**Содержание**

Введение4

1 Анализ задачи 6

1.1 Постановка задачи6

1.1.1 Организационно-экономическая сущность задачи6

1.1.2 Функциональные требования6

1.1.3 Описание входной, выходной и условно-постоянной информации7

1.1.4 Нефункциональные требования8

1.2 Диаграмма вариантов использования 8

1.3 Разработка плана работы над проектом10

1.4 Выбор стратегии разработки и модели жизненного цикла10

1.5 Инструменты разработки14

2 Проектирование задачи15

2.1 Разработка системы меню15

2.2 разработка UML-диаграмм15

2.2.1 Диаграмма последовательности15

2.2.2 Диаграмма деятельности15

2.2.3 Функциональная модель15

2.2.4 Разработка пользовательского интерфейса15

3 Реализация17

3.1 Руководство программиста17

3.1.1 Организация данных17

3.1.2 Структура программы17

3.1.3Спецификация программы19

4Тестирование21

4.1 Тесты на использование21

4.2 Отчёт о результатах тестирования21

5 Руководство пользователя22

5.1 Руководство системного программиста22

5.2 Запуск программы22

5.3 Инструкция по работе с программой23

5.4 Завершение работы с программой27

Заключение28

Список использованных источников29

Приложение А30

Приложение Б32

Приложение В34

Приложение Г36

Приложение Д38

Приложение Ж40

Приложение И43

Приложение К46

**Введение**

На практику была поставлена задача, разработать программу на тему: «Игровое приложение о криптографии».

Цель практики заключается в знакомстве массового пользователя с возможными способами шифрования сообщений, их стойкостью и преимуществами.

Создаваемое приложение будет рассчитано для любого рода пользователей. Применить данную программу смогут не только люди, имеющее какое-то представление о криптографии, но и просто люди, заинтересовавшиеся в данной теме.

Далее приведём краткое описание разделов пояснительной записки.

Первый раздел носит название «Анализ задачи». В нем вы сможете ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, определение её организационно-экономической сущности. Также в этом разделе вы сможете узнать о том, как данная задача решается в настоящее время. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе «Инструменты разработки» будет рассмотрена среда, в которой создаётся данный проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачей.

В разделе «Проектирование задачи» будут рассмотрены основные аспекты разработки программного продукта. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет чётко описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации, описана разработка системы справочной информации.

«Реализация задачи» – это третий раздел пояснительной записки, в котором описываются все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе будут чётко описаны функции пользователя и их структура. Здесь можно будет найти таблицу, в которой будет представлена полная аннотация файлов, используемых в данном проекте.

Четвёртый раздел – «Тестирование». В нем будет описано полное и функциональное тестирование данной программы, т.е. будет оттестирован каждый пункт меню, каждая операция, которая выполняется приложением. Будут смоделированы все возможные действия пользователя при работе с программой, начиная от запуска до выхода.

В разделе «Руководство пользователя» будет описано назначение, область применения, среда функционирования программы.

«Заключение» будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств.

В «Список использованных источнике» будет приведён список используемых при разработке источников.

В приложениях к пояснительной записке будет приведён листинг программы с необходимыми комментариями.

Схема работы системы будет представлена в графической части.

В приложении А предоставлена диаграмма вариантов использования.

В приложении Б предоставлена диаграмма Ганта.

В приложении В предоставлена структура игрового приложения.

В приложении Г предоставлена диаграмма последовательности.

В приложении Д предоставлена диаграмма классов.

В приложении Ж представлены UX прототипы.

В приложении И представлен отчёт о тестировании.

В приложении К представлен листинг.

**1 Анализ задачи**

**1.1 Постановка задачи**

**1.1.1 Организационно-экономическая сущность задачи**

Наименование задачи: видеоигра «Crypto Dungeon».

Цель разработки: создание видеоигры для обучения основам криптографии.

Назначение:данный продукт разрабатывается для людей любого возраста, желающих изучить основы криптографии.

Периодичность использования ПП: по желанию пользователя.

Источники и способы получения данных:

учебные пособия:

1. Книга шифров Саймон Сингх;

интернет статьи:

1. <https://ru.wikipedia.org/wiki/Шифр_Виженера>;
2. <https://ru.wikipedia.org/wiki/Шифр_Цезаря>;
3. https://ru.wikipedia.org/wiki/AES\_(стандарт\_шифрования).

Информационная связь с другими задачами: данный программный продукт связан с наукой криптографией.

Обзор существующих аналогичных ПП:

В ходе анализа был обнаружен продукт из данной предметной области. Decipherism является хорошей игрой из данной области, однако от нашего продукта очень сильно отличается, так как выполнен в 2д, а также имеет гораздо меньший функционал, являясь только иллюстратором работы шифров, а также маленьким задачником.

**1.1.2 Функциональные требования**

Жанр данной игры – инди-игра, приключения.

Игрок посещает различные планеты, на которых находятся заброшенные строения. В этих местах он может отыскать шифры и сокровища. Шифры раскрывают историю всего мира или места, где шифр был найден. Так же раскрытые сообщения могут вести на новые локации и содержать в себе координаты новых мест с сокровищами.

В игре имеется игровая валюта, которая начисляется за продажу сокровищ на базе через компьютер. После продажи предмет исчезает из инвентаря, а игрок получает игровую валюту.

Каждое строения является процедурно генерируемым.

Чтобы разгадать шифр требуется купить специальное оборудование для ручной дешифровки или брутфорса ключа. Оборудование можно улучшать, повышая его скорость. Оборудования размещается на базе игрока.

Покупка предметов и оборудования реализуется через компьютер на базе игрока. Купленное оборудование доставляется в коробках, после нажатия на которые размещается автоматически на базе или попадает в инвентарь.

Игрок может купить некоторые предметы, которые обладают свойствами временно используемых, например, одноразовый взлом дверей.

Предметы и сокровища, найденные игроком, размещаются в инвентаре, который ограничен определённом количеством ячеек. Шифры неограниченно размещаются в папке пользователя.

Пользователь имеет папку, в которой в любой момент может просмотреть найденные шифры и перечитать их.

У игрока есть корабль, который он используется для путешествия между разными локациями и базой.

