**Введение**

Целью учебного проекта заключается в разработке игрового приложения «Virtual World», предназначенного, для отвлечения от реального мира, расслабления и хорошего времепровождения.

Создаваемая программа будет рассчитана для людей старше 7 лет.

В пояснительной записке проекта представлены следующие разделы.

Раздел «Анализ задачи» содержит описание постановки задачи, включая исследование предметной области, определение организационно-экономической сущности задачи и существующих методов решения.

В разделе «Инструменты разработки» описана среда разработки, используемая для создания игрового продукта «Virtual World», а также указаны минимальные и оптимальные требования к аппаратным характеристикам.

Раздел «Проектирование задачи» содержит описание основных аспектов разработки игрового продукта, включая организацию данных, пользовательский интерфейс, алгоритмы обработки информации.

В разделе «Реализация задачи» описываются все элементы и объекты, используемые при реализации игрового приложения, а также функции пользователя и их структура.

Раздел «Тестирование» описывает процесс 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.

«Заключение» содержит краткую формулировку задачи, основные результаты работы, описание использованных методов и средств, а также степень автоматизации процессов на различных этапах разработки.

В “Литературе” будет приведен список используемых при разработке источников.

В приложениях к пояснительной записке будет представлен листинг программы.

Схемы работы системы приложения будет представленна в графической части.

1. **Анализ задачи**
   1. **Постановка задачи**
2. **Организационно-экономическая сущность задачи**

**Наименование задачи:** Virtual World (игра)

**Цель разработки:** создать симулятор жизни в качестве игры

**Назначение:** данный программный продукт разрабатывается для людей любого возраста, желающим поиграть

**Периодичность использования:** по желанию

**Источники и способы получения данных:** наблюдения за поведением персонажей, анализ подобных игра

**Обзор существующих аналогичных ПП:** Проанализированы следующие аналогичные продукты в этой сфере:

1. The Sims

The Sims - это однопользовательская интерактивная игра-симулятор жизни, которая дает вам возможность создавать и управлять своим собственным персонажем Sims. Вы можете создавать семьи, строить дома и выполнять различные повседневные дела, такие как сон, еда, работа и т.д.

1. Spore

Симулятор не отдельно взятой цивилизации и даже не всей планеты. Это симулятор целой вселенной. Игра стартует на самой нижней ступеньке эволюции. Мы, в роли амебы, плаваем в океане, поедаем другие микроорганизмы и всячески боремся за свое, пока еще микроскопическое место под солнцем.

1. **Функциональные требования**

Описание перечня функций и задач, которые должен выполнять будущий ПП:

Игрок:

1. Выбор персонажа

2. Выбор локации

3. Выбор действий

4. Выбор предмета

5. Открытие карты

6. Просмотр достижений

7. Настройки

Правила:возрастное ограничение 7+, при получении достижения игроку открывается новый вид одежды, предмета и т.п. и деньги.

Сюжет игры: Открывается приложение и идёт загрузка игры. Мы попадаем на начальный экран, после чего нажимаем кнопку «Создать новый мир». Создается новый мир, игрок выбирает внешность своему персонажу (выбрать пол, цвет глаз, цвет волос, одежда, обувь), далее оказывается на локации дома, может видоизменять предметы вокруг себя на данной локации. Можно посетить локацию магазина, после получения 3-х достижений, на этой локации можно покупать различные предметы. Будет базовый капитал с начала игры, позже деньги можно будет получить за выполнения различных видов достижений. Чтобы заработать деньги, пользователь может зайти в меню достижений и посмотреть их список.

Управление: мышью и клавишами клавиатуры WASD

Взаимодействие игрока с предметами: по щелчку мыши

Перечень предметов, с которыми игрок можно взаимодействовать:

Дом:

* Плита – при нажатии можно выбрать действие «Готовка»
* Кровать - при нажатии можно выбрать действие «Спать»
* Диван - при нажатии можно выбрать действие «Сесть»
* Телевизор - при нажатии можно выбрать действие «Смотреть канал»
* Ванна - при нажатии можно выбрать действие «Принять ванну», «Сходить в душ»
* Санузел - при нажатии можно выбрать действие «Справить нужду»
* Книжный шкаф - при нажатии можно выбрать действие «Почитать книгу»
* Стул - при нажатии можно выбрать действие «Сесть»
* Игровая приставка – при нажатии можно выбрать действие «Играть»
* Холодильник – при нажатии можно выбрать действия «Открыть», «Взаимодействие с продуктами»
* Стол – при нажатии можно выбрать действие «Положить на стол»

Магазин:

Игрок может перемещаться по разным отделам: Продукты, мебель, игрушки, книги, бытовая техника, одежда. Игрок может взять продукты в магазине с полок и при нажатии левой кнопки мыши, продукты автоматически попадают в корзину. Из корзины продукты можно убрать, нажав на крестик. Продукты будут возвращаться на полки. Продуктов можно положить неограниченное количество в корзину. Для покупки продуктов необходимо нажать на кнопку купить. Игрок перемещается к кассе, где кассир сообщает сумму покупки. Необходимо нажать на кнопку оплатить, если денег недостаточно, то кассир говорит «недостаточно средств» и покупка не производится. Игроку предлагается убрать продукты из корзины.

1. **Описание процессов с входной, выходной и условно-постоянной информацией**

Таблица 1 – Функции программы с описанием с входной, выходной и условно-постоянной информации

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Категория пользо-вателей | Наименова-ние процесса | Краткое описание алгоритма выполнения процесса | Входная информация | Выходная информация | Условно  -  Постоянная  Информация |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | Игрок | Выбор персонажа | При входе в игру дается возможность выбрать внешность персонажа из возможных вариантов | Отсутствует | Игрок будет играть за персонажа с выбранной внешностью | Данные игры |
| 2 | Игрок | Выбор локаций | Возможность посетить 2 локации в мире игры | Отсутствует | Персонаж игрока будет находиться на выбранной локации | Данные игры |
| 3 | Игрок | Выбор действий | Игрок может задавать своему персонажу определенные действия | Отсутствует | Персонаж будет выполнять выбранные игроков действия | Данные игры |
| 4 | Игрок | Выбор предмета | Игрок может выбрать предмет, с которым будет взаимодействовать персонаж | Отсутствует | Получение выбора действий | Данные игры |
| 5 | Игрок | Открытие карты | Игрок может открыть карту мира чтобы выбрать локацию | Отсутствует | Дальнейшая возможность выбора локации | Отсутствует |
| 6 | Игрок | Настройки | Игрок может поменять настройки игры | Звук, музыка, разрешение | Применение новых настроек | Варианты настроек |

1. **Эксплуатационные требования**

Требования к применению: Помогает отвлечься от реальной жизни и расслабиться

Требования к реализации: Для разработки данного ПП должен использоваться язык Unity, основанный на C#

Требования к надёжности: дальнейшие обновления ПО

Требования к интерфейсу: брать в пример локации, действия, предметы, персонажей из реальной жизни

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

Таблица 2 **–** Выбор модели жизненного цикла на основе характеристик требований:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории требований | Каскадная | V-  образная | RAD | Инкрементная | Быстрого  прототипиро  вания | Эволюционная |
| 1. | Являются ли требования к проекту легко определимыми и реализуемыми? | Да | Да | Да | Нет | Нет | Нет |
| 2. | Могут ли требования быть сформулированы в начале ЖЦ? | Да | Да | Да | Да | Нет | Нет |
| 3. | Часто ли будут изменяться требования на протяжении ЖЦ? | Нет | Нет | Нет | Нет | Да | Да |
| 4. | Нужно ли демонстрировать требования с целью их определения? | Нет | Нет | Да | Нет | Да | Да |
| 5. | Требуется ли проверка концепции программного средства или системы? | Нет | Нет | Да | Нет | Да | Да |
| 6. | Будут ли требования изменяться или уточняться с ростом сложности системы  (программного средства) в ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Нужно ли реализовать основные требования на ранних этапах разработки? | Нет | Нет | Да | Да | Да | Да |

**Вычисления:** 3 за каскадную, 3 за V- образную, 6 за RAD, 4 за инкрементную, 4 за быстрого прототипирования и 4 за эволюционную.

