОЗ / {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Максимальное\_здоровье} ОЗ";

}

}

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 1)

{

new Заклинания.Воскрешение(ПорядокАтаки[OrderOf] as Персонаж\_с\_магией, СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex], МножительСилыРун[1]);

if ((ПорядокАтаки[OrderOf] as Персонаж).Пол == true)

{

ConsoleOutput = $"{(ПорядокАтаки[OrderOf] as Персонаж).Имя} воскресил и вылечил {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} на 50 единиц здоровья. " +

$"Сейчас у {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} " +

$"{СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Очки\_Здоровья} ОЗ / {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Максимальное\_здоровье} ОЗ";

}

else

{

ConsoleOutput = $"{(ПорядокАтаки[OrderOf] as Персонаж).Имя} воскресила и вылечила {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} на 50 единиц здоровья. " +

$"Сейчас у {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Имя} " +

$"{СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Очки\_Здоровья} ОЗ / {СписокПерсонажей[ComboBox\_ЦельЗаклинания.SelectedIndex].Максимальное\_здоровье} ОЗ";

}

}

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 2)

{

new Заклинания.Огненный\_Шар(ПорядокАтаки[OrderOf] as Персонаж\_с\_магией, СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex], МножительСилыРун[1]);

ConsoleOutput = $"{(ПорядокАтаки[OrderOf] as Персонаж\_с\_магией).Имя} запустил огненный шар в " +

$"{СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага} и нанёс ему 50 единиц урона. " +

$"Сейчас у {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага} " +

$"{СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ЗдоровьеВрага} ОЗ / {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].МаксимальноеЗдоровье} ОЗ";

}

if (ComboBox\_ВыборЗаклинания.SelectedIndex == 3)

{

Заклинания.Заморозка Заклинание = new Заклинания.Заморозка(ПорядокАтаки[OrderOf] as Персонаж\_с\_магией, СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex], МножительСилыРун[1]);

ConsoleOutput = $"{(ПорядокАтаки[OrderOf] as Персонаж\_с\_магией).Имя} заморозил {СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага}. " +

$"{СписокВрагов[ComboBox\_ЦельЗаклинания.SelectedIndex].ИмяВрага} попускает ход.";

}

DialogResult = true;

}

}

}

MainWindow.xaml:

<Window 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" mc:Ignorable="d" x:Class="Ролевая\_игра\_\_WPF\_.MainWindow" Title="Ролевая игра" Height="450" Width="800" WindowStartupLocation="CenterScreen" ResizeMode="NoResize">

<Grid Margin="17,15,12.6,15">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="153\*" />

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="35\*" />

</Grid.ColumnDefinitions>

<Rectangle Fill="#FFEBEBEB" HorizontalAlignment="Left" Height="380" Stroke="#FF000000" VerticalAlignment="Top" Width="613" Margin="0,1,0,0" Grid.ColumnSpan="2" />

<ScrollViewer x:Name="ScrollViewer" Margin="10,10,0,0" Height="354" Width="590" HorizontalAlignment="Left" VerticalAlignment="Top" VerticalScrollBarVisibility="Auto" >

<TextBlock x:Name="Console" Background="{x:Null}" Width="560" TextWrapping="Wrap" ScrollViewer.CanContentScroll="True" ScrollViewer.VerticalScrollBarVisibility="Auto" />

</ScrollViewer>

<Button x:Name="Button\_Inventory" Content="" Grid.Column="2" HorizontalAlignment="Left" Margin="53,302,0,0" VerticalAlignment="Top" Width="35" Height="35" IsEnabled="False" Click="Button\_Inventory\_Click" d:IsLocked="True">

<Button.Background>

<ImageBrush Stretch="UniformToFill" ImageSource="Resources/Inventary.png"/>

</Button.Background>

</Button>

<Grid x:Name="Menu\_Grid" Grid.Column="2" Margin="8,1,0,0" d:IsLocked="True" d:IsHidden="True">

<Button x:Name="MainWindow\_Button\_AddHero" Content="Добавить персонажа" HorizontalAlignment="Left" VerticalAlignment="Top" Width="125" Height="38" IsDefault="True" Click="MainWindow\_Button\_AddHero\_Click" />

<Button x:Name="MainWindow\_Button\_ShowInfo" Content="Состояние текущего &#xA;героя" HorizontalAlignment="Left" Margin="0,43,0,0" VerticalAlignment="Top" Width="125" Height="38" IsEnabled="False" AutomationProperties.IsColumnHeader="True" Click="MainWindow\_Button\_ShowInfo\_Click" />

<Button x:Name="MainWindow\_Button\_SwitchHero" Content="Сменить героя" HorizontalAlignment="Left" Margin="0,86,0,0" VerticalAlignment="Top" Width="125" Height="38" IsEnabled="False" AutomationProperties.IsColumnHeader="True" Click="MainWindow\_Button\_SwitchHero\_Click" />

<Button x:Name="Button\_Save" Content="" HorizontalAlignment="Left" Margin="5,301,0,0" VerticalAlignment="Top" Width="35" Height="35" Click="Button\_Save\_Click" Foreground="{x:Null}" BorderBrush="Black" IsEnabled="False">

<Button.Background>

<ImageBrush ImageSource="Resources/Save.png" Stretch="UniformToFill" TileMode="Tile"/>

</Button.Background>

</Button>

<Button x:Name="Button\_Load" Content="" HorizontalAlignment="Left" Margin="85,301,0,0" VerticalAlignment="Top" Width="35" Height="35" Click="Button\_Load\_Click" ForceCursor="True">

<Button.Background>

<ImageBrush ImageSource="Resources/Load.png" Stretch="UniformToFill"/>

</Button.Background>

</Button>

<Button x:Name="MainWindow\_Button\_GoForAdventure" Content="Отправиться на &#xA;приключения" HorizontalAlignment="Left" Margin="0,341,0,0" VerticalAlignment="Top" Width="125" Height="38" IsEnabled="False" Click="MainWindow\_Button\_GoForAdventure\_Click" />

</Grid>

<Grid x:Name="Adventure\_Grid" Margin="8,1,0,0" Grid.Column="2">

<Button x:Name="Button\_Choice\_1" Content="Button" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="106" Height="70" Click="Button\_Choice\_1\_Click"/>

<Button x:Name="Button\_Choice\_2" Content="Button" HorizontalAlignment="Left" Margin="10,85,0,0" VerticalAlignment="Top" Width="106" Height="70" Click="Button\_Choice\_2\_Click"/>

<Button x:Name="Button\_Choice\_3" Content="Button" HorizontalAlignment="Left" Margin="10,160,0,0" VerticalAlignment="Top" Width="106" Height="70" Click="Button\_Choice\_3\_Click"/>

<Button x:Name="Button\_GiveUp" Content="Сдаться / Убежать" HorizontalAlignment="Left" Margin="0,341,0,0" VerticalAlignment="Top" Width="125" Height="38" IsEnabled="False" Click="Button\_GiveUp\_Click" />