В строениях находятся монстры, которые мешают игроку забрать сокровища и шифры, нанося ему урон. Если игрок теряет все свои жизни и товарищей, то игра начинается заново, сбрасывая весь прогресс игроков. В случае, если хотя бы один игрок вернулся на базу живым, то остальные игроки возродятся, потеряв своё оборудование и игровую валюту. Игрок стремится к богатству и знаниям, постепенно открывая всё новые и новые загадки.

Игрок улучшает свою базу и закупает новое оборудование. Исследует заброшенные строения, побеждает монстров и забирает сокровища.

Так же в компьютере можно пройти обучение различным шифрам, посмотрев презентации.

**1.1.3 Описание входной, выходной и условно-постоянной информации**

Входной информацией является:

* Уровень сложности, выбранный пользователем;
* Настройки игры.

Выходной информацией является:

* Статистика игры, прогресс пользователя, достижения.

В качестве условно постоянной информации выступает:

* Правила игры, описание уровней, параметры приборов.
* Текстовые документы, таблицы.

**1.1.4 Нефункциональные требования**

Требования к применению: игра предоставляет возможность быстро начать игру, сохранить прогресс и получить обратную связь от системы, а также возможность узнать о статусе текущих заданий и достижений.

Требования к реализации: игра будет разработана с использованием современных технологий, таких как Unity для обеспечения высокого качества графики и производительности. Продукт поддерживает многопользовательский режим.

Требования к надежности: игра будет стабильной и надежной, с минимальными перебоями в работе.

Требования к интерфейсу: игра будет обладать интуитивно понятным и удобным интерфейсом. Будут предоставлены различные настройки для адаптации игры под предпочтения пользователя. Игра будет поддерживать различные языки интерфейса. Продукт будет иметь возможность получения помощи или подсказок внутри игры.

Требования к обратной связи: игра будет предоставлять обратную связь игроку о его прогрессе и достижениях, будет предусмотрена возможность обратной связи от игроков для улучшения игры.

Приложения будет ориентировано на десктоп, 64-битную систему под windows от 7 версии.

Минимальные требования:

* ОС: Windows 7/8/10;
* Процессор: Dual-core от 2.0 ГГц;
* Оперативная память: 8 ГБ;
* Видеокарта: NVIDIA GeForce GTX 1050;
* Место на диске: 2 ГБ свободного пространства;
* DirectX: Версия 11;
* Монитор с разрешением экрана full hd;
* Клавиатура 75%.

**1.2 Диаграмма вариантов использования**

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

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

Актером или действующим лицом является любая сущность, взаимодействующая с системой извне. Это может быть как живое существо, так и любая друга система, которая может служить источником воздействия на моделируемую систему так, как определяет сам разработчик. На рисунке 1 представлено графическое обозначение актера.

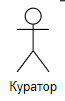


Рисунок 1 – Графическое обозначение актера

Вариант использования является стандартным языком UML и применяется для спецификаций общих особенностей системы и любой другой сущности. Отдельные варианты использования обозначаются на диаграмме эллипсом, в котором содержится его краткое название. Пример представлен на рисунке 2.

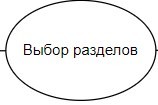


Рисунок 2 – Графическое обозначение вариантов использования

Отношение ассоциации является главным понятием языка UML и используется при построении всех графических моделей. Оно служит для обозначения роли актера и отдельном варианте использования. На диаграмме отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Пример отношения ассоциации представлен на рисунке 3.

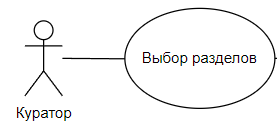
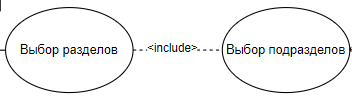


Рисунок 3 – Графическое обозначение отношения ассоциации

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данная линия помечается ключевым словом <<include>>. Пример изображен на рисунке 4.

Рисунок 4 – Графическое обозначение отношения включения

Определяя для выбранного актера варианты использования и устанавливая отношения между вариантами использования, получим полную диаграмму вариантов использования, её можно увидеть в Приложении А на рисунке А.1.

**1.3 Разработка плана работы над проектом**

Диаграмма Ганта – это популярный тип столбчатых диаграмм, который используется для иллюстрации плана, графика работ по какому-либо проекту.

Является одним из методов планирования проектов. Используется в приложениях по управлению проектами.

Диаграмма Ганта по разработке данного программного обеспечения представлена в приложении Б.

**1.4 Выбор стратегии разработки и модели жизненного цикла**

С помощью нескольких таблиц и ответов на вопросы возможно выяснить модель жизненного цикла программного обеспечения.

Выбор модели жизненного цикла на основе характеристик требований находится в таблице 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица 1 – Выбор модели жизненного цикла на основе характеристик требований | | | | | | |
| **Критерии категории требований** | **Каскадная** | **V-образная** | **RAD** | **Инкрементная** | **Быстрого прототипирования** | **Эволюционная** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| Являются ли требования к проекту легко определимыми и реализуемыми? | Да | Да | Да |  |  |  |
| Могут ли требования быть сформулированы в начале ЖЦ? | Да | Да | Да | Да |  |  |
| Часто ли будут изменяться требования на протяжении ЖЦ? |  |  |  | Да | Да | Да |
| Нужно ли демонстрировать требования с целью их определения? |  |  | Да | Да | Да | Да |
| Требуется ли проверка концепции программного средства или системы? |  |  | Да | Да | Да | Да |
| Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ? |  |  |  | Да | Да | Да |
| Нужно ли реализовать основные требования на ранних этапах разработки? |  |  | Да | Да | Да | Да |
| Итого | 2 | 2 | 5 | 6 | 5 | 5 |

Вычисления: 2 за каскадную, 2 за V-образную, 5 за RAD, 6 за инкрементную, 5 за быстрого прототипирования и 5 за эволюционную.

Итог: на основе результатов заполнения таблицы 1 подходящей является инкрементная модель.