**Итог:** на основе результатов заполнения табл. 2 подходящей является RAD модель и инкрементная модель.

Таблица 3 – Выбор модели жизненного цикла на основе характеристик команды разработчиков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории команды разработчиков проекта | Каскадная | V-  образная | RAD | Инкрементная | Быстрого  прототипирова  ния | Эволюционная |
| 1. | Являются ли проблемы предметной области проекта новыми для большинства разработчиков? | Нет | Нет | Нет | Нет | Да | Да |
| 2. | Являются ли инструментальные средства, используемые в проекте,  новыми для большинства разработчиков? | Да | Да | Нет | Нет | Нет | Да |
| 3. | Изменяются ли роли участников проекта на протяжении ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Является ли структура процесса разработки более значимой для разработчиков, чем гибкость? | Да | Да | Нет | Да | Нет | Нет |
| 5. | Важна ли легкость распределения человеческих ресурсов проекта? | Да | Да | Да | Да | Нет | Нет |
| 6. | Приемлет ли команда разработчиков оценки, проверки, стадии разработки? | Да | Да | Нет | Да | Да | Да |

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

**Итог:** на основе результатов заполнения табл. 3 подходящими являются каскадная, V-образная и каскадная модели.

Таблица 4 **–** Выбор модели жизненного цикла на основе характеристик коллектива пользователей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории коллектива пользователей | Каскадная | V  образная | RAD | Инкрементная | Быстрого  прототипирован  ия | Эволюционная |
| 1. | Будет ли присутствие пользователей ограничено в ЖЦ разработки? | Да | Да | Нет | Да | Нет | Да |
| 2. | Будут ли пользователи оценивать текущее состояние программного | Нет | Нет | Нет | Да | Да | Да |
| 3. | Будут ли пользователи вовлечены во все фазы ЖЦ разработки? | Нет | Нет | Да | Нет | Да | Нет |
| 4. | Будет ли заказчик отслеживать ход выполнения проекта? | Нет | Нет | Нет | Нет | Да | Да |

**Вычисления:** 2 за каскадную, 2 за V-образную, 2 за RAD, 1 за инкрементную, 2 за быстрого прототипирования и 1 за эволюционную.

**Итог:** на основе результатов заполнения табл. 4 подходящей являются каскадная V-образная, RAD модели и модель быстрого прототипирования.

Таблица 5 **–** Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № критерия | Критерии категории типов проекта и рисков | Каскадная | V  образная | RAD | Инкрементная | Быстрого  прототипиров  ания | Эволюционная |
| 1. | Разрабатывается ли в проекте продукт нового для организации направления? | Нет | Нет | Нет | Да | Да | Да |
| 2. | Будет ли проект являться расширением существующей системы? | Да | Да | Да | Да | Нет | Нет |
| 3. | Будет ли проект крупно- или среднемасштабным? | Нет | Нет | Нет | Да | Да | Да |
| 4. | Ожидается ли длительная эксплуатация продукта? | Да | Да | Нет | Да | Нет | Да |

Продолжение таблицы 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5. | Необходим ли высокий уровень надежности продукта проекта? | Нет | Да | Нет | Да | Нет | Да |
| 6. | Предполагается ли эволюция продукта проекта в течение ЖЦ? | Нет | Нет | Нет | Да | Да | Да |
| 7. | Велика ли вероятность изменения системы (продукта) на этапе сопровождения? | Нет | Нет | Нет | Да | Да | Да |
| 8. | Является ли график сжатым? | Нет | Нет | Да | Да | Да | Да |
| 9. | Предполагается ли повторное использование компонентов? | Нет | Нет | Да | Да | Да | Да |
| 10. | Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)? | Нет | Нет | Нет | Нет | Да | Да |

**Вычисления:** 3 за каскадную, 4 за V-образную, 4 за RAD, 8 за инкрементную, 6 за быстрого прототипирования и 8 за эволюционную.

**Итог:** на основе результатов заполнения табл. 5 подходящей является инкрементная модель.

**За каскадную – 12**

**За V-образную – 13**

**За RAD – 15**

**За инкрементную – 17**

**За быстрого прототипирования – 15**

**За эволюционную - 15**

**Общий итог: *в итоге заполнения таблиц – наиболее подходящей является инкрементная модель***

# **Инструменты разработки**

Для разработки игрового приложения "Virtual World" была выбрана среда разработки Unity. В данном разделе представлено обоснование выбора этой среды и требования к аппаратным и операционным ресурсам для обеспечения правильного и эффективного функционирования приложения.

Unity — кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие визуальной сред разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

Для обеспечения правильного и наиболее эффективного функционирования игрового приложения "Virtual World" рекомендуются следующие требования к аппаратным и операционным ресурсам:

* разрешение экрана: для корректного отображения программы рекомендуется использовать разрешение экрана не менее 1920x1080 или 1366x768 пикселей. Если разрешение экрана ниже указанных значений, программа может быть автоматически отображена с использованием разрешения 800x600 пикселей. В этом случае возможно снижение визуального качества и удобства использования;
* операционная система: Игровое приложение "Virtual World" разработано для работы на операционной системе Windows. Рекомендуется использовать современные версии операционной системы, такие как Windows 10 или более поздние, чтобы обеспечить оптимальную совместимость и функциональность;
* оптимальная конфигурация: для наилучшей производительности и эффективности работы приложения рекомендуется использовать компьютеры с более высокими характеристиками, такими как мощный процессор, достаточный объем оперативной памяти и быстрый жесткий диск. Это позволит обеспечить быструю обработку данных и отзывчивость приложения при выполнении сложных вычислений;

Перечисленные требования к аппаратным и операционным ресурсам являются рекомендациями для обеспечения оптимальной работы игрового приложения "Virtual World". Приложение может функционировать и на более старых или менее мощных конфигурациях, но может возникнуть снижение производительности или некоторые ограничения.

Конфигурация ПК:

* процессор Intel(R) Core(TM) i3-10110U CPU @ 2.10GHz 2.59 GHz;
* объём ОЗУ 8,00 ГБ;
* операционная система Windows 10.

Вспомогательные ПО:

* Figma;
* C#;
* Photoshop;
* Canva

1. **Проектирование задачи**
   1. **Разработка системы меню, навигации**



Рисунок 1 – Главное меню

* 1. **Организация данных**

Проектирование задачи – это очень важный и ответственный этап в разработке любого приложения.

Важным является он вследствие того, что методы, по средствам которых пользователь управляет формами, построены на высокой степени специализации каждого из компонентов.

Необходимым условием при разработке данного приложения является описание организации данных, т.е. логическая и физическая структура данных в контексте среды разработки. В разрабатываемой программе будут использоваться три вида данных.