</Grid>

</Grid>

</Window>

MainWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Windows;

using System.Windows.Controls;

namespace Ролевая\_игра\_\_WPF\_

{

public partial class MainWindow : Window

{

private AddHero AddHeroWindow;

private Inventory InventoryWindow;

public List<Персонаж> Персонажи = new List<Персонаж>();

public List<bool> Обладает\_магией = new List<bool>();

public List<Предметы.Зелье> ПереченьПредметов = new List<Предметы.Зелье>();

private string ConsoleBuffer;

private int ТекущийПерсонаж;

public int NextAdventureIs = 0;

private AdventureScripts adventureScripts;

private bool[] AdventureScriptsNonRepeat = new bool[12];

/// <summary>

/// Здоровье: [x1] [x2] [x3]

/// Эффективность использования заклинаний: [x1.2] [x1] [x0.75]

/// Урон: [x1] [x1,5] [x2]

/// </summary>

public double[] МножительСилыРун = new double[3] { 1, 1.2, 1 };

public bool IsBattleMode { get; set; } = false;

public List<object> ПорядокАтаки = new List<object>();

public List<Враги> СписокТекущихВрагов = new List<Враги>();

private Attack AttackDialog;

public bool[] СписокГероевВЗащите;

public MainWindow()

{

InitializeComponent();

Console.Text = "Добро пожаловать, герой. Я твой рассказчик, Система Одностороннего Повествования Истории, но можешь называть меня СОПИ ♥. \n" +

"Для начала предалагаю начать с создания твоего героя или загрузить свое прохождение из прошлой жизни.\n";

ТекущийПерсонаж = 0;

Menu\_Grid.Visibility = Visibility.Visible;

Adventure\_Grid.Visibility = Visibility.Hidden;

KeyDown += (e, args) =>

{

if (args.Key == System.Windows.Input.Key.F1)

System.Diagnostics.Process.Start("HelpFile.chm");

};

}

private void MainWindow\_Button\_AddHero\_Click(object sender, RoutedEventArgs e)

{

AddHeroWindow = new AddHero(Персонажи.Count, МножительСилыРун);

AddHeroWindow.ShowDialog();

bool? dialogResult = AddHeroWindow.DialogResult;

bool flag = true;

if (dialogResult.GetValueOrDefault() == flag & dialogResult.HasValue)

{

Персонаж ВозможныйПерсонаж = AddHeroWindow.Получить\_Персонажа();

if (ВозможныйПерсонаж != null)

{

Персонажи.Add(AddHeroWindow.Получить\_Персонажа());

Обладает\_магией.Add(false);

bool пол = AddHeroWindow.Пол;

if (пол)

{

ConsoleWriteLine($"Появился новый герой! Он {((Персонажи[Персонажи.Count - 1] as Персонаж) as Персонаж).Раса} в возрасте " +

$"{Персонаж.Лет\_Лета\_Год((Персонажи[Персонажи.Count - 1] as Персонаж).Возраст)} " +

$"и зовут его {(Персонажи[Персонажи.Count - 1] as Персонаж).Имя}\n");

}

else

{

ConsoleWriteLine($"Появилась новая героиня! Она {(Персонажи[Персонажи.Count - 1] as Персонаж).Раса} в возрасте " +

$"{Персонаж.Лет\_Лета\_Год((Персонажи[Персонажи.Count - 1] as Персонаж).Возраст)} " +

$"и зовут её {(Персонажи[Персонажи.Count - 1] as Персонаж).Имя}\n");

}

}

else //Персонажи с магией

{

Персонажи.Add(AddHeroWindow.Получить\_Персонажа\_с\_магией());

Обладает\_магией.Add(true);

bool пол2 = AddHeroWindow.Пол;

if (пол2)

{

ConsoleWriteLine($"Появился новый герой! Он {(Персонажи[Персонажи.Count - 1] as Персонаж\_с\_магией).Раса} с познаниями в магии и в возрасте " +

$"{Персонаж.Лет\_Лета\_Год((Персонажи[Персонажи.Count - 1] as Персонаж\_с\_магией).Возраст)}, " +

$"а зовут его {(Персонажи[Персонажи.Count - 1] as Персонаж\_с\_магией).Имя}\n");

}

else

{

ConsoleWriteLine($"Появилась новая героиня! Она {(Персонажи[Персонажи.Count - 1] as Персонаж\_с\_магией).Раса} с познаниями в магии и в возрасте " +

$"{Персонаж.Лет\_Лета\_Год((Персонажи[Персонажи.Count - 1] as Персонаж\_с\_магией).Возраст)}, " +

$"а зовут её {(Персонажи[Персонажи.Count - 1] as Персонаж\_с\_магией).Имя}\n");

}

}

MainWindow\_Button\_ShowInfo.IsEnabled = true;

Button\_Save.IsEnabled = true;

Button\_Inventory.IsEnabled = true;

MainWindow\_Button\_GoForAdventure.IsEnabled = true;

ТекущийПерсонаж = Персонажи.Count - 1;

}

if (Персонажи.Count == 1 || Персонажи.Count == 2)

{

MainWindow\_Button\_SwitchHero.IsEnabled = true;

}

if (Персонажи.Count == 3)

{

MainWindow\_Button\_SwitchHero.IsEnabled = true;

MainWindow\_Button\_AddHero.IsEnabled = false;

}

}

private void MainWindow\_Button\_ShowInfo\_Click(object sender, RoutedEventArgs e)

{

if ((string)MainWindow\_Button\_ShowInfo.Content == "Вернуться к \nповествованию")

{

MainWindow\_Button\_ShowInfo.Content = "Состояние текущего \nгероя";

Button\_Inventory.IsEnabled = true;

Button\_Load.IsEnabled = true;

Button\_Save.IsEnabled = true;

Console.Text = ConsoleBuffer;

MainWindow\_Button\_SwitchHero.IsEnabled = true;

MainWindow\_Button\_GoForAdventure.IsEnabled = true;

}

else

{

MainWindow\_Button\_ShowInfo.Content = "Вернуться к \nповествованию";

Button\_Inventory.IsEnabled = false;

Button\_Load.IsEnabled = false;

Button\_Save.IsEnabled = false;

MainWindow\_Button\_SwitchHero.IsEnabled = false;

MainWindow\_Button\_GoForAdventure.IsEnabled = false;

ConsoleBuffer = Console.Text;

if (Обладает\_магией[ТекущийПерсонаж] == false)

{

Console.Text = (Персонажи[ТекущийПерсонаж] as Персонаж).ToString();

}

else //Персонажи с магией

{

Console.Text = (Персонажи[ТекущийПерсонаж] as Персонаж\_с\_магией).ToString();

}

}

}

private void MainWindow\_Button\_SwitchHero\_Click(object sender, RoutedEventArgs e)

{

if (ТекущийПерсонаж == Персонажи.Count - 1)

{

ТекущийПерсонаж = 0;

ConsoleWriteLine($"Инициативу принимает {(Персонажи[ТекущийПерсонаж] as Персонаж).Имя}\n");

}

else

{

ТекущийПерсонаж++;

ConsoleWriteLine($"Инициативу принимает {(Персонажи[ТекущийПерсонаж] as Персонаж).Имя}\n");

}

}

#region SaveLoad\_Region

private void Button\_Save\_Click(object sender, RoutedEventArgs e)

{

//Сохранение информации о персонажах

try

{

DataSet ds = new DataSet();