Выбор модели жизненного цикла на основе характеристик команды разработчиков находится в таблице 2.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 2 – Выбор модели жизненного цикла на основе характеристик команды разработчиков | | | | | | | |
| **Критерии категории команды разработчиков проекта** | **Каскадная** | **V-образная** | **RAD** | **Инкрементная** | **Быстрого прототипирования** | **Эволюционная** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| Являются ли проблемы предметной области проекта новыми для большинства разработчиков? |  |  |  | Да | Да | Да |
| Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков? | Да | Да |  |  |  | Да |
| Изменяются ли роли участников проекта на протяжении ЖЦ? |  |  |  | Да | Да | Да |
| Является ли структура процесса разработки более значимой для разработчиков, чем гибкость? | Да | Да |  | Да |  |  |
| Важна ли легкость распределения человеческих ресурсов проекта? | Да | Да | Да | Да |  |  |
| Приемлет ли команда разработчиков оценки, проверки, стадии разработки? | Да | Да |  | Да | Да | Да |
| Итого | 4 | 4 | 1 | 5 | 3 | 4 |

Вычисления: 4 за каскадную, 4 за V-образную, 1 за RAD, 5 за инкрементную, 3 за быстрого прототипирования и 4 за эволюционную.

Итог: на основе результатов заполнения таблицы 2 подходящей является инкрементная модель.

Выбор модели жизненного цикла на основе характеристик коллектива пользователей находится в таблице 3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Таблица 3 – Выбор модели жизненного цикла на основе характеристик коллектива пользователей | | | | | | |
| **Критерии категории коллектива пользователей** | **Каскадная** | **V-образная** | **RAD** | **Инкрементная** | **Быстрого прототипирования** | **Эволюционная** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| Будет ли присутствие пользователей ограничено в ЖЦ разработки? | Да | Да |  | Да | Да | Да |
| Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки? |  |  |  | Да | Да | Да |
| Будут ли пользователи вовлечены во все фазы ЖЦ разработки? |  |  | Да | Да | Да | Да |
| Будет ли заказчик отслеживать ход выполнения проекта? |  |  |  | Да | Да | Да |
| Итого | 1 | 1 | 1 | 4 | 4 | 4 |

Вычисления: 1 за каскадную, 1 за V-образную, 1 за RAD, 4 за инкрементную, 4 за быстрого прототипирования и 4 за эволюционную.

Итог: на основе результатов заполнения таблицы 3 подходящими являются инкрементная, быстрого прототипирования и эволюционная модели.

Выбор модели жизненного цикла на основе характеристик типа проектов и рисков находится в таблице 4.

Таблица 4 – Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Критерии категории типов проекта и рисков** | **Каскадная** | **V-образная** | **RAD** | **Инкрементная** | **Быстрого прототипирования** | **Эволюционная** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| Разрабатывается ли в проекте продукт нового для организации направления? |  |  |  | Да | Да | Да |
| Будет ли проект являться расширением существующей системы? | Да | Да | да | Да |  |  |
| Будет ли проект крупно- или среднемасштабным? |  |  |  | Да | Да | Да |
| Ожидается ли длительная эксплуатация продукта? | Да | Да |  | Да |  | Да |
| Необходим ли высокий уровень надежности продукта проекта? |  | Да |  | Да | Да | Да |
| Предполагается ли эволюция продукта проекта в течение ЖЦ? |  |  |  | Да | Да | Да |
| Велика ли вероятность изменения системы (продукта) на этапе сопровождения? |  |  |  | Да | Да | Да |
| Является ли график сжатым? |  |  | Да | Да | Да | Да |
| Предполагается ли повторное использование компонентов? |  |  | Да | Да | Да | Да |
| Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)? |  |  |  |  | Да | Да |
| Итого | 2 | 3 | 4 | 9 | 8 | 9 |

Вычисления: 2 за каскадную, 3 за V-образную, 4 за RAD, 9 за инкрементную, 8 за быстрого прототипирования и 9 за эволюционную.

Итог: на основе результатов заполнения таблицы 4 подходящей является инкрементная модели.

Общий итог: в итоге заполнения табл. 1 – 4 наиболее подходящей является инкрементная модель, однако так как не все требования известны, по итогу выбрана эволюционная модель.

Общая статистика по всем таблицам:

* Каскадная: 9;
* V-образная: 10;
* RAD: 11;
* Инкрементная: 24;
* Быстрого прототипирования: 20;
* Эволюционная: 22.

**1.5 Инструменты разработки**

Для разработки проекта выбран игровой движок Unity версия 2022.3.25f1 с использованием языка C#, так как он является объектно-ориентированным и включает в себя большое количество удобного функционала для разработки 3д, 2д, а также игр с дополненной реальностью и виртуальной реальностью.

Выгоды от проектирования в среде Windows с помощью Unity:

* устраняется необходимость в повторном вводе данных;
* обеспечивается согласованность проекта и его реализации;
* увеличивается производительность разработки и переносимость программ.

Также для разработки программы необходимы:

* Help & Manual – редактор для создания справок;
* Visual Studio 2022 – редактор для написания кода;
* Word 2021 – необходим для написания пояснительной записки;
* Power Point 2021 – необходим для создания отчётной презентации;
* Drawio – сайт для создания диаграмм и схем;
* MagicaVoxel – программа для создания воксельных 3д моделей;
* Блокнот – для создания сохранений.

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

* процессор Intel Core i5;
* объем ОЗУ 16 гб;
* объем места на HDD – 1 тб;
* видеоподсистема 1920х1080 точек с глубиной цвета 32 Bit;
* ОС: Windows 10.

**2 Проектирование задачи**

**2.1 Разработка системы меню**

Для хорошей работы игрового приложения важно его проектирование, так как оно влияет на то, как игрок взаимодействуют с приложением. Качественное игровое приложение должно быть простым в использовании, иметь удобную навигацию, быстро загружаться и удовлетворять запросы игроков.

Полную структуру игрового приложения можно рассмотреть в приложении В.

**2.2 Разработка UML-диаграмм**

**2.2.1 Диаграмма последовательности**

Диаграмма последовательности – UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента.

Объекты обозначаются прямоугольниками с подчеркнутыми именами.

Сообщения (вызовы методов) – линиями со стрелками.

Возвращаемые результаты – пунктирными линиями со стрелками.