Первым видом являются данные, которые будут введены разработчиком на этапе реализации задачи. Сюда можно отнести изображения (иконки), описание, исходные коды сортировок.

Вторым видом данных, используемых в программе, является вводимая пользователем информация. Входной информацией в разрабатываемой программе будут являться данные, которые будут отсортированы: количество данных (количество элементов), и сами данные (элементы). Количество возможных введенных пользователем элементов будет ограничено, а именно 15 штук. Тип данных: string (т. е пользователь может вводить только текст). Необязательно заполнять сами элементы вручную, поскольку эти данные, по желанию пользователя, могут быть сгенерированы самой программой. Подходящей структурой данных для хранения элементов является: динамический одномерный массив.

Третьим видом данных является результат программы – отсортированные пользовательские данные. Его также относят к отдельному виду, так как ни пользователь, ни разработчик его не вводят, а программа сама получает его в результате выполнения определенных действий.

Таким образом, организация данных является важной задачей при разработке данной и любой программы.

# **Процессы**

Согласно всем перечисленным требованиям и указаниям, которые были рассмотрены в разделе «Анализ задачи», было определено, чем конкретно должна заниматься разрабатываемая программа.

Было бы важным на этапе проектирования рассмотреть еще состав и внешний вид используемой формы. Хотя проектирование формы заранее не так принципиально.

Разработанная функциональная модель представлена в приложении В.

* 1. **Разработка пользовательского интерфейса**

Поставленной задачей на практику была реализация ux/ui интерфейсов. При разработке интерфейсов были использованы следующие разрешения: 1440px, 834px, 360px. Использовались преимущественно оттенки всех цветов. Основные разделы доступны с первой страницы. Таким образом был реализован понятный пользовательский интерфейс, созданы макеты под такие устройства как: компьютер. Целью проекта была реализацияя интерфейса.

<https://www.figma.com/file/pYC6sFbHthF1OebkomnEgn/Untitled?type=design&node-id=0%3A1&mode=design&t=7zrtTJkXV9jFkeQe-1>

https://www.figma.com/file/gL7zvmhLThHLzUIpxWqXYy/UX?type=design&node-id=0%3A1&mode=design&t=KxP6LYDCRJDPRpZX-1

# **2.5 Разработка UML-диаграмм**

UML-диграммы нужны для создания «чертежей» программы, схем, которые показывают, как будет устроено программное обеспечение изнутри, — то есть для проектирования. В данном проекте будет представленно 4 UML-диграммы: диаграмма вариантов использования и диаграмма последовательности.

На диаграммах вариантов использования отображается взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему.

Разработанная диаграмма вариантов использования представлена в приложении Г.

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. На диаграмме последовательности более детально рассмотренно оформление заказа.

Разработанная диаграмма последовательности представлена в приложении Д.

Разработанная диаграмма деятельности позволяет более детально визуализировать функционал в профиле сотрудника.

Разработанная диаграмма деятельности представлена в приложении Е.

Диаграмма классов демонстрирует общую структуру иерархии классов системы в ПП, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними.

Разработанная диаграмма классов представлена в приложении Ж.

Диаграмма объектов описывает оформление заказа продукции шеф-повара.

Разработанная диаграмма объектов представлена в приложении К.

# **Реализация**

* 1. **Руководство программиста**
     1. **Организация данных**

В данном приложении я использую данные, которые встроены в Unity, также используются библиотеки, этих данных будет достаточно для реализации проекта. Также в это приложение я вношу свои данные: картинки, текст, анимацию, библиотеки, звук и музыку, листинг, который позволит выполнить некоторые действия.

* + 1.  **Структура программы**

Рисунок 2 – Главное меню

На существующей странице приложения присутствует меню, содержащую в себе всю информацию, которая есть в самом приложении. Каждый пункт навигации приложения соответствует разделу приложения.

* + 1. **Структура и описание процедур и функций пользователя**

Игрок может переходить по кнопкам меню, получать достижения, переходить по локациям, менять гардероб, редактировать музыку и звук кнопок, также игрок может взаимодействовать с предметами в доме, может покупать продукты в магазине и оплачивать их.

* + 1. **Спецификация программы**

Данный программный продукт позволяет отвлечься от реальной жизни и расслабиться.

1. **Тестирование**
   1. **Тесты на использование**

В ходе разработки программного продукта были составлены тесты, которые необходимо выполнить в дальнейшем. Тесты составлены таким образом, чтобы предусмотреть максимальное количество возможных действий.

Таблица 6 – проведение тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название теста | Действия | Исходная информация | Ожидаемая информация |
| 1 | 2 | 3 | 4 | 5 |
| Т1 | Проверка открытия игры | Открытие приложения | - | Открытие приложения |
| Т2 | Проверка закрытия игры | Закрытие приложения | - | Закрытие приложения |
| Т3 | Проверка кнопок, переходящих на другие сцены | Переход на следующую сцену | - | Переход и открытие следующей сцены |
| Т4 | Проверка ввода имени персонажа | Пользователь вводит имя в поле ввода | Имя персонажа | Пользователь ввёл имя персонажа, и оно сохранилось |
| Т5 | Проверка музыки | Музыка играет на всех сценах | - | При входе в игру включается музыка |
| Т6 | Проверка звука кнопок | При нажатии какой-либо кнопки, воспроизводится звук | - | Звук, который воспроизводится только при нажатии на кнопку |
| Т7 | Проверка громкости музыки | Редактирование музыки с помощью slider | - | Игрок перетягивает slider вверх-вниз, тем самым меняя громкость музыки |
| Т9 | Проверка «Загрузки» игры | После того, как пользователь нажал «Создать новый мир», происходит загрузка мира | - | Загружается новый мир |
| Т10 | Проверка Логотипа | Запуск .exe файла | - | Запуск файла и отображение логотипа на экране |
| Т11 | Проверка передвижение персонажа | Передвигается ли персонаж с помощью клавиш WASD | - | Персонаж передвигается по периметру |

Продолжение таблицы 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Т12 | Проверка анимации | Меняется ли анимация с покоя на ходьбу при передвижении | - | Анимация персонажа на локации |
| Т13 | Проверка ограничения пространства | Может ли персонаж выходить за предел ограниченной зоны | - | Персонаж находится в пределах ограниченного пространства |
| Т14 | Проверка поворота персонажа по курсору | Персонаж поворачивается в ту сторону куда игрок направляет мышь | - | Персонаж поворачивается в ту сторону куда игрок направляет мышь |

* 1. **Отчёт о результатах тестирования**

Таблица 7 – результаты тестов

|  |  |
| --- | --- |
| № | Статус |
| Т1 | Выполнено успешно |
| Т2 | Выполнено успешно |
| Т3 | Выполнено успешно |
| Т4 | Выполнено успешно |
| Т5 | Выполнено успешно |
| Т6 | Выполнено успешно |
| Т7 | Выполнено успешно |
| Т8 | Выполнено успешно |
| Т9 | Выполнено успешно |
| Т10 | Выполнено успешно |
| Т11 | Выполнено успешно |
| Т12 | Выполнено успешно |
| Т13 | Выполнено успешно |
| Т14 | Выполнено успешно |

1. **Руководство пользователя**
   1. **Общие сведения**

Наименованием конфигурации является «Virtual World» для людей старше 7+. Назначение – программный продукт разрабатывается для людей старше 7+ для того, чтобы они могли расслабиться, поиграть и отвлечься от реального мира.