DataTable dt = new DataTable

{

TableName = "Персонажи"

};

dt.Columns.Add("ID");

dt.Columns.Add("Имя");

dt.Columns.Add("Раса");

dt.Columns.Add("Возраст");

dt.Columns.Add("Пол");

dt.Columns.Add("Обладает\_магией");

dt.Columns.Add("Ослаблен");

dt.Columns.Add("Болен");

dt.Columns.Add("Отравлен");

dt.Columns.Add("Парализован");

dt.Columns.Add("Мёртв");

dt.Columns.Add("Может\_говорить");

dt.Columns.Add("Может\_двигаться");

dt.Columns.Add("Максимальное\_здоровье");

dt.Columns.Add("Очки\_здоровья");

dt.Columns.Add("Очки\_опыта");

dt.Columns.Add("Максимальная\_мана");

dt.Columns.Add("Очки\_маны");

ds.Tables.Add(dt);

for (int i = 0; i < Персонажи.Count; i++)

{

if (Обладает\_магией[i] == false)

{

Персонаж персонаж = Персонажи[i] as Персонаж;

DataRow строка = ds.Tables["Персонажи"].NewRow();

строка["ID"] = персонаж.ID;

строка["Имя"] = персонаж.Имя;

строка["Раса"] = персонаж.Раса;

строка["Возраст"] = персонаж.Возраст;

строка["Пол"] = персонаж.Пол;

строка["Обладает\_магией"] = "False";

строка["Ослаблен"] = персонаж.Состояние[0];

строка["Болен"] = персонаж.Состояние[1];

строка["Отравлен"] = персонаж.Состояние[2];

строка["Парализован"] = персонаж.Состояние[3];

строка["Мёртв"] = персонаж.Состояние[4];

строка["Может\_говорить"] = персонаж.Может\_говорить;

строка["Может\_двигаться"] = персонаж.Может\_двигаться;

строка["Максимальное\_здоровье"] = персонаж.Максимальное\_здоровье;

строка["Очки\_здоровья"] = персонаж.Очки\_Здоровья;

строка["Очки\_опыта"] = персонаж.Очки\_Опыта;

ds.Tables["Персонажи"].Rows.Add(строка);

}

else

{

Персонаж\_с\_магией персонаж = Персонажи[i] as Персонаж\_с\_магией;

DataRow строка = ds.Tables["Персонажи"].NewRow();

строка["ID"] = персонаж.ID;

строка["Имя"] = персонаж.Имя;

строка["Раса"] = персонаж.Раса;

строка["Возраст"] = персонаж.Возраст;

строка["Пол"] = персонаж.Пол;

строка["Обладает\_магией"] = "True";

строка["Ослаблен"] = персонаж.Состояние[0];

строка["Болен"] = персонаж.Состояние[1];

строка["Отравлен"] = персонаж.Состояние[2];

строка["Парализован"] = персонаж.Состояние[3];

строка["Мёртв"] = персонаж.Состояние[4];

строка["Может\_говорить"] = персонаж.Может\_говорить;

строка["Может\_двигаться"] = персонаж.Может\_двигаться;

строка["Максимальное\_здоровье"] = персонаж.Максимальное\_здоровье;

строка["Очки\_здоровья"] = персонаж.Очки\_Здоровья;

строка["Очки\_опыта"] = персонаж.Очки\_Опыта;

строка["Максимальная\_мана"] = персонаж.Максимальная\_мана;

строка["Очки\_маны"] = персонаж.Очки\_Маны;

ds.Tables["Персонажи"].Rows.Add(строка);

}

}

//сохранение информации о тексте в консоли

dt = new DataTable

{

TableName = "Консоль"

};

dt.Columns.Add("Консоль\_Текст");

ds.Tables.Add(dt);

DataRow СтрокаКонсоли = ds.Tables["Консоль"].NewRow();

СтрокаКонсоли["Консоль\_Текст"] = Console.Text;

ds.Tables["Консоль"].Rows.Add(СтрокаКонсоли);

//сохранение информации о предметах в инвентаре

dt = new DataTable

{

TableName = "Инвентарь"

};

dt.Columns.Add("Название");

dt.Columns.Add("Числовое\_значение"); //кол-во предметов либо значение восстановления (HP/MP)

ds.Tables.Add(dt);

foreach (Предметы.Зелье Предмет in ПереченьПредметов)

{

DataRow Строка = ds.Tables["Инвентарь"].NewRow();

Строка["Название"] = Предмет.Название;

if (Предмет is Предметы.Зелье\_Лечения)

{

if (Предмет is Предметы.Малое\_Зелье\_Лечения)

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Малое\_Зелье\_Лечения).Количество\_Восполняемого\_Здоровья;

}

else if (Предмет is Предметы.Среднее\_Зелье\_Лечения)

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Среднее\_Зелье\_Лечения).Количество\_Восполняемого\_Здоровья;

}

else

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Большое\_Зелье\_Лечения).Количество\_Восполняемого\_Здоровья;

}

}

else

{

if (Предмет is Предметы.Бутылек\_Маны)

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Бутылек\_Маны).Количество\_Восполняемой\_Маны;

}

else if (Предмет is Предметы.Фласка\_маны)

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Фласка\_маны).Количество\_Восполняемой\_Маны;

}

else

{

Строка["Числовое\_значение"] = (Предмет as Предметы.Банка\_маны).Количество\_Восполняемой\_Маны;

}

}

ds.Tables["Инвентарь"].Rows.Add(Строка);

}

ds.WriteXml("SaveFile.svfl");

MessageBox.Show($"Игра сохранена", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Asterisk);

}

catch (Exception ex)

{

StreamWriter streamWriter = new StreamWriter("ErrorLog.log");

streamWriter.Write(ex);

MessageBox.Show($"Произошла ошибка при сохранении. В директории с игрой был создан лог файл для отладки", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

streamWriter.Close();

}

}

private void Button\_Load\_Click(object sender, RoutedEventArgs e)

{

if (Персонажи.Count == 0)

{

DoLoadFile();

}

else

{

MessageBoxResult DialogResult = MessageBox.Show("Вы уверены что хотите загрузить игру? Текущая игра будет безвозвратно утеряна", "Подтвердить действие?", MessageBoxButton.YesNo, MessageBoxImage.Warning);

if (DialogResult == MessageBoxResult.Yes)

{

DoLoadFile();

}

}

if (File.Exists("SaveFile.svfl"))

{

Button\_Inventory.IsEnabled = true;

MainWindow\_Button\_SwitchHero.IsEnabled = true;

MainWindow\_Button\_GoForAdventure.IsEnabled = true;

}

}

private void DoLoadFile()

{

if (File.Exists("SaveFile.svfl"))

{

Персонажи.Clear();

ПереченьПредметов.Clear();

//загрузка персонажей

DataSet ds = new DataSet();

ds.ReadXml("SaveFile.svfl");

foreach (DataRow строка in ds.Tables["Персонажи"].Rows)

{

if (строка[5] as string == "True")