Прямоугольники на вертикальных линиях под каждым из объектов показывают «время жизни» (фокус) объектов.

На диаграмме отображён процесс декодирования сообщения.

Диаграмму последовательности можно рассмотреть в приложении Г.

**2.2.2 Диаграмма классов**

Диаграмма классов – UML-диаграмма, на которой изображено внутреннее устройство, а также отношения некоторой группы классов между собой.

Классы обозначаются прямоугольниками с пунктами внутри.

Пункты обозначают поля и методы соответственно, так же указана доступность методов и полей, их параметры и возвращаемый тип.

На диаграмме отображена иерархия наследования шифров, а так же их внутреннее устройство.

Диаграмму последовательности можно рассмотреть в приложении Д.

**2.3 Разработка пользовательского интерфейса**

Важным элементом проектирования данного программного продукта является описание внешнего интерфейса разрабатываемого игрового приложения.

Для разработки визуального дизайна продукта использовались сдержанные цвета для удобства использования программного продукта.

В ходе разработки был спроектирован дизайн для программного продукта «Crypto Dungeon».

Для организации эффективной работы пользователя нужно создать целостный программный продукт данной предметной области, в котором все компоненты будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс игрока. Программный продукт должен позволить пользователю решать задачи, затрачивая значительно меньше усилий, чем при работе с разрозненными объектами. Все исходные данные будут разделены на несколько групп.

Прототип – это наглядная модель пользовательского интерфейса. В сущности, это «черновик», созданный на основе представления разработчика о потребностях пользователя. Итоговое отображение программы может отличаться от прототипа. Прототипы UX представлены в приложении Ж.

**3 Реализация**

**3.1 Руководство программиста**

**3.1.1 Организация данных**

Основным средством хранения информации в приложении являются: obj, mtl модели для хранения игровых 3д моделей, png изображения для изображений предметов, кнопок, cs файлы для хранения скриптов. В проекте также присутствуют текстовые файлы для хранения прогресса прохождения.

Основными функциями приложения являются:

* вхождение в игру;
* выдача ответа на игровые вопросы;
* сбор листков с шифрами;
* расшифровка сообщений;
* подбор предметов;
* пауза игры;
* выбор предметов для покупки;
* выбор обучения;
* прохождение обучения.

**3.1.2 Структура программы**

Данный проект содержит 3 сцены, рассмотрим их назначение:

* сцена «MainMenu» – главное меню;
* сцена «Home» –­­ отвечает за главную игровую локацию;
* сцена «Tutorial» – является сценой для обучения;

Также проект содержит скрипты, которые отвечают за определённые

функции, рассмотрим их назначение:

* скрипт «ComputerButtonOperations» – содержит методы для работы меню компьютера;
* скрипт «GameMenuButtonOperations» – содержит методы для работы меню во время игры;
* скрипт «ShopButtonOperations» – содержит методы для работы меню магазина;
* скрипт «AlphabetManager» – отвечает за алфавиты разных языков;
* скрипт «Cipher» – является базовым классом, определяющим базовый интерфейс и поля;
* скрипт «Helper» – содержит вспомогательные методы для работы с шифрами;
* скрипт «CaesarsWithKeyСipher» – реализация шифра «Цезаря с ключом»;
* скрипт «CaesarsСipher» – реализация шифра «Цезаря»;
* скрипт «LadderCipher» – реализация шифра «Лесенки»;
* скрипт «ReverseCipher» – реализация шифра «Перевёртыша»;
* скрипт «SkitalaCipher» – реализация шифра «Скитала»;
* скрипт «VigenereCipher» – реализация шифра «Виженера»;
* скрипт «DebugCiphersButtonOperations» – содержит методы для отладки работы шифров;
* скрипт «DebugSavesButtonOperations» – содержит методы для отладки работы сохранений;
* скрипт «DebugLogs» – содержит методы для отладки работы приложения через логирование;
* скрипт «EasterEggsActions» – содержит действия для назначения, для последующей активации секретного события;
* скрипт «InputKeysReader» – отвечает за чтение ввода игрока;
* скрипт «InputKeysReaderWithChecker» – отвечает за чтение ввода игрока и проверки на соответствие ввода конкретной комбинации;
* скрипт «ChooseSlot» – отвечает за выбор текущего слота инвентаря;
* скрипт «Inventory» – отвечает за инвентарь игрока;
* скрипт «Items» – хранит данные об отдельном предмете;
* скрипт «PickUp» – отвечает за подбор предмета;
* скрипт «Slot» – отвечает за слот хранения предмета в инвентаре;
* скрипт «CiphersLogTest» – отвечает за проверку каждого шифра на корректность;
* скрипт «BasketItem» – отвечает за предметы в корзине;
* скрипт «Item» – отвечает за оюбом предмет
* скрипт «ShopItem» – отвечает за хранения данных об элементе;
* скрипт «ShopItemsActions» – отвечает за хранений дейсттвий предмета;
* скрипт «ShopItemScriptableObject» – отвечает за сохранений данных о предмете;
* скрипт «ShopScriptableObject» – отвечает за характеристик прелмета из памяти;
* скрипт «Basket» – отвечает за корзину игрока;
* скрипт «Shop» – отвечает за покупки;
* скрипт «AutoDecoder» – отвечает за автоматическое дешифрование сообщения;
* скрипт «Decoder» – родительский класс, устанавливающий интерфейс работы дешифрования;
* скрипт «MiniGameDecoder» – отвечает за чтение дешифровании сообщения через мини-игру
* скрипт «DecodeScreen» – отвечает за вывод текущего ввода пользователя;
* скрипт «Movement» – отвечает за передвижение игрока
* скрипт «PlayerInteracts» – содержит методы для обработки взаимодействий с миром;
* скрипт «Raycast» – отвечает за получение текущего предмета, на который нацелен игрок;
* скрипт «RandomCipherGenerator» – сооздаёт случайный шифр;
* скрипт «DataValidator» – отвечает за сравнение данных на правильность через Regex;
* скрипт «GameSaves» – отвечает за сохранения в игре;
* скрипт «SavesSerializer» – отвечает за сериализацию данных;
* скрипт «LoadingTimer» – отвечает за таймер;
* скрипт «TimeConverter» – отвечает за перевод секунд в правильный формат времени;
* скрипт «Subtitle» – отвечает за субтитры;
* скрипт «SubtitlesPlayer» – отвечает за чтение вывод субтитров;
* скрипт «Slider» – нужен для переключения слайдов презентации;
* скрипт «TutorialCameraRotation» – отвечает за вращения камеры на сцене обучения;
* скрипт «DrawImages» – отвечает за отображаемые изображения;
* скрипт «Enums» – содержит самописные перечисления;
* скрипт «MainMenu» – содержит методы для работы главного меню;
* скрипт «Paths» – отвечает за поддержание удобного доступа к файлам проекта;
* скрипт «PlayerScriptableObject» – отвечает за данные игрока;
* скрипт «TextConverter» – отвечает за красивое отображение текста;
* скрипт «Title» – отвечает за титры;
* скрипт «TitlesPlayer» – отвечает за воспроизведение титров в конце игры;