* 1. **Запуск веб-ресурса**

В самом начале при запуске программы, нас встречает Заставка как показано на рисунке 3.

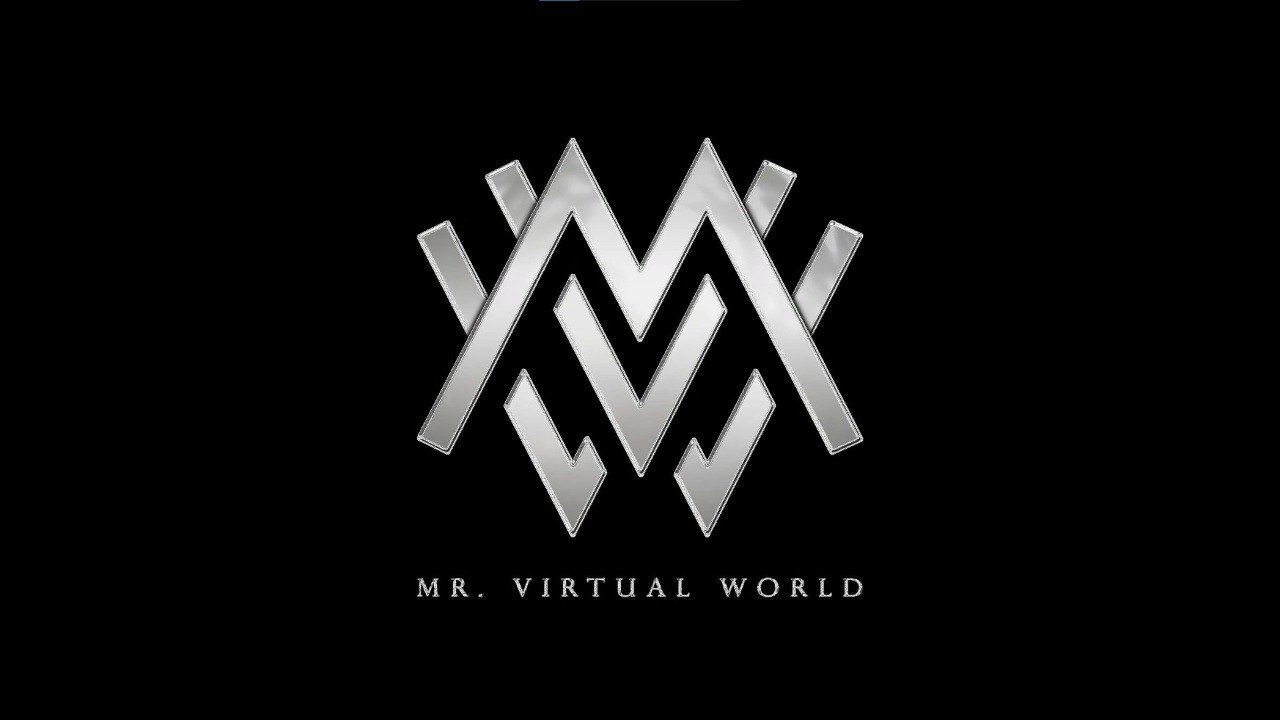


Рисунок 3 - Заставка

* 1. **Загрузка Веб-ресурса**

После запуска программы происходит загрузка игры, как показано на рисунке 4.



Рисунок 4 – Загрузка

* 1. **Инструкция по работе с конфигурацией**

После загрузки игры, игрока встречает начальный экран, как показано на рисунке 5.

Рисунок 5 – Начальный экран

После того, как игрок нажмёт кнопку «Создать новый мир», он перейдёт на сцену «Создания персонажа», как показано на рисунке 6.

Рисунок 6 – Создание персонажа

После того, как игрок создал своего персонажа, он переходит на локацию «Дом», как показано на рисунке 7.



Рисунок 7 – Локация «Дом»

Также игрок может перейти на карту города, как показано на рисунке 8.



Рисунок 8 – Карта города

С карты города, игрок может попасть не только в «Дом», но и на локацию «Магазин», как показано на рисунке 9.



Рисунок 9 – Локация «Магазин»

Также в данных локациях присутствует кнопка перехода на главное меню, где игрок может для себя что-то выбрать и посмотреть, как показано на рисунке 10.



Рисунок 10 – Главное меню

В главном меню находятся «Достижения», в которых игрок может посмотреть какие достижения он получил, как показано на рисунке 11.

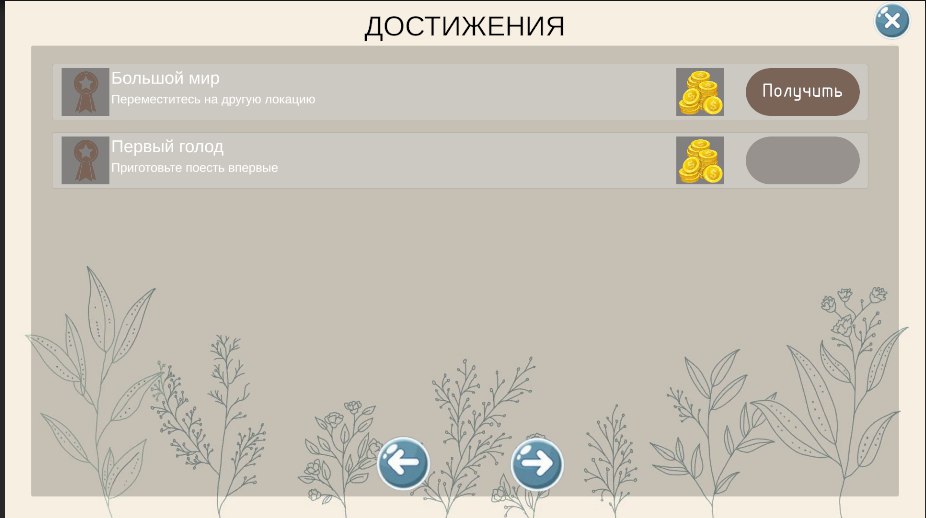


Рисунок 11 – Достижения

**Заключение**

Целью данного проекта являлась разработка игрового приложения “Virtual Wotld”.

В ходе реализации поставленной задачи был укреплён пройденный курс программирования в среде Unity, а также получено много дополнительной информации о её возможностях.

Следует также учесть, что в поставленной задачи был реализован простой интерфейс, который позволяет использовать приложение пользователю, не обладающему дополнительными знаниями ЭВМ.

После тщательного тестирования приложения были выявлены некоторые недоработки, которые были частично исправлены на стадии проектирования, и полностью исключены на стадии тестирования программы. В целом при реализации программы, были выполнены все условия, перечисленные в предыдущих разделах пояснительной записки.

Таким образом, можно сказать, что программа была реализована вполне успешно.