{

Персонажи.Add(new Персонаж\_с\_магией(

Convert.ToUInt32(строка["ID"]),

Convert.ToString(строка["Имя"]),

Convert.ToBoolean(строка["Пол"]),

Convert.ToUInt32(строка["Возраст"]),

Convert.ToString(строка["Раса"]),

Convert.ToBoolean(строка["Ослаблен"]),

Convert.ToBoolean(строка["Болен"]),

Convert.ToBoolean(строка["Отравлен"]),

Convert.ToBoolean(строка["Парализован"]),

Convert.ToBoolean(строка["Мёртв"]),

Convert.ToBoolean(строка["Может\_говорить"]),

Convert.ToBoolean(строка["Может\_двигаться"]),

Convert.ToUInt32(строка["Максимальное\_здоровье"]),

Convert.ToUInt32(строка["Очки\_Здоровья"]),

Convert.ToUInt32(строка["Очки\_Опыта"]),

Convert.ToUInt32(строка["Максимальная\_мана"]),

Convert.ToUInt32(строка["Очки\_Маны"]),

МножительСилыРун

));

}

else

{

Персонажи.Add(new Персонаж(

Convert.ToUInt32(строка["ID"]),

Convert.ToString(строка["Имя"]),

Convert.ToBoolean(строка["Пол"]),

Convert.ToUInt32(строка["Возраст"]),

Convert.ToString(строка["Раса"]),

Convert.ToBoolean(строка["Ослаблен"]),

Convert.ToBoolean(строка["Болен"]),

Convert.ToBoolean(строка["Отравлен"]),

Convert.ToBoolean(строка["Парализован"]),

Convert.ToBoolean(строка["Мёртв"]),

Convert.ToBoolean(строка["Может\_говорить"]),

Convert.ToBoolean(строка["Может\_двигаться"]),

Convert.ToUInt32(строка["Максимальное\_здоровье"]),

Convert.ToUInt32(строка["Очки\_Здоровья"]),

Convert.ToUInt32(строка["Очки\_Опыта"]),

МножительСилыРун

));

}

}

//загрузка консоли

DataRow СтрокиКонсоли = ds.Tables["Консоль"].Rows[0];

Console.Text = Convert.ToString(СтрокиКонсоли["Консоль\_Текст"]);

ТекущийПерсонаж = Персонажи.Count - 1;

foreach (Персонаж Перс in Персонажи)

{

if (Перс is Персонаж\_с\_магией)

{

Обладает\_магией.Add(true);

}

else

{

Обладает\_магией.Add(false);

}

}

//загрузка предметов

if (ds.Tables["Инвентарь"] != null)

{

foreach (DataRow строка in ds.Tables["Инвентарь"].Rows)

{

if (строка[0] as string == "[+25 ОЗ] Малое зелье лечения")

{

ПереченьПредметов.Add(new Предметы.Малое\_Зелье\_Лечения());

}

else if (строка[0] as string == "[+50 ОЗ] Среднее зелье лечения")

{

ПереченьПредметов.Add(new Предметы.Среднее\_Зелье\_Лечения());

}

else if (строка[0] as string == "[+75 ОЗ] Большое зелье лечения")

{

ПереченьПредметов.Add(new Предметы.Большое\_Зелье\_Лечения());

}

else if (строка[0] as string == "[+25 ОМ] Бутылек Маны")

{

ПереченьПредметов.Add(new Предметы.Бутылек\_Маны());

}

else if (строка[0] as string == "[+50 ОМ] Фласка маны")

{

ПереченьПредметов.Add(new Предметы.Фласка\_маны());

}

else if (строка[0] as string == "[+75 ОМ] Банка маны")

{

ПереченьПредметов.Add(new Предметы.Банка\_маны());

}

}

}

InventoryWindow = new Inventory(Персонажи, ПереченьПредметов);

if (Персонажи.Count == 3)

{

MainWindow\_Button\_AddHero.IsEnabled = false;

}

MainWindow\_Button\_ShowInfo.IsEnabled = true;

Button\_Save.IsEnabled = true;

}

else

{

MessageBox.Show("Файл сохранения не найден. Убедитесь что файл сохранения располагается в директории с игрой", "Ошибка.", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

#endregion

private void Button\_Inventory\_Click(object sender, RoutedEventArgs e)

{

InventoryWindow = new Inventory(Персонажи, ПереченьПредметов);

InventoryWindow.ShowDialog();

ПереченьПредметов = InventoryWindow.GetInventory();

}

private void MainWindow\_Button\_GoForAdventure\_Click(object sender, RoutedEventArgs e)

{

Menu\_Grid.Visibility = Visibility.Hidden;

Adventure\_Grid.Visibility = Visibility.Visible;

СписокГероевВЗащите = new bool[Персонажи.Count];

SetRuneRank(1);

adventureScripts = new AdventureScripts(NextAdventureIs, this);

}

public string Get\_Button\_Choice\_1()

{

return (string)Button\_Choice\_1.Content;

}

public void Change\_Button\_Choice\_1(string Text)

{

Button\_Choice\_1.Content = Text;

}

public string Get\_Button\_Choice\_2()

{

return (string)Button\_Choice\_2.Content;

}

public void Change\_Button\_Choice\_2(string Text)

{

Button\_Choice\_2.Content = Text;

}

public string Get\_Button\_Choice\_3()

{

return (string)Button\_Choice\_3.Content;

}

public void Change\_Button\_Choice\_3(string Text)

{

Button\_Choice\_3.Content = Text;

}

private void Button\_Choice\_1\_Click(object sender, RoutedEventArgs e)

{

if (Button\_Choice\_1.Content.ToString() == "[Закрыть \nигру]")

{

Application.Current.Shutdown();

}

else if (!IsBattleMode)

{

adventureScripts.Choices\_Add(1);

ConsoleWriteLine(1);

adventureScripts.Воспроизведение\_Шагов();

}

else

{

int OrderOf = adventureScripts.OrderOf;

AttackDialog = new Attack(СписокТекущихВрагов, Персонажи[OrderOf]);

AttackDialog.ShowDialog();

if (AttackDialog.DialogResult == true)

{

СписокТекущихВрагов = AttackDialog.GetСписокТекущихВрагов();

if (AttackDialog.IsEnemyKilled == false)

{

if (Персонажи[OrderOf].Пол == true)

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} нанёс {AttackDialog.ЗначениеАтаки} единиц урона здоровью цели: " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага}. Текущие значение здоровья врага стало " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ЗдоровьеВрага} единиц здоровья.");

}

else

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} нанесла {AttackDialog.ЗначениеАтаки} единиц урона здоровью цели: " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага}. Текущие значение здоровья врага стало " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ЗдоровьеВрага} единиц здоровья.");

}

}

else

{

if (Персонажи[OrderOf].Пол == true)

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} нанёс {AttackDialog.ЗначениеАтаки} единиц урона здоровью цели: " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага}. {СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага} убит.");

}

else

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} нанесла {AttackDialog.ЗначениеАтаки} единиц урона здоровью цели: " +

$"{СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага}. {СписокТекущихВрагов[AttackDialog.SelectedEnemy].ИмяВрага} убит.");

}

}

Random random = new Random();

Персонажи[OrderOf].Добавить\_ОчкиОпыта((uint)random.Next(100, 150));

adventureScripts.IncreaseOrderOf();

}

adventureScripts.Battle(СписокТекущихВрагов, ПорядокАтаки, СписокГероевВЗащите);