**3.1.3 Спецификация программы**

Точное название проекта и его состав приводится в Таблице 6.

Таблица 6 –– Спецификация программы

|  |  |
| --- | --- |
| **Имя файла** | **Назначение** |
| **1** | **2** |
| Crypto Dungeon.exe | Исполняемый файл проекта |
| \*.txt | Текстовые файлы для хранения информации о титрах в презентации обучения и титрах в конце игры |
|  |  |
| Продолжение таблицы №6 | |
| **1** | **2** |
| \*.uniti | Файл, который может содержит данные о сценах в Unity. |
| \*.uniti.meta | Метаданные для файла .uniti, которые содержат информацию о его идентификаторе и других свойствах. |
| \*.asset | Файл, содержащий данные активов (например, текстуры, модели, аудиофайлы и т.д.) в Unity. |
| \*.asset.meta | Метаданные для файла .asset, содержащие информацию о его идентификаторе и других свойствах. |
| \*.meta | meta - Общий файл метаданных для различных активов в Unity, содержащий информацию об идентификаторе и других свойствах. |
| \*.anim | anim - Файл анимации, который содержит данные о движении объектов в Unity. |
| \*.anim.meta | Метаданные для файла .anim. |
| \*.mat | Формат материалов в Unity |
| \*.mat.meta | Метаданные для файла .mat |
| \*.shader | Формат шейдеров в Unity |
| \*.shader.meta | Метаданные для файла .shader |
| \*.mp4 | Видео файлы |
| \*.DLL | Библиотека динамической компоновки, котораясодержит код на C# и используется в Unity. |
| \*.cs | Файл исходного кода на языке C#. |
| \*.cs.meta | файл метаданных для файла .cs. |
| \*.prefab | файл префаба, который представляет собой шаблон объекта в Unity. |
| \*.prefab.meta | файл метаданных для файла .prefab. |

**4 Тестирование**

**4.1 Тесты на использование**

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

Разработанные тест-кейсы и статус их выполнения представлены в приложение К.

Расписание работ над проектом представлено в таблице 6.

Таблица 6 – Расписание работ над проектом

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Дата** | **Деятельность** | **Продолжительность, ч** |
| Гейдель Евгений | 01.12.2024 | Разработка тестов | 1 |
| Гейдель Евгений | 02.12.2024 | Тестирование игрового приложения | 3 |
| Гейдель Евгений | 03.12.2024 | Составление отчетов о найденных дефектах | 2 |
| Гейдель Евгений | 03.12.2024 | Исправление найденных ошибок | 2 |
| Гейдель Евгений | 04.12.2024 | Проведение регрессионного тестирования | 2 |
| Гейдель Евгений | 04.12.2024 | Составление отчета о результатах тестирования | 3 |

**4.2 Отчёт о результатах тестирования**

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

Статистика по всем дефектам представлена в таблице 7.

Пройденные тесты можно найти в приложении К.

Таблица 7 – Статистика по всем дефектам

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

**5 Руководство пользователя**

Игровое приложение «Crypto Dungeon» позволит пользователю провести время с пользой, получить эстетическое удовольствие, поспособствует улучшению таких навыков, как логическое мышление, навыки запоминания, увеличит знания об криптографии, а также научит расшифровывать некоторые шифры.

Создаваемое приложение ориентировано на пользователей старше 16 лет.

Качество и скорость работы приложения всегда зависит от самих характеристик персонального компьютера. Поэтому приложение должно было быть протестировано на разных машинах.

Тестирование проводилось на разных персональных компьютерах и результаты были удовлетворительные.

Данный программный продукт можно использовать в любое время, так как он не ограничен в использовании.

Загрузка игрового приложения предполагает наличие свободной памяти в размере 2 Гб.

**5.1 Руководство системного программиста**

Для того, чтобы установить программу необходимо разархивировать файл CryptoDungeon.zip.

**5.2 Запуск программы**

После этого, как игра была разархивирована, по двойному нажатию на «Crypto Dungeon.exe» игра будет запускаться.

**5.3 Инструкция по работе с программой**

При запуске игры пользователя встречает главное меню, в котором игрок может начать игру, настроить игру и выйти из игры. Главное меню представлено на рисунке №5.



Рисунок 5 – Главное меню

У пользователя есть возможность кликнуть по кнопке одиночная игра, сетевая игра, настройки, выход.

При нажатии на кнопку «Сетевая игра» ничего случится, она разработана для дальнейшей поддержки приложения.

При нажатии на кнопку «Выход» приложение будет закрыто.

При нажатии на кнопку «Настройки» будет выполнен переход в меню настроек, пока что оно пустое, разработано для дальнейшей поддержки приложения. Меню настроек представлено на рисунке №6.



Рисунок 6 – Меню настроек

При нажатии на кнопку «Назад» мы вернемся в главное меню

При нажатии на кнопку «Одиночная игра» будет выполнен переход в меню выбора сохранения. Меню выбора сохранений представлено на рисунке №7.



Рисунок 7 – Меню выбора сохранения

При нажатии на кнопку «Новая игра» мы перейдём на сцену, в которой происходят игровые события. Игровая сцена представлено на рисунке №8.