**Список используемых источников**

1. Онлайн-обучение по программированию на Unity [Электронный ресурс]. – Режим доступа: [ТОП-20 курсов Unity - рейтинг обучения в Сравни ру (sravni.ru)](https://www.sravni.ru/kursy/programmirovanie-unity/?utm_source=yandex&utm_medium=cpc&utm_term=%D0%B8%D0%B7%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20unity&position_type=premium&utm_campaign=sravni_education_search_world_programmirovanie_102651813&utm_placement=none_%7bdevice%7d&utm_content=k50id--0100000048871477215_48871477215--cid--102651813--gid--5351985789--aid--15511470803--adp--no--pos--premium1--src--search_none--dvc--desktop--%D0%93%D1%80%D0%BE%D0%B4%D0%BD%D0%BE_10274_15511470803&yclid=6858385937608736767) – Дата доступа: 21.05.2023
2. Геймдев с нуля [Электронный ресурс]. – Режим доступа: [Онлайн академия Unity разработки (ctpk.academy)](https://ctpk.academy/?yclid=11001245680214999039) – Дата доступа: 25.05.2023
3. Учись разрабатывать игры без программирования [Электронный ресурс]. – Режим доступа: [Учитесь разрабатывать игры без программирования | Unity](https://unity.com/ru/learn) – Дата доступа: 30.05.2023
4. Learn game development Unity [Элекронный ресурс]. – Режим доступа: [Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn](https://learn.unity.com/?ref=dtf.ru) – Дата доступа: 02.06.2023
5. Controlling Animation [Электронный ресурс]. – Режим доступа: [Controlling Animation - Unity Learn](https://learn.unity.com/tutorial/controlling-animation) – Дата доступа 07.06.2023
6. Анимация в Unity [Электронный ресурс]. – Режим доступа: [Анимация в Unity / Skillbox Media](https://skillbox.ru/media/gamedev/kak_sdelat_2d_animatsiyu_v_unity/) – Дата допуска 15.06.2023