}

}

private void Button\_Choice\_2\_Click(object sender, RoutedEventArgs e)

{

if (!IsBattleMode)

{

adventureScripts.Choices\_Add(2);

ConsoleWriteLine(2);

adventureScripts.Воспроизведение\_Шагов();

}

else

{

int OrderOf = adventureScripts.OrderOf;

Magic magic = new Magic(Персонажи, СписокТекущихВрагов, ПорядокАтаки, OrderOf, МножительСилыРун);

magic.ShowDialog();

if (magic.DialogResult == true)

{

ConsoleWriteLine(magic.ConsoleOutput);

adventureScripts.IncreaseOrderOf();

adventureScripts.Battle(СписокТекущихВрагов, ПорядокАтаки, СписокГероевВЗащите);

}

}

}

private void Button\_Choice\_3\_Click(object sender, RoutedEventArgs e)

{

if (!IsBattleMode)

{

adventureScripts.Choices\_Add(3);

ConsoleWriteLine(3);

adventureScripts.Воспроизведение\_Шагов();

}

else

{

int OrderOf = adventureScripts.OrderOf;

СписокГероевВЗащите[OrderOf] = true;

if (Персонажи[OrderOf].Пол == true)

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} принял защитную стойку. Входящий урон снижен в 2 раза.");

}

else

{

ConsoleWriteLine($"CОПИ: {Персонажи[OrderOf].Имя} приняла защитную стойку. Входящий урон снижен в 2 раза.");

}

adventureScripts.IncreaseOrderOf();

adventureScripts.Battle(СписокТекущихВрагов, ПорядокАтаки, СписокГероевВЗащите);

}

}

public void Block\_Button\_Choice\_1()

{

Button\_Choice\_1.Visibility = Visibility.Hidden;

}

public void UnBlock\_Button\_Choice\_1()

{

Button\_Choice\_1.Visibility = Visibility.Visible;

}

public void Block\_Button\_Choice\_2()

{

Button\_Choice\_2.Visibility = Visibility.Hidden;

}

public void UnBlock\_Button\_Choice\_2()

{

Button\_Choice\_2.Visibility = Visibility.Visible;

}

public void Block\_Button\_Choice\_3()

{

Button\_Choice\_3.Visibility = Visibility.Hidden;

}

public void UnBlock\_Button\_Choice\_3()

{

Button\_Choice\_3.Visibility = Visibility.Visible;

}

public void ConsoleWriteLine(string Text)

{

Console.Text += '\n';

Console.Text += '\n';

Console.Text += Text;

ScrollViewer.ScrollToEnd();

}

public void ConsoleWriteLine(int НомерВыбора)

{

Console.Text += '\n';

Console.Text += '\n';

switch (НомерВыбора)

{

case 1:

{

string temp = (string)Button\_Choice\_1.Content;

temp = temp.Replace('\n', ' ');

Console.Text += $" - {temp}";

}

break;

case 2:

{

string temp = (string)Button\_Choice\_2.Content;

temp = temp.Replace('\n', ' ');

Console.Text += $" - {temp}";

}

break;

case 3:

{

string temp = (string)Button\_Choice\_3.Content;

temp = temp.Replace('\n', ' ');

Console.Text += $" - {temp}";

}

break;

}

ScrollViewer.ScrollToEnd();

}

private void Button\_GiveUp\_Click(object sender, RoutedEventArgs e)

{

IsBattleMode = false;

adventureScripts.Choices\_Add(1);

adventureScripts.Choices\_Add(1);

if (Персонажи.Count == 1 && Персонажи[0].Пол == true)

{

ConsoleWriteLine("- Стража! У нас предатель и трус! Убить героя!");

}

else if (Персонажи.Count == 1 && Персонажи[0].Пол == false)

{

ConsoleWriteLine("- Стража! У нас предательница и трусиха! Убить героиню!");

}

else

{

ConsoleWriteLine("- Стража! У нас предатели и трусы! Убить героев!");

}

foreach (Персонаж персонаж in Персонажи)

{

персонаж.ИзменениеСостоянияЗдоровья("урон", 1000);

}

adventureScripts.GameOver();

}

public void SetRuneRank(int i)

{

switch (i)

{

case 1:

{

МножительСилыРун[0] = 1;

МножительСилыРун[1] = 1.2;

МножительСилыРун[2] = 1;

foreach (Персонаж персонаж in Персонажи)

{

персонаж.RuneUpdate(МножительСилыРун);

}

}

break;

case 2:

{

МножительСилыРун[0] = 2;

МножительСилыРун[1] = 1;

МножительСилыРун[2] = 1.5;

foreach (Персонаж персонаж in Персонажи)

{

персонаж.RuneUpdate(МножительСилыРун);

}

}

break;

case 3:

{

МножительСилыРун[0] = 3;

МножительСилыРун[1] = 0.75;

МножительСилыРун[2] = 2;

foreach (Персонаж персонаж in Персонажи)

{

персонаж.RuneUpdate(МножительСилыРун);

}

}

break;

default:

{

SetRuneRank(1);

}

break;

}

}

}

}

Враги.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Ролевая\_игра\_\_WPF\_

{

public abstract class Враги

{

public abstract string ИмяВрага { get; protected set; }

public abstract int ЗдоровьеВрага { get; protected set; }

public abstract int СилаАтаки { get; protected set; }

public abstract int МаксимальноеЗдоровье { get; }

public abstract bool IsFrozen { get; set; }

public void ОтнятьЗдоровье(int СилаАтакиПерсонажа) => ЗдоровьеВрага -= СилаАтакиПерсонажа;

public class ГлаваСтражи : Враги

{

public override string ИмяВрага { get; protected set; } = "Глава стражи";

public override int ЗдоровьеВрага { get; protected set; } = 500;

public override int СилаАтаки { get; protected set; } = 10;

public override int МаксимальноеЗдоровье { get; } = 500;

public override bool IsFrozen { get; set; } = false;

}

}

}

Заклинания.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Ролевая\_игра\_\_WPF\_

{

internal class Заклинания

{

public abstract class Заклинание

{

public virtual string Название { get; }

public virtual uint Стоимость { get; }

}

public class Лечение : Заклинание

{

new public static string Название { get; } = "Заклинание Лечения";

public override uint Стоимость { get; } = 15;

public uint Сила\_Воздействия { get; } = 25;

public Лечение(Персонаж\_с\_магией Источник, Персонаж Цель, double Множитель\_Стоимости)

{

Источник.ИзменениеСостоянияМаны("расходование", (uint)(Стоимость \* Множитель\_Стоимости));

Цель.ИзменениеСостоянияЗдоровья("лечение", Сила\_Воздействия);

}

}

public class Воскрешение : Заклинание

{

new public static string Название { get; } = "Заклинание Воскрешения";

public override uint Стоимость { get; } = 100;

public Воскрешение(Персонаж\_с\_магией Источник, Персонаж Цель, double Множитель\_Стоимости)

{

Источник.ИзменениеСостоянияМаны("расходование", (uint)(Стоимость \* Множитель\_Стоимости));

Цель.ИзменениеСостоянияЗдоровья("лечение", 50);