Рисунок 8 – Игровая сцена

Управление персонажем осуществляется мышкой и при помощи кнопок W, A, S, D, а также пробела: пробел отвечает за прыжок, W отвечает за ходьбу вперед, A – ходьба влево, S – ходьба назад, D – ходьба вправо, вращение мышкой отвечает за управление камерой.

При приближении к предметам, а также запискам появляется возможность подобрать их, нажав кнопку E. Записка представлена на рисунке №9

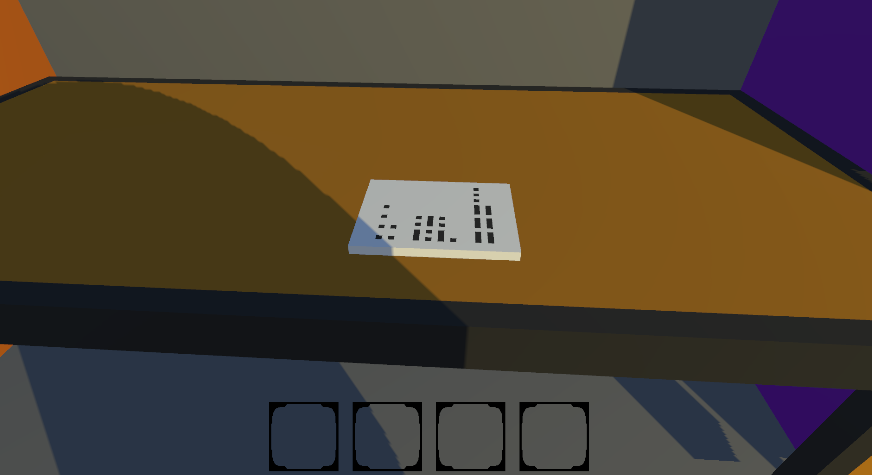


Рисунок 9 – Записка на столе

Нажав кнопку Е, перекладываем записку в инвентарь. Это видно на рисунке №10.



Рисунок 10 – Записка в инвентаре

Записку можно положить в фиолетовый дешифратор, который узнает шифр записки, для дальнейшего использования, нажав кнопку Е с выбранной запиской. Фиолетовый дешифратор представлен на рисунке №11.

Нажав кнопку «Е» по окончанию таймера мы заберем записку в инвентарь. Перемещение записки в инвентарь видно на рисунке №12.



Рисунок 11 – Процесс определения шифра



Рисунок 12 – Записка в инвентаре

После повторного получения записки, можно подойти к зеленому дешифратору и начать ручную дешифровку, нажав кнопку Е. Ручную дешифровку можно видеть на рисунке №13.



Рисунок 13 – Ручная дешифровка

**5.4 Завершение работы с программой**

Завершить работу игрового приложения можно двумя способами:

* в главном меню присутствует кнопка «Выход», после нажатия на которую игра закрывается, чтобы перейти в главное меню из игры достаточно нажать кнопку «esc», затем выбрать меню;

**Заключение**

Целью данной практики по предмету ТРПО является разработка и проектирование программного продукта – игрового приложения «Crypto Dungeon». Данный программный продукт позволит пользователю провести время с пользой, получить эстетическое удовольствие, поспособствует улучшению таких навыков, как логическое мышление, навыки запоминания, увеличит знания об криптографии, а также научит расшифровывать некоторые шифры.

В ходе реализации поставленной задачи были получены знания по использованию платформы Unity, а также закреплены знания курса предмета «Технология разработки программного обеспечения».

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

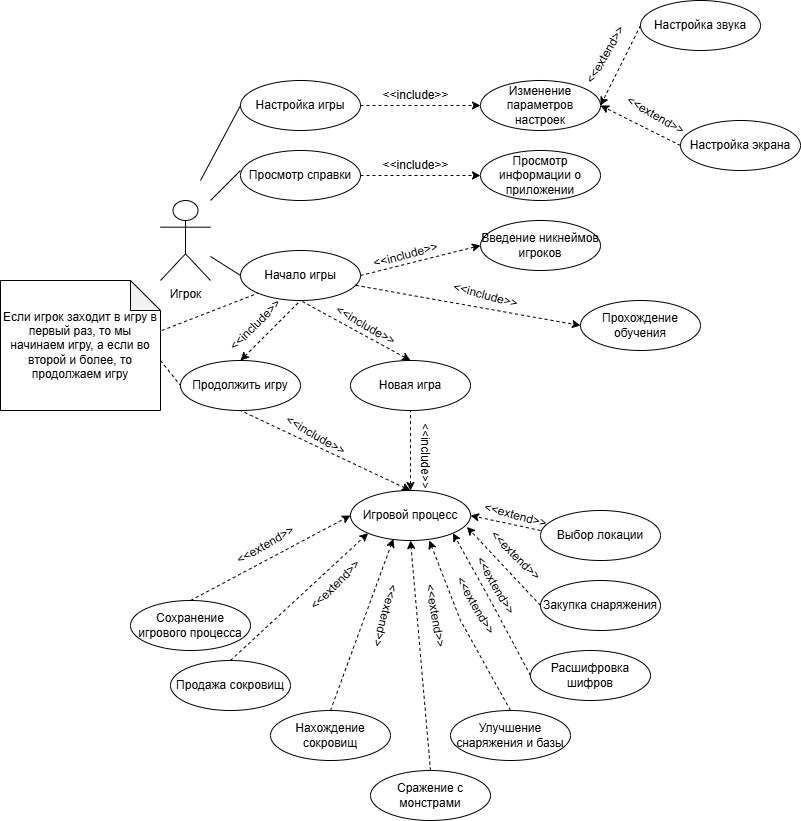
В ходе тестирования все исключительные ситуации были обработаны. Проект работает без сбоев и ошибок. В поставленной задаче был реализован простой и понятный пользовательский интерфейс.

Исходя из этого, можно сделать вывод, что прототип программы реализована успешно.