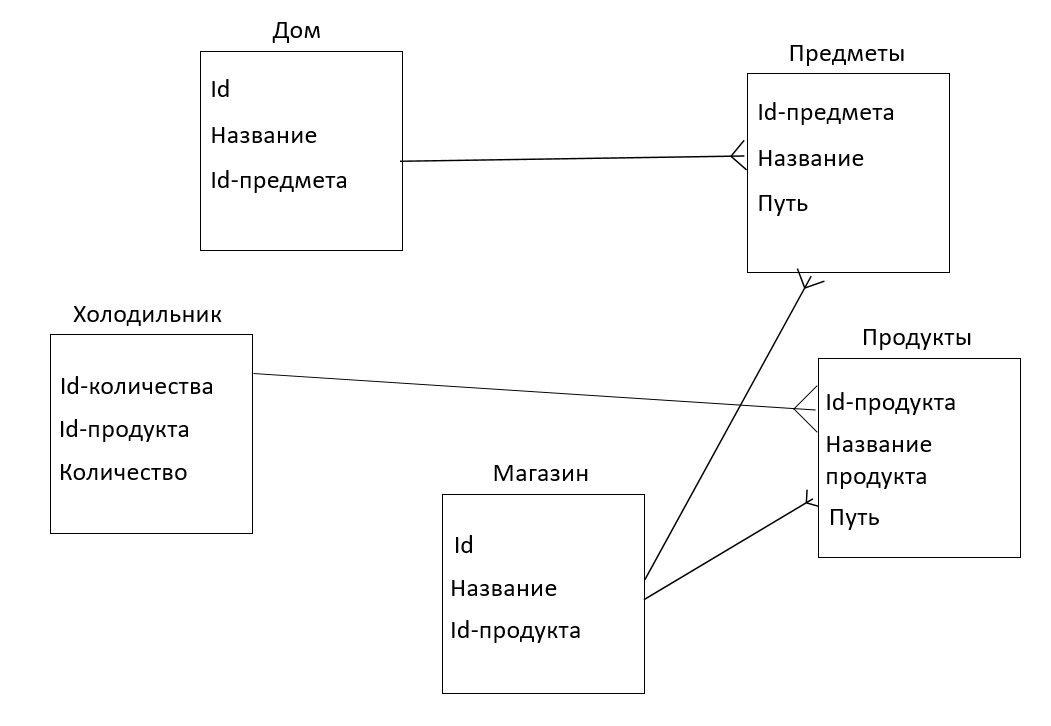
**Приложение А**

Структурное проектирование ПО

Рисунок А.1 – Структурное проектирование

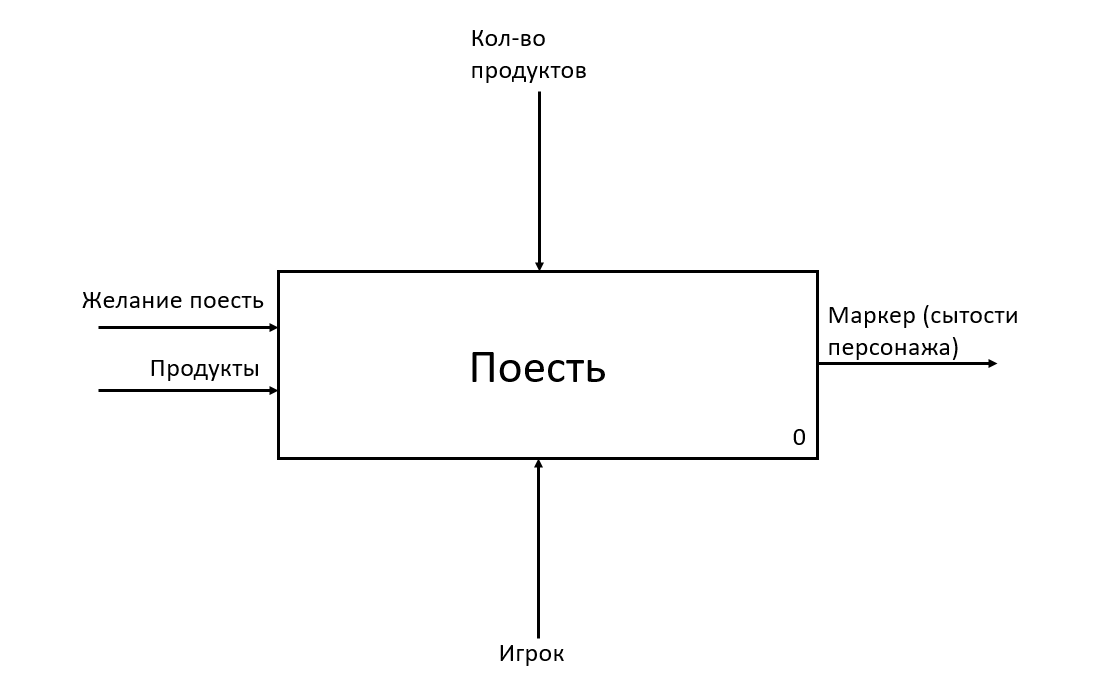
**Приложение Б**

Модель данных

Рисунок Б.1 – Модель данных

**Приложение В**

Функциональная модель

Рисунок В.1 – Моделирование бизнес-процессов

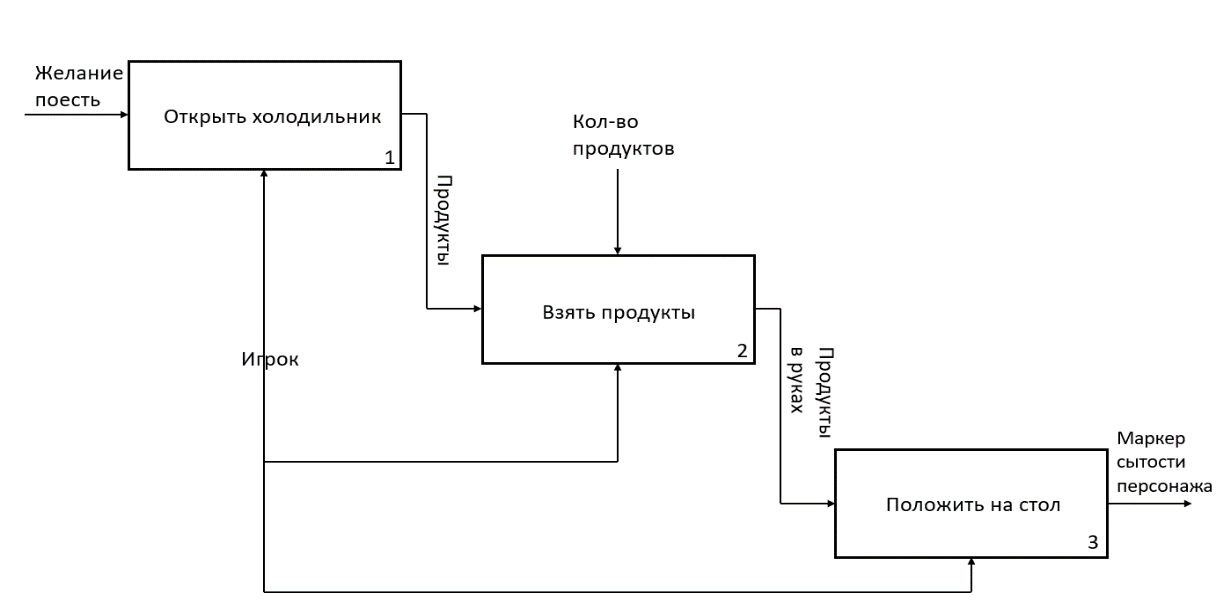


Рисунок В.2 – Моделирование бизнес-процессов

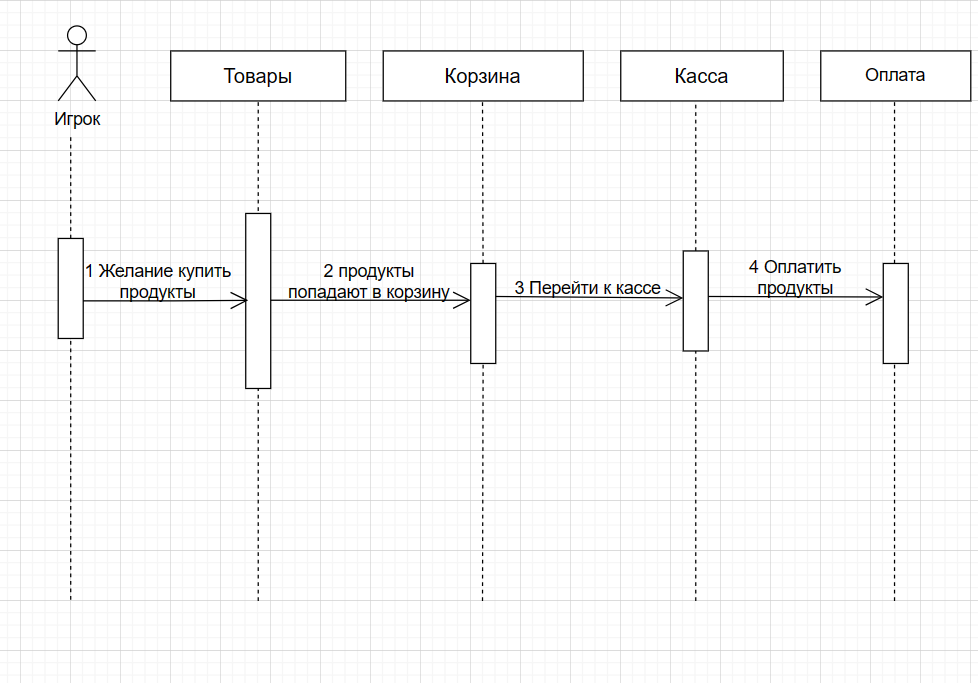
**Приложение Г**

Диаграмма вариантов использования

Рисунок Г.1 – Диаграмма вариантов использования

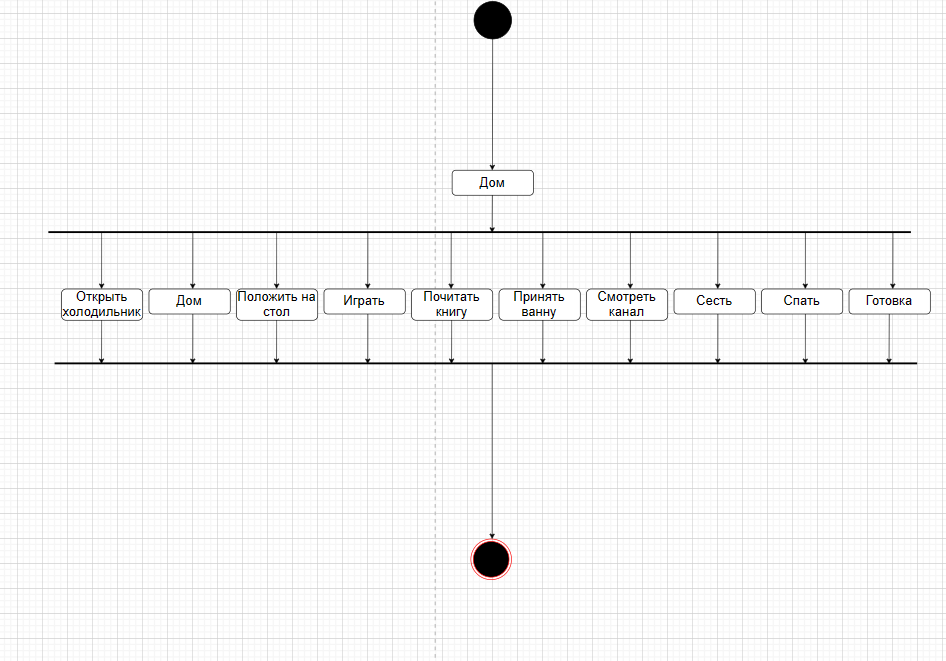
**Приложение Д**

Диаграмма последовательности

****Рисунок Д.1 – Диаграмма последовательности

**Приложение Е**

Диаграмма деятельности

****Рисунок Е.1 – Диаграмма деятельности

**Приложение Ж**

Диаграмма классов

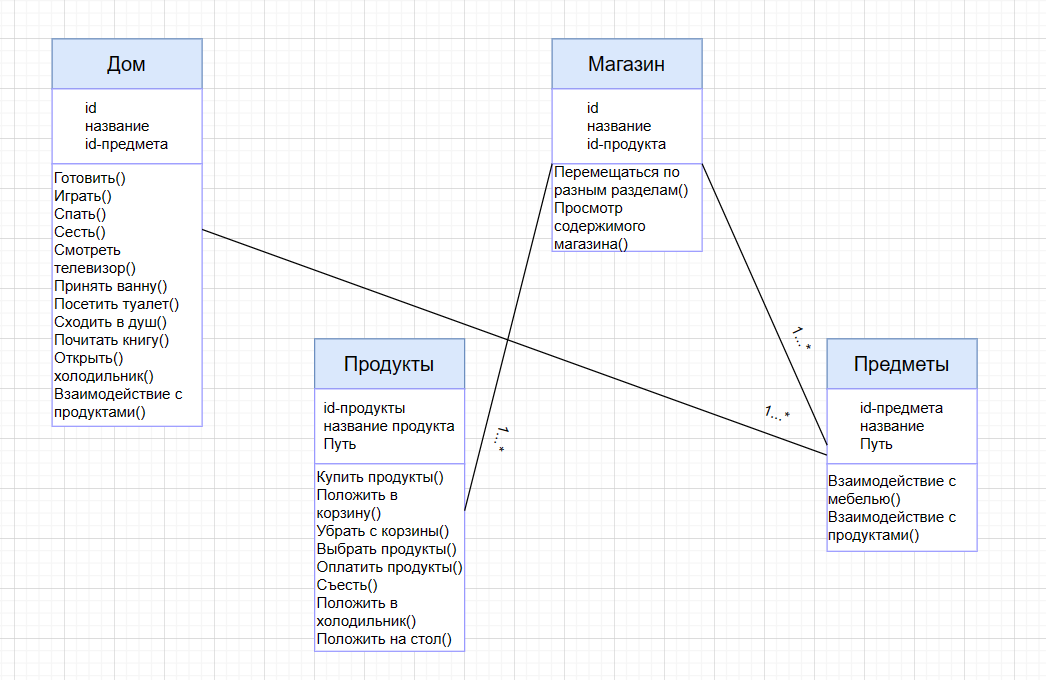
****

Рисунок Ж.1 – Диаграмма классов

**Приложение К**

Диаграмма объектов