}

}

public class Огненный\_Шар : Заклинание

{

new public static string Название { get; } = "Заклинание Огненного Шара";

public override uint Стоимость { get; } = 40;

public Огненный\_Шар(Персонаж\_с\_магией Источник, Враги Цель, double Множитель\_Стоимости)

{

Источник.ИзменениеСостоянияМаны("расходование", (uint)(Стоимость \* Множитель\_Стоимости));

Цель.ОтнятьЗдоровье(50);

}

}

public class Заморозка : Заклинание

{

new public static string Название { get; } = "Заклинание Заморозки";

public override uint Стоимость { get; } = 20;

public Заморозка(Персонаж\_с\_магией Источник, Враги Цель, double Множитель\_Стоимости)

{

Источник.ИзменениеСостоянияМаны("расходовние", (uint)(Стоимость \* Множитель\_Стоимости));

Цель.IsFrozen = true;

}

}

}

}

Персонаж.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Ролевая\_игра\_\_WPF\_

{

public class Персонаж

{

public uint ID { get; protected set; }

public string Имя { get; protected set; }

public string Раса { get; protected set; }

public uint Возраст { get; protected set; }

/// <summary>

/// Если true, то мужской, если false, то женский

/// </summary>

public bool Пол { get; protected set; }

/// <summary>

/// ослаблен, болен, отравлен, парализован, мёртв

/// </summary>

public bool[] Состояние = new bool[5] { false, false, false, false, false };

public bool Может\_говорить { get; protected set; } = true;

public bool Может\_двигаться { get; protected set; } = true;

public uint Максимальное\_здоровье\_Начальное { get; protected set; }

public uint Максимальное\_здоровье { get; protected set; }

public uint Очки\_Здоровья { get; protected set; }

public uint Очки\_Опыта { get; protected set; } = 0;

public double[] МножительСилыРун;

/// <summary>

/// Создание нового персонажа

/// </summary>

/// <param name="ID">Уникальный номер (автоподсчет)</param>

/// <param name="Имя">Имя героя</param>

/// <param name="Пол">Пол героя</param>

/// <param name="Возраст">Возраст героя</param>

/// <param name="Раса">Раса героя</param>

public Персонаж(uint ID, string Имя, bool Пол, uint Возраст, string Раса, double[] МножительСилыРун)

{

this.ID = ID;

this.Имя = Имя;

this.Пол = Пол;

this.Возраст = Возраст;

this.Раса = Раса;

this.МножительСилыРун = МножительСилыРун;

Установить\_константы();

}

public Персонаж(uint ID, string Имя, bool Пол, uint Возраст, string Раса, bool Ослаблен, bool Болен, bool Отравлен, bool Парализован, bool Мертв, bool Может\_говорить, bool Может\_двигаться, uint Максимальное\_здоровье, uint Очки\_Здоровья, uint Очки\_Опыта, double[] МножительСилыРун)

{

this.ID = ID;

this.Имя = Имя;

this.Пол = Пол;

this.Возраст = Возраст;

this.Раса = Раса;

bool[] Сжатые\_состояния = { Ослаблен, Болен, Отравлен, Парализован, Мертв };

Состояние = Сжатые\_состояния;

this.Может\_говорить = Может\_говорить;

this.Может\_двигаться = Может\_двигаться;

this.Максимальное\_здоровье = Максимальное\_здоровье;

this.Очки\_Здоровья = Очки\_Здоровья;

this.Очки\_Опыта = Очки\_Опыта;

this.МножительСилыРун = МножительСилыРун;

}

protected virtual void Установить\_константы()

{

if (Раса == "человек")

{

Максимальное\_здоровье = (uint)(100 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 100;

Очки\_Здоровья = Максимальное\_здоровье;

}

if (Раса == "гном")

{

Максимальное\_здоровье = (uint)(150 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 150;

Очки\_Здоровья = Максимальное\_здоровье;

}

if (Раса == "эльф" || Раса == "эльфийка")

{

Максимальное\_здоровье = (uint)(130 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 130;

Очки\_Здоровья = Максимальное\_здоровье;

}

if (Раса == "орк")

{

Максимальное\_здоровье = (uint)(200 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 200;

Очки\_Здоровья = Максимальное\_здоровье;

}

if (Раса == "гоблин")

{

Максимальное\_здоровье = (uint)(120 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 120;

Очки\_Здоровья = Максимальное\_здоровье;

}

}

/// <param name="value">Сколько опыта добавить?</param>

public void Добавить\_ОчкиОпыта(uint value)

{

Очки\_Опыта += value;

}

public override string ToString()

{

string Информация = "";

Информация += $"\n - - Информация о состоянии - -\n";

Дебаффы(Состояние);

if (Может\_говорить)

{

Информация += $"{Имя} может говорить\n";

}

else

{

Информация += $"{Имя} говорить не может\n";

}

if (Может\_двигаться)

{

Информация += $"{Имя} может двигаться навстречу приключениям\n";

}

else

{

Информация += $"{Имя}, к сожалению, навстречу приключениям двигаться не может\n";

}

Информация += $"\n - - Информация о характеристиках - -\n";

Информация += $"Текущее здоровье: {Очки\_Здоровья} HP\n";

Информация += $"Максимальное здоровье: {Максимальное\_здоровье} HP\n";

Информация += $"Текущее количество опыта: {Очки\_Опыта} EXP\n";

Информация += $"\n - - Информация об отрицательных эффектах - -\n";

Дебаффы(Состояние);

return Информация;

}

public static string Лет\_Лета\_Год(uint Возраст)

{

if (Возраст == 1 || Возраст == 21 || Возраст == 31 || Возраст == 41 || Возраст == 51 || Возраст == 61 || Возраст == 71 || Возраст == 81 || Возраст == 91)

{

return Возраст + " год";

}

else if ((Возраст >= 2 && Возраст <= 4) || (Возраст >= 22 && Возраст <= 24) || (Возраст >= 32 && Возраст <= 34) || (Возраст >= 42 && Возраст <= 44) || (Возраст >= 52 && Возраст <= 54) || (Возраст >= 62 && Возраст <= 64) || (Возраст >= 72 && Возраст <= 74) || (Возраст >= 82 && Возраст <= 84) || (Возраст >= 92 && Возраст <= 94))

{

return Возраст + " года";

}

else if ((Возраст >= 5 && Возраст <= 20) || (Возраст >= 25 && Возраст <= 30) || (Возраст >= 35 && Возраст <= 40) || (Возраст >= 45 && Возраст <= 50) || (Возраст >= 55 && Возраст <= 60) || (Возраст >= 65 && Возраст <= 70) || (Возраст >= 75 && Возраст <= 80) || (Возраст >= 85 && Возраст <= 90) || (Возраст >= 95 && Возраст <= 100))

{

return Возраст + " лет";

}

return "";

}

protected string Дебаффы(bool[] Состояние)

{

string Информация = "";

if (Состояние[0] == false && Состояние[1] == false && Состояние[2] == false && Состояние[3] == false && Состояние[4] == false)

{

Информация += $"{Имя} не имеет отрицательных эффектов\n";

}

if (Состояние[4] == true)

{

if (Пол)

{

Информация += $"Будут сложены и увековечены легенды о подвигах великого героя {Имя}. (Герой мертв)\n";