**Список использованных источников**

* 1. Саймон Сингх Книга шифров 1999 416 страниц [Книга] – Дата доступа: 07.11.2024;
  2. Википедия [Электронный ресурс] –<https://ru.wikipedia.org/wiki/Шифр_Виженера> – Дата доступа: 10.11.2024;
  3. Википедия [Электронный ресурс] –<https://ru.wikipedia.org/wiki/Шифр_Цезаря> – Дата доступа: 15.11.2024;
  4. Википедия [Электронный ресурс] –https://ru.wikipedia.org/wiki/AES\_(стандарт\_шифрования) – Дата доступа: 20.11.2024.

**Приложение А**   
Диаграмма вариантов использования

Рисунок А.1 – Диаграмма вариантов использования

**Приложение Б**   
Диаграмма Ганта

Рисунок Б.1 – Диаграмма Ганта

**Приложение В**   
Структура игрового приложения

Рисунок В.1 – Структура игрового приложения

**Приложение Г**   
Диаграмма последовательности

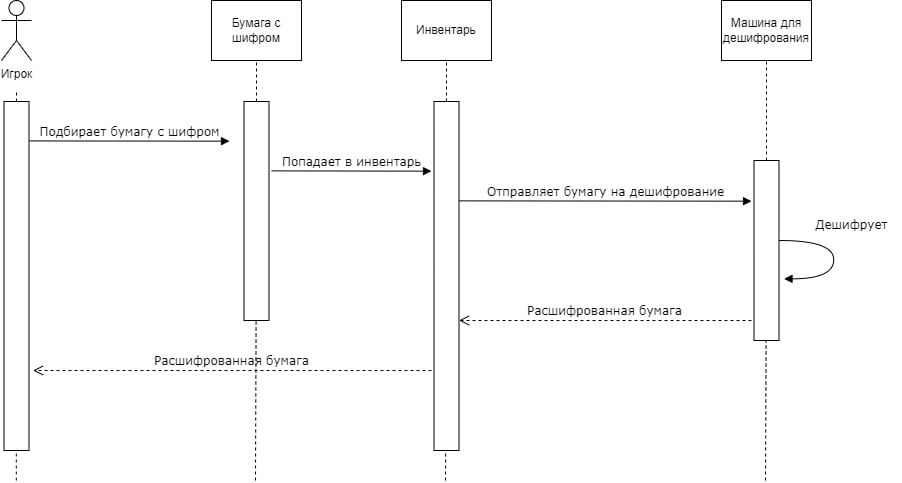
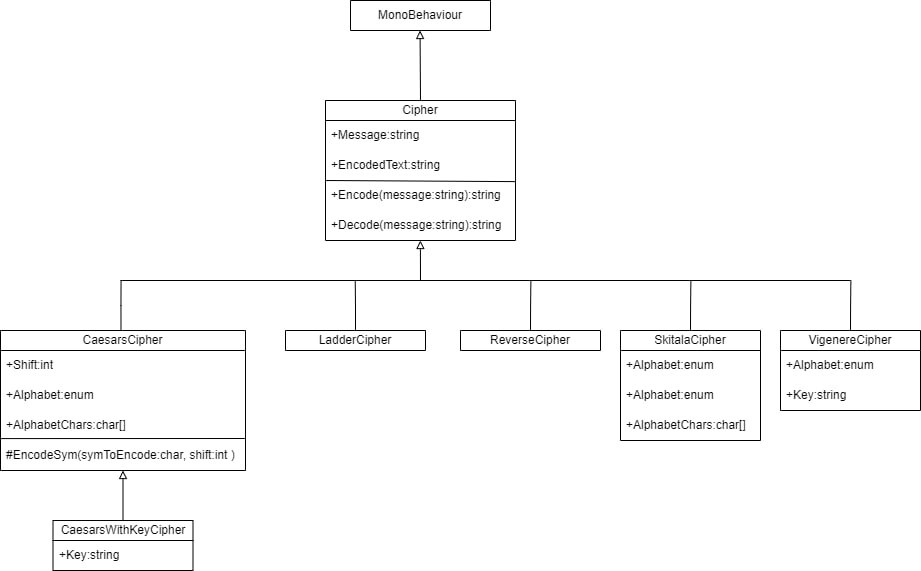
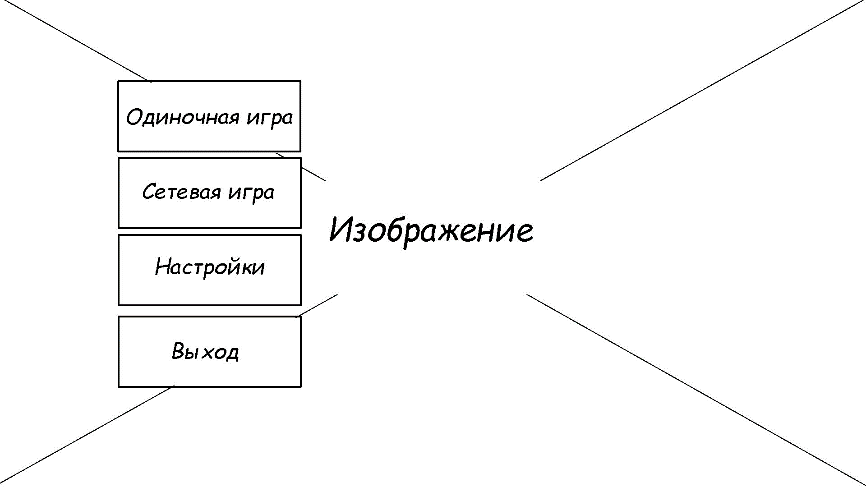
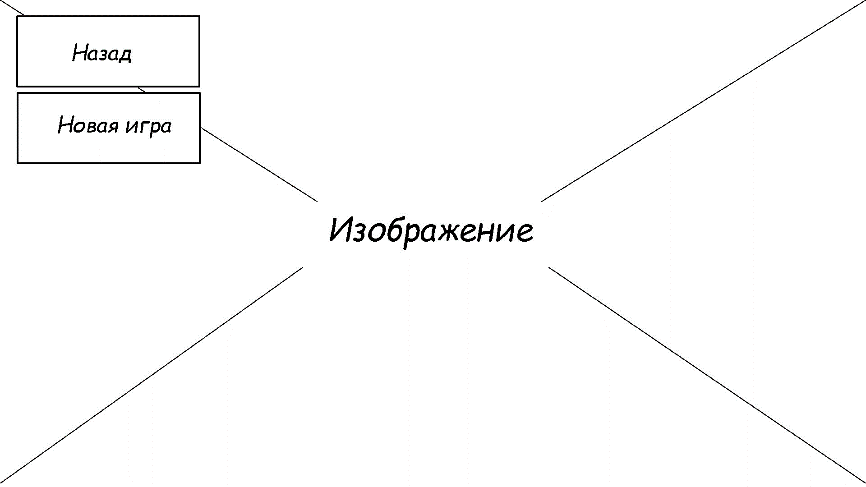