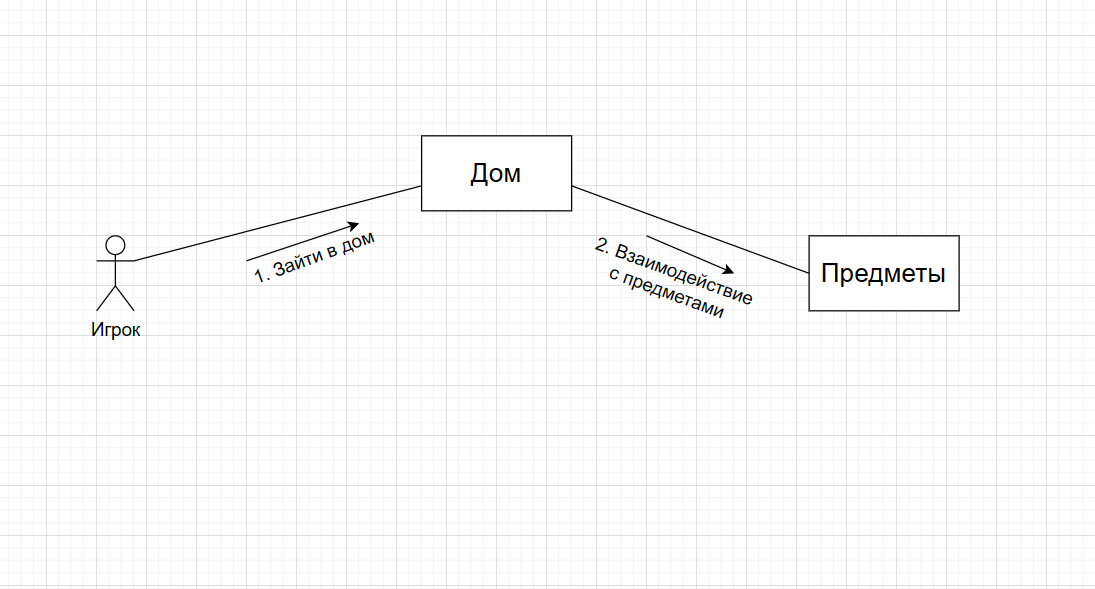
****

Рисунок К.1 – Диаграмма объектов

**Приложение Л**

UX/UI макеты

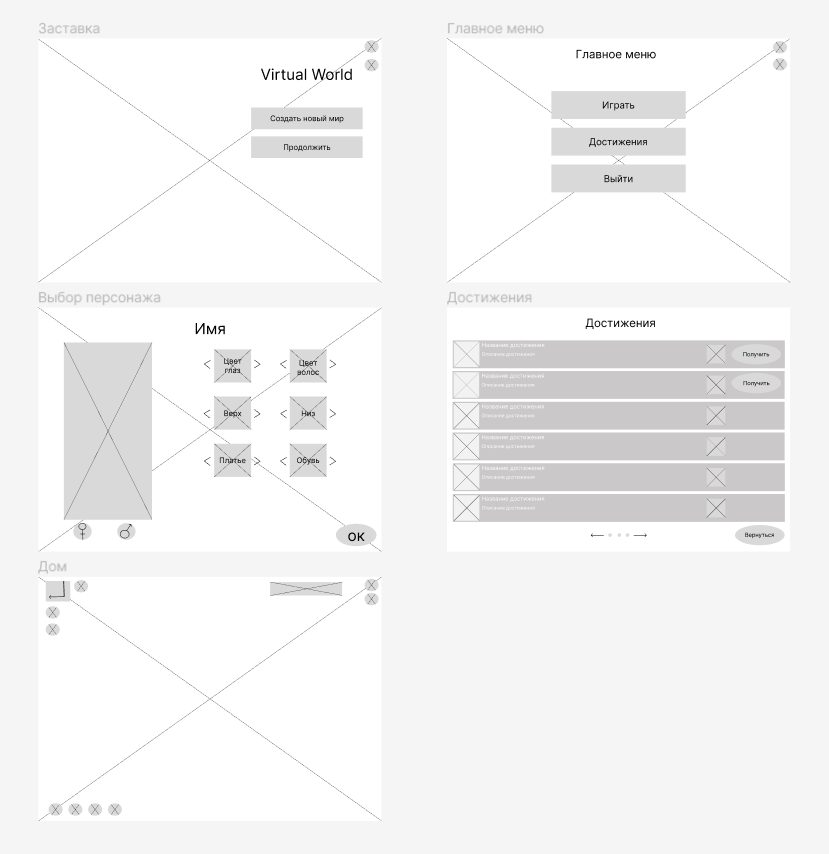
Рисунок Л.1 – UX макет

Рисунок Л.2 – UI макет

**Приложение М**

Листинг программы

//------------------------------------------------------------------------------

// <auto-generated>

// This code was auto-generated by com.unity.inputsystem:InputActionCodeGenerator

// version 1.7.0 // from Assets/Script/PlayerInputActions.inputactions

//

// Changes to this file may cause incorrect behavior and will be lost if

// the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine.InputSystem;

using UnityEngine.InputSystem.Utilities;

public partial class [@PlayerInputActions](https://t.me/PlayerInputActions): IInputActionCollection2, IDisposable

{ public InputActionAsset asset { get; }

public [@PlayerInputActions](https://t.me/PlayerInputActions)()