}

else

{

Информация += $"Будут сложены и увековечены легенды о подвигах великой героини {Имя}. (Героиня мертва)\n";

}

}

else

{

if (Состояние[0] == true)

{

if (Пол)

{

Информация += $"{Имя} ослаблен\n";

}

else

{

Информация += $"{Имя} ослаблена\n";

}

}

if (Состояние[1] == true)

{

if (Пол)

{

Информация += $"{Имя} болен\n";

}

else

{

Информация += $"{Имя} больна\n";

}

}

if (Состояние[2] == true)

{

if (Пол)

{

Информация += $"{Имя} отравлен\n";

}

else

{

Информация += $"{Имя} отравлена\n";

}

}

}

return Информация;

}

/// <summary>

/// Изменение состояния текущего персонажа

/// </summary>

/// <param name="действие">Принимает: "урон", "лечение"</param>

/// <param name="сила\_воздействия">Принимает: uint</param>

public void ИзменениеСостоянияЗдоровья(string действие, uint сила\_воздействия)

{

if (действие == "урон")

{

if (Очки\_Здоровья <= сила\_воздействия)

{

Очки\_Здоровья = 0;

}

else

{

Очки\_Здоровья -= сила\_воздействия;

}

}

else if (действие == "лечение")

{

if (сила\_воздействия + Очки\_Здоровья > Максимальное\_здоровье)

{

Очки\_Здоровья = Максимальное\_здоровье;

}

else

{

Очки\_Здоровья += сила\_воздействия;

}

}

ИзменениеСостоянияЭффектов();

}

private void ИзменениеСостоянияЭффектов()

{

int ДесятьПроцентовОтHP = (int)Максимальное\_здоровье / 10;

//если здоровье персонажа становится <10%, персонаж переходит из состояния «здоров» в состояние «ослаблен».

if (Очки\_Здоровья <= ДесятьПроцентовОтHP)

{

Состояние[0] = true; //ослаблен

}

if (Очки\_Здоровья >= ДесятьПроцентовОтHP)

{

Состояние[0] = false; //ослаблен

}

if (Очки\_Здоровья == 0)

{

Состояние[0] = false; //ослаблен

Состояние[1] = false; //болен

Состояние[2] = false; //отравлен

Состояние[3] = false; //парализован

Состояние[4] = true; //мёртв

}

}

public int ПолучитьЗначениеАтаки()

{

Random random = new Random();

return (int)Math.Round(random.Next(20, 40) \* МножительСилыРун[2], 0);

}

public void RuneUpdate(double[] НаследованныеМножителиСилыРун)

{

МножительСилыРун = НаследованныеМножителиСилыРун;

}

}

public class Персонаж\_с\_магией : Персонаж

{

public uint Максимальная\_мана { get; protected set; }

public uint Очки\_Маны { get; protected set; }

public Персонаж\_с\_магией(uint ID, string Имя, bool Пол, uint Возраст, string Раса, double[] МножительСилыРун) : base(ID, Имя, Пол, Возраст, Раса, МножительСилыРун)

{

this.ID = ID;

this.Имя = Имя;

this.Пол = Пол;

this.Возраст = Возраст;

this.Раса = Раса;

Установить\_константы();

}

public Персонаж\_с\_магией(uint ID, string Имя, bool Пол, uint Возраст, string Раса, bool Ослаблен, bool Болен, bool Отравлен, bool Парализован, bool Мертв, bool Может\_говорить, bool Может\_двигаться, uint Максимальное\_здоровье, uint Очки\_Здоровья, uint Очки\_Опыта, uint Максимальная\_мана, uint Очки\_Маны, double[] МножительСилыРун) : base(ID, Имя, Пол, Возраст, Раса, Ослаблен, Болен, Отравлен, Парализован, Мертв, Может\_говорить, Может\_двигаться, Максимальное\_здоровье, Очки\_Здоровья, Очки\_Опыта, МножительСилыРун)

{

this.ID = ID;

this.Имя = Имя;

this.Пол = Пол;

this.Возраст = Возраст;

this.Раса = Раса;

bool[] Сжатые\_состояния = { Ослаблен, Болен, Отравлен, Парализован, Мертв };

Состояние = Сжатые\_состояния;

this.Может\_говорить = Может\_говорить;

this.Может\_двигаться = Может\_двигаться;

this.Максимальное\_здоровье = Максимальное\_здоровье;

this.Очки\_Здоровья = Очки\_Здоровья;

this.Очки\_Опыта = Очки\_Опыта;

this.Максимальная\_мана = Максимальная\_мана;

this.Очки\_Маны = Очки\_Маны;

}

protected override void Установить\_константы()

{

if (Раса == "человек")

{

Максимальное\_здоровье = (uint)(100 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 100;

Очки\_Здоровья = Максимальное\_здоровье;

Максимальная\_мана = 100;

Очки\_Маны = Максимальная\_мана;

}

if (Раса == "гном")

{

Максимальное\_здоровье = (uint)(150 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 150;

Очки\_Здоровья = Максимальное\_здоровье;

Максимальная\_мана = 50;

Очки\_Маны = Максимальная\_мана;

}

if (Раса == "эльф" || Раса == "эльфийка")

{

Максимальное\_здоровье = (uint)(130 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 130;

Очки\_Здоровья = Максимальное\_здоровье;

Максимальная\_мана = 150;

Очки\_Маны = Максимальная\_мана;

}

if (Раса == "орк")

{

Максимальное\_здоровье = (uint)(200 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 200;

Очки\_Здоровья = Максимальное\_здоровье;

Максимальная\_мана = 50;

Очки\_Маны = Максимальная\_мана;

}

if (Раса == "гоблин")

{

Максимальное\_здоровье = (uint)(120 \* МножительСилыРун[0]);

Максимальное\_здоровье\_Начальное = 120;

Очки\_Здоровья = Максимальное\_здоровье;

Максимальная\_мана = 130;

Очки\_Маны = Максимальная\_мана;

}

}

public override string ToString()

{

string Информация = "";

Информация += $"\n - - Информация о состоянии - -\n";

Информация += Дебаффы(Состояние);

if (Может\_говорить)

{

Информация += $"{Имя} может говорить\n";

}

else

{

Информация += $"{Имя} говорить не может\n";

}

if (Может\_двигаться)

{

Информация += $"{Имя} может двигаться навстречу приключениям\n";

}

else

{

Информация += $"{Имя}, к сожалению, навстречу приключениям двигаться не может\n";

}

Информация += $"\n - - Информация о характеристиках - -\n";

Информация += $"Текущее здоровье: {Очки\_Здоровья} HP\n";

Информация += $"Максимальное здоровье: {Максимальное\_здоровье} HP\n";

Информация += $"Текущее количество маны: {Очки\_Маны} MP\n";

Информация += $"Максимальное количество маны: {Максимальная\_мана} MP\n";

Информация += $"Текущее количество опыта: {Очки\_Опыта} EXP\n";

return Информация;

}

/// <summary>

/// Изменение состояния маны

/// </summary>

/// <param name="действие">Принимает: "расходование", "восполнение"</param>

/// <param name="сила\_воздействия">Принимает: uint</param>

public void ИзменениеСостоянияМаны(string действие, uint сила\_воздействия)

{

if (действие == "расходование")