Рисунок Г.1 – Диаграмма последовательности

**Приложение Д**   
Диаграмма классов

Рисунок Д.1 – Диаграмма классов

**Приложение Ж**   
UX прототипы

 Рисунок Ж.1 – UX прототип главного меню - Главная

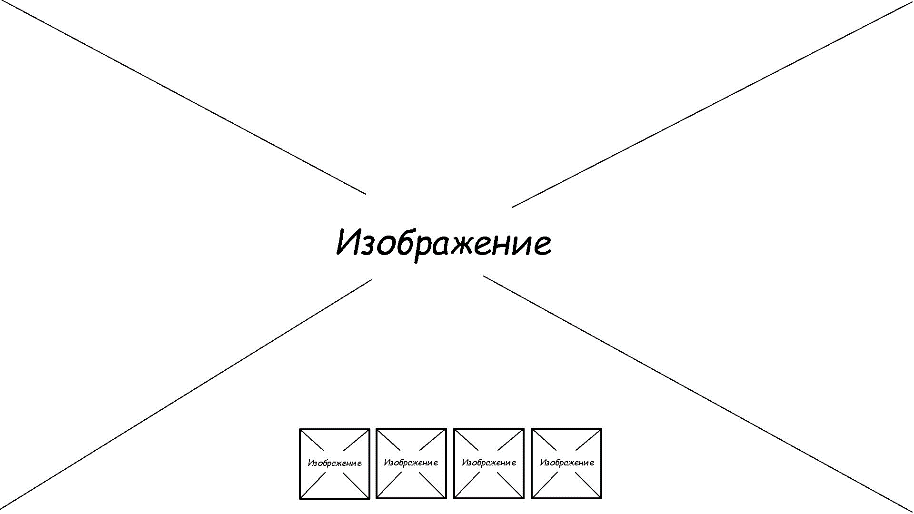
 Рисунок Ж.2 – UX прототип главного меню – Одиночная игра

Рисунок Ж.3 – UX прототип инвентаря

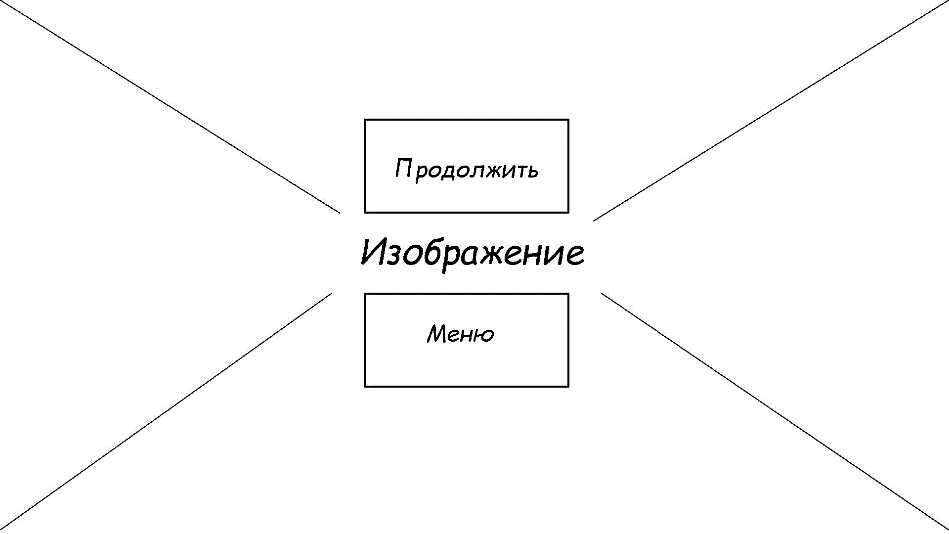
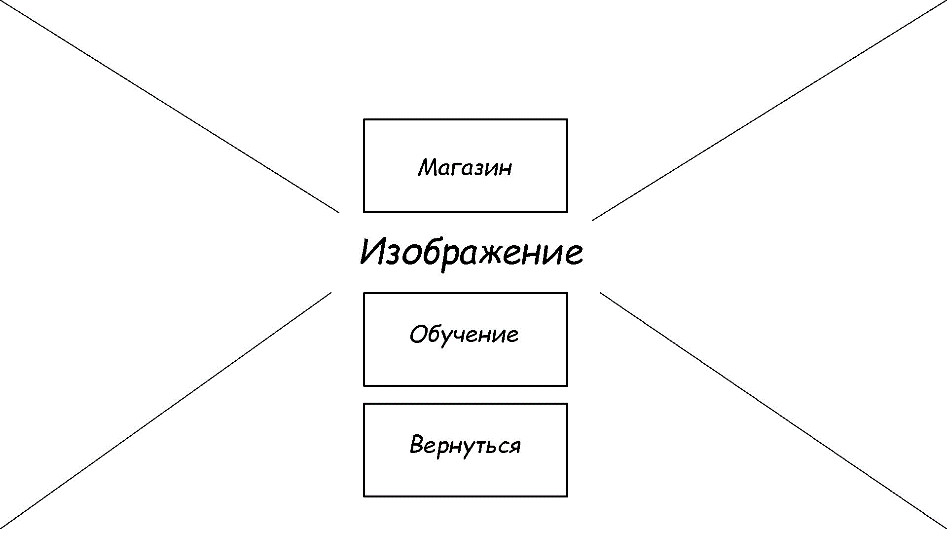
 Рисунок Ж.4 – UX прототип меню паузы

Рисунок Ж.5 – UX прототип меню компьютера

**Приложение И**  
Протестированные функции