{

asset = InputActionAsset.FromJson(@"{

""name"": ""PlayerInputActions"",

""maps"": [ { ""name"": ""player"",

""id"": ""38a45417-a511-4739-907f-0796c4e370ec"",

""actions"": [

{

""name"": ""Move"",

""type"": ""Value"",

""id"": ""aff2bf28-b93a-4a46-bc9f-5bf6d116ccf8"",

""expectedControlType"": ""Vector2"",

""processors"": """",

""interactions"": """",

""initialStateCheck"": true

}

],

""bindings"": [

{

""name"": ""WASD"",

""id"": ""fe776725-aed0-4e96-aa52-2c0cda4aaa6f"",

""path"": ""2DVector"",

""interactions"": """",

""processors"": """",

""groups"": """",

""action"": ""Move"",

""isComposite"": true,

""isPartOfComposite"": false

},

{

""name"": ""up"",

""id"": ""ea2b8c5b-2a9e-44eb-b69a-f530b43033f3"",

""path"": ""<Keyboard>/w"",

""interactions"": """",

""processors"": """",

""groups"": """",

""action"": ""Move"",

""isComposite"": false,

""isPartOfComposite"": true

},

{

""name"": ""down"",

""id"": ""7cfa8c5c-1e32-4e33-a75e-650a55ca6f74"",

""path"": ""<Keyboard>/s"",

""interactions"": """",

""processors"": """",

""groups"": """",

""action"": ""Move"",

""isComposite"": false,

""isPartOfComposite"": true

},

{

""name"": ""left"",

""id"": ""c69e9f3d-3b56-4b03-8688-ec171089e12c"",

""path"": ""<Keyboard>/a"",

""interactions"": """",

""processors"": """",

""groups"": """",

""action"": ""Move"",

""isComposite"": false,

""isPartOfComposite"": true

},

{

""name"": ""right"",

""id"": ""8f10a885-a6ba-4f3d-9c44-4ff05c0c4e1e"",

""path"": ""<Keyboard>/d"",

""interactions"": """",

""processors"": """",

""groups"": """",

""action"": ""Move"",

""isComposite"": false,

""isPartOfComposite"": true

}

]

}

],

""controlSchemes"": []

}");

// player

m\_player = asset.FindActionMap("player", throwIfNotFound: true); m\_player\_Move = m\_player.FindAction("Move", throwIfNotFound: true);

}

public void Dispose()

{ UnityEngine.Object.Destroy(asset);

}

public InputBinding? bindingMask

{

get => asset.bindingM

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class InTown : MonoBehaviour

{

public void OpenMenu()

{

SceneManager.LoadScene("Town");

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Scenes : MonoBehaviour

{

public void OpenMenu()

{

SceneManager.LoadScene("Dost");

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class HoverEffect : MonoBehaviour

{

private Vector3 originalPosition;

public float hoverHeight = 2f;

void Start()

{

// Сохраняем изначальную позицию объекта

originalPosition = transform.position;

}

void OnMouseEnter()

{

// При наведении мыши поднимаем объект на указанную высоту

Vector3 hoverPosition = originalPosition + new Vector3(0f, hoverHeight, 0f);

transform.position = hoverPosition;

}

void OnMouseExit()

{

// При уходе мыши возвращаем объект на изначальную позицию

transform.position = originalPosition;

}

}

using UnityEngine;

using System.Collections;

public class SmoothBlinking : MonoBehaviour

{

public float blinkDuration = 1.0f; // Длительность мигания в секундах

public float startDelay = 0.5f; // Задержка перед началом мигания

private bool isBlinking = false;

private void Start()

{

// Запускаем метод мигания с задержкой

Invoke("StartBlinking", startDelay);

}

private void StartBlinking()

{

// Если уже не мигаем, запускаем корутину мигания

if (!isBlinking)

{

isBlinking = true;

StartCoroutine(Blink());

}

}

private IEnumerator Blink()

{

// Плавное затухание

float timer = 0f;

while (timer < blinkDuration)

{

float alpha = Mathf.Lerp(1f, 0f, timer / blinkDuration);

SetAlpha(alpha);

timer += Time.deltaTime;

yield return null;

}

// Пауза между состояниями

yield return new WaitForSeconds(0.5f);

// Плавное появление

timer = 0f;

while (timer < blinkDuration)

{

float alpha = Mathf.Lerp(0f, 1f, timer / blinkDuration);

SetAlpha(alpha);

timer += Time.deltaTime;

yield return null;

}

// Переход к новому циклу мигания

isBlinking = false;

StartBlinking();

}

private void SetAlpha(float alpha)

{

// Установка значения альфа-канала в компоненте

Color color = GetComponent<SpriteRenderer>().color;

color.a = alpha;

GetComponent<SpriteRenderer>().color = color;

}

}

using System.Collections;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class LoadingScreen : MonoBehaviour

{

public string sceneToLoad;

public GameObject loadScreen;

public Slider bar; public void Load()

{

loadScreen.SetActive(true);

//SceneManager .LoadScene("Menu");

StartCoroutine(LoadAsync());

}

IEnumerator LoadAsync()

{

AsyncOperation asyncLoad =

SceneManager.LoadSceneAsync(sceneToLoad);

while (!asyncLoad.isDone )

{

bar.value = asyncLoad.progress;

yield return null;

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CloseApp : MonoBehaviour

{

public void ExitGame()

{

Application.Quit();

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Play : MonoBehaviour

{

public void OpenMenu()

{

SceneManager.LoadScene("Передвижения");

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Scenes1 : MonoBehaviour

{

public void OpenMenu()

{

SceneManager.LoadScene("Menu");

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Colliderch : MonoBehaviour

{ public EdgeCollider2D boundaryCollider;

void Update()

{

// Получаем текущую позицию персонажа

Vector2 currentPosition = transform.position;

// Проверяем, находится ли текущая позиция внутри границ

if (!IsInsideBoundary(currentPosition))

{

// Если персонаж выходит за границы, корректируем его позицию

Vector2 newPosition =

GetClosestPointInsideBoundary(currentPosition);

transform.position = new Vector2(newPosition.x, newPosition.y);

}

}

bool IsInsideBoundary(Vector2 position)

{

// Проверяем, находится ли позиция внутри границ

return boundaryCollider.bounds.Contains(position);

}

Vector2 GetClosestPointInsideBoundary(Vector2 position)

{

// Получаем ближайшую точку внутри границ

return boundaryCollider.bounds.ClosestPoint(position);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class btnFX : MonoBehaviour

{

public AudioSource myFx;

public AudioClip hoverFx;

public AudioClip clickFx;

public void HoverSound()

{

myFx.PlayOneShot(hoverFx);

}

public void ClickSound()

{

myFx.PlayOneShot(clickFx);

}

} using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class btnFy : MonoBehaviour

{

public AudioSource myFy;

public AudioClip hoverFy;

public AudioClip clickFy;

public void HoverSound()

{

myFy.PlayOneShot(hoverFy);

}

public void ClickSound()

{

myFy.PlayOneShot(clickFy);

}

}  
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class VolumeValue : MonoBehaviour

{

private AudioSource audioSrc;

private float musicVolume = 1f;

// Start is called before the first frame update

void Start()

{

audioSrc = GetComponent<AudioSource>();

}

// Update is called once per frame

void Update()

{

audioSrc.volume = musicVolume;

}

public void SetVolume(float vol)

{

musicVolume = vol;

}

} using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Имяперса : MonoBehaviour

{

[SerializeField] private int Score;

[SerializeField] private int Bonus = 1;

[SerializeField] private int PriceItem = 10;

[SerializeField] private GameObject PanelName;

[SerializeField] private InputField InputName;

[SerializeField] private Text TextScore;

[SerializeField] private Text TextName;

private void Start()

{

TextScore.text = Score + "$";

}

public void OnClick()

{

Score += Bonus;

TextScore.text = Score + "$";

}

public void SetName()

{

if (InputName.text == "")

{

Debug.Log("Ошибка");

}

else

{

PanelName.SetActive(false);

TextName.text = " " + InputName.text;

Debug.Log("Успешно");

}

}

} using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ChangeImage : MonoBehaviour

{

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

}

}