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](#_Toc7510353)

[2 Объектно-ориентированный анализ и проектирование системы 3](#_Toc7510354)

[2.1 Сущность задачи 4](#_Toc7510355)

[2.2 Проектирование модели 5](#_Toc7510356)

[3 Вычислительная система 8](#_Toc7510357)

[3.1 Требования к аппаратным и операционным ресурсам 8](#_Toc7510358)

[3.2 Инструменты разработки 8](#_Toc7510359)

[4 Проектирование задачи 10](#_Toc7510360)

[4.1 Требование к приложению 10](#_Toc7510361)

[4.2 Концептуальный прототип 11](#_Toc7510362)

[4.3 Организация данных 12](#_Toc7510363)

[4.5 Проектирование справочной системы приложения 12](#_Toc7510364)

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.

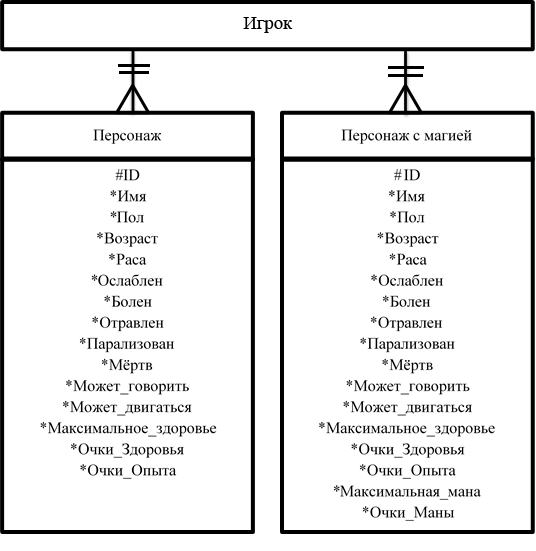


Рисунок 2.1- Диаграмма «сущность-связь» в нотации Баркера

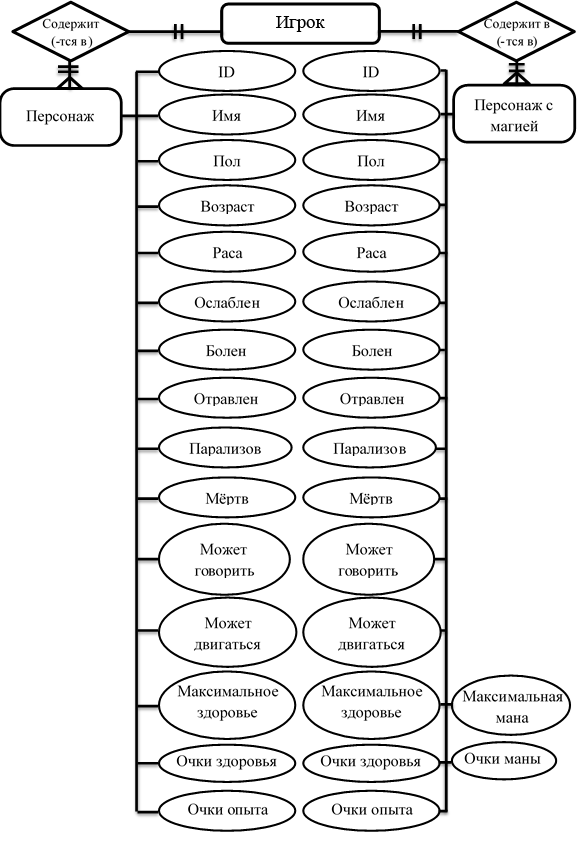


Рисунок 2.2 - Диаграмма ERD в нотации Ричарда Чена

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 базы данных «Medzdor.mdf»;
* файл журнала транзакций «Medzdor \_log.ldf»;
* файл программы «Monzdr.exe»
* файл отчетов «\*.rdlc»;
* файл форм «\*.resx»
* файл форм «\*.cs»;
* файл справочной системы «help.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 100 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более;
* монитор;
* мышь, клавиатура;
* принтер.

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

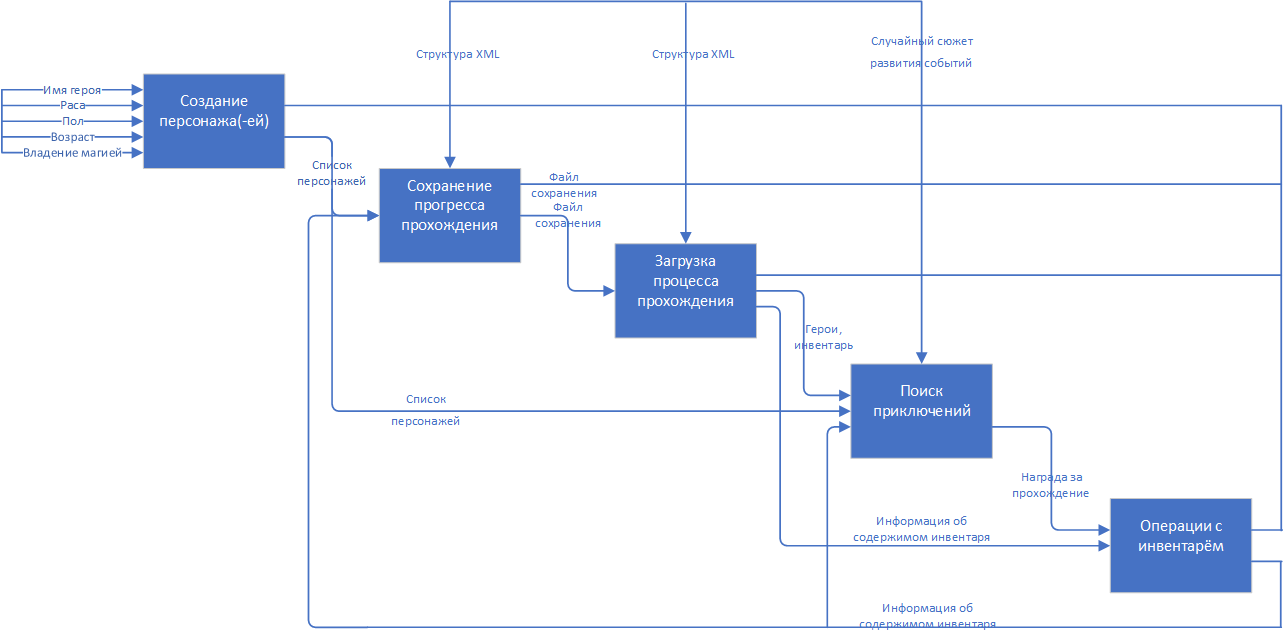
Далее блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена 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 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 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 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.