{

Очки\_Маны -= сила\_воздействия;

}

if (действие == "восполнение")

{

Очки\_Маны += сила\_воздействия;

if (Очки\_Маны > Максимальная\_мана)

{

Очки\_Маны = Максимальная\_мана;

}

}

}

}

}

Предметы.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Ролевая\_игра\_\_WPF\_

{

public class Предметы

{

public abstract class Зелье

{

public virtual string Название { get; }

}

public abstract class Зелье\_Лечения : Зелье

{

public virtual uint Количество\_Восполняемого\_Здоровья { get; }

}

public class Малое\_Зелье\_Лечения : Зелье\_Лечения

{

/// <summary>

/// [+25 ОЗ] Малое зелье лечения

/// </summary>

public Малое\_Зелье\_Лечения() { }

public override string Название { get; } = "[+25 ОЗ] Малое зелье лечения";

public override uint Количество\_Восполняемого\_Здоровья { get; } = 25;

}

/// <summary>

/// [+50 ОЗ] Среднее зелье лечения

/// </summary>

public class Среднее\_Зелье\_Лечения : Зелье\_Лечения

{

/// <summary>

/// [+50 ОЗ] Среднее зелье лечения

/// </summary>

public Среднее\_Зелье\_Лечения(){}

public override string Название { get; } = "[+50 ОЗ] Среднее зелье лечения";

public override uint Количество\_Восполняемого\_Здоровья { get; } = 50;

}

/// <summary>

/// [+75 ОЗ] Большое зелье лечения

/// </summary>

public class Большое\_Зелье\_Лечения : Зелье\_Лечения

{

/// <summary>

/// [+75 ОЗ] Большое зелье лечения

/// </summary>

public Большое\_Зелье\_Лечения(){}

public override string Название { get; } = "[+75 ОЗ] Большое зелье лечения";

public override uint Количество\_Восполняемого\_Здоровья { get; } = 75;

}

public class Зелье\_маны : Зелье

{

public virtual uint Количество\_Восполняемой\_Маны { get; }

}

/// <summary>

/// [+25 ОМ] Бутылек Маны

/// </summary>

public class Бутылек\_Маны : Зелье\_маны

{

/// <summary>

/// [+25 ОМ] Бутылек Маны

/// </summary>

public Бутылек\_Маны(){}

public override string Название { get; } = "[+25 ОМ] Бутылек Маны";

public override uint Количество\_Восполняемой\_Маны { get; } = 25;

}

/// <summary>

/// [+50 ОМ] Фласка маны

/// </summary>

public class Фласка\_маны : Зелье\_маны

{

/// <summary>

/// [+50 ОМ] Фласка маны

/// </summary>

public Фласка\_маны(){}

public override string Название { get; } = "[+50 ОМ] Фласка маны";

public override uint Количество\_Восполняемой\_Маны { get; } = 50;

}

/// <summary>

/// [+75 ОМ] Банка маны

/// </summary>

public class Банка\_маны : Зелье\_маны

{

/// <summary>

/// [+75 ОМ] Банка маны

/// </summary>

public Банка\_маны(){}

public override string Название { get; } = "[+75 ОМ] Банка маны";

public override uint Количество\_Восполняемой\_Маны { get; } = 75;

}

}

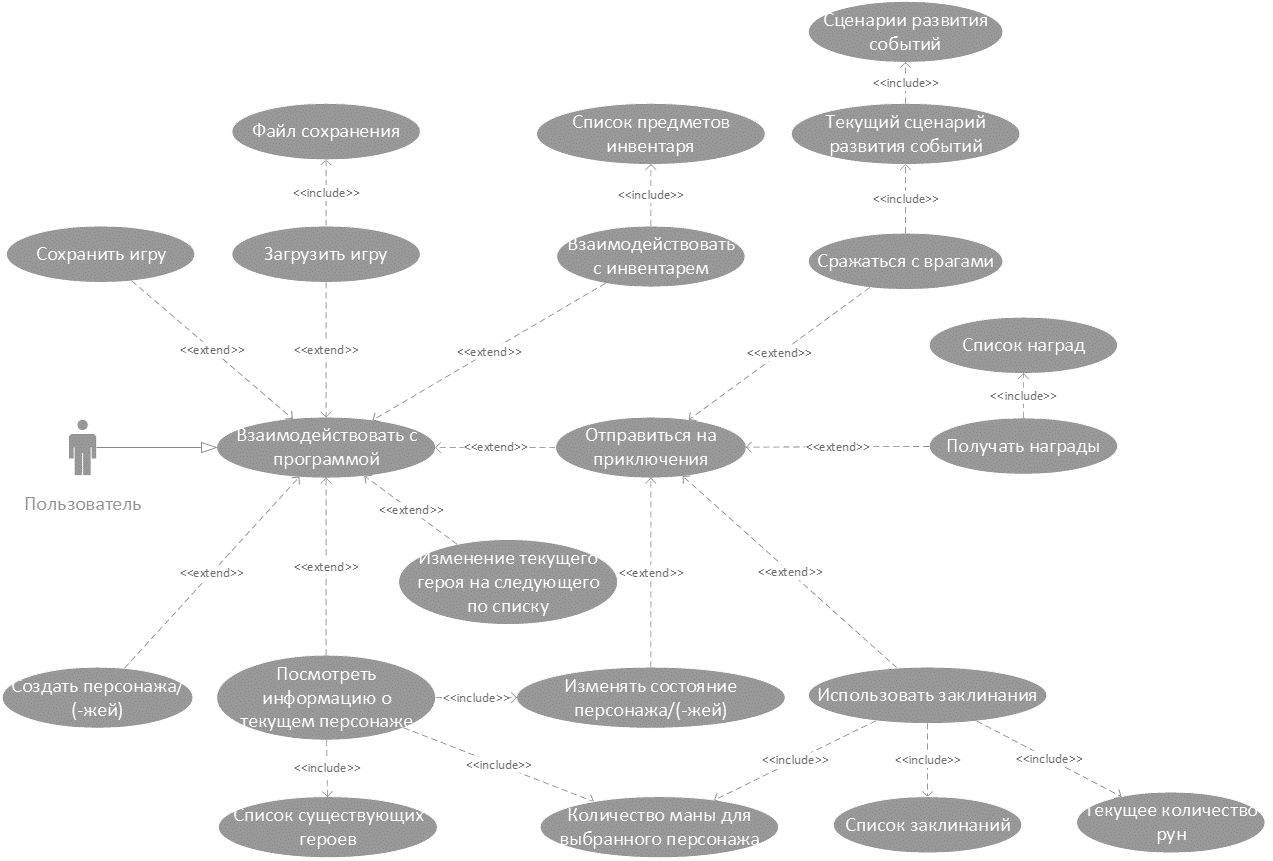
}



Масса

Лит.

Масштаб



Листов 5

Н.В.Ржеутская

Лист 1

Реценз.

Т. Контр.

*Ролевая игра «Герои подземелья»*

Диаграмма вариантов использования

Утверд.

Н. Контр.

К.В.Максимук

Разраб.

Провер.

*ОП Т.693011.401 ГЧ*

Инв.№подл.

КБиП

Взам.инв.№

Подп. и дата

Инв.№дубл.

Подп. и дата

У