Список используемых источников

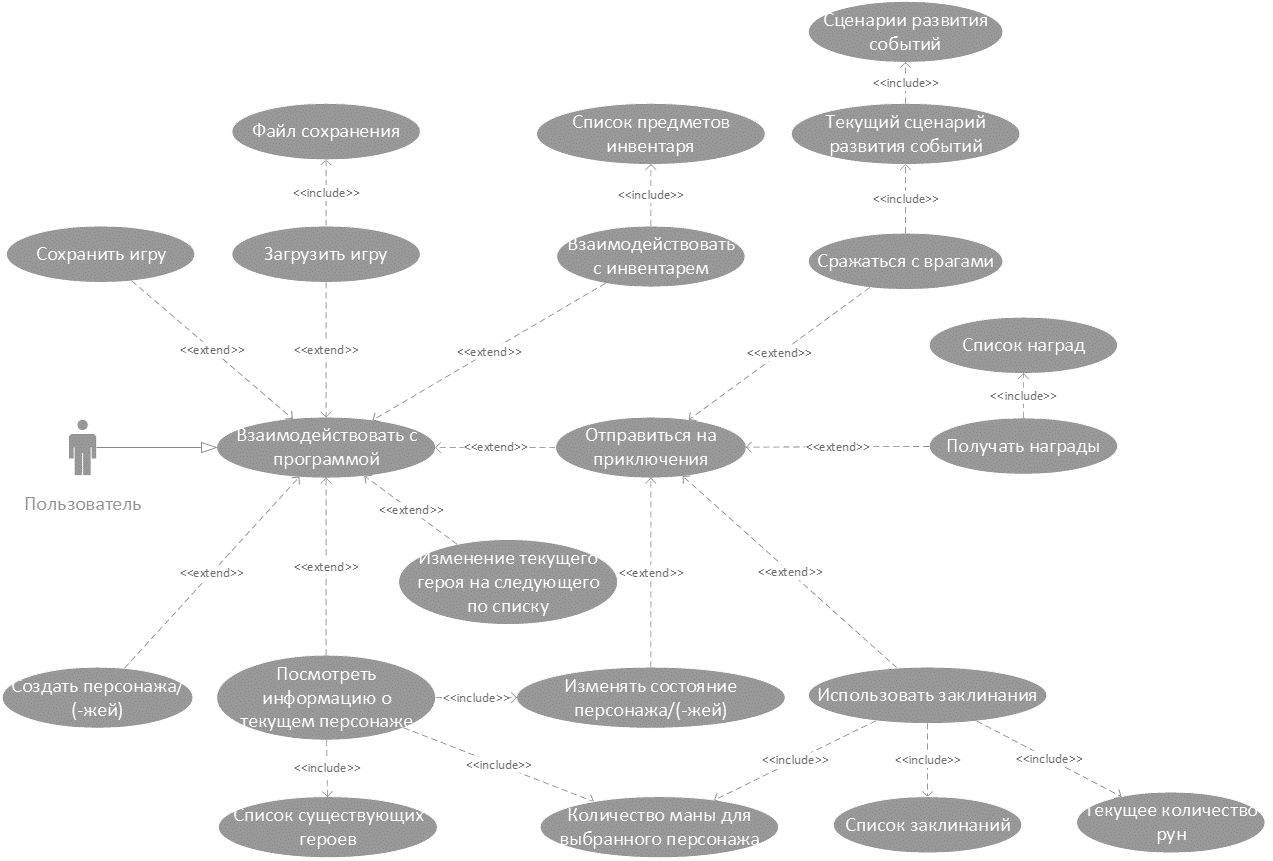
1. Грабер, М. SQL. Описание SQL92, SQL99 и SQLJ/ Грабер М.: Лори, 2003 г.
2. Дейт, К. Дж. Введение в системы баз данных/ Дейт К.Дж.: Вильямс, 2006 г.
3. Мартин, К.Р. Чистый Код. Создание анализ и рефакторинг. Библиотека программистов./ К.Р. Мартин.: Питер, 2010 г.
4. Орлов, С.А. Технология разработки 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.



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