МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2023 г.

Разработка приложения виртуальной реальности для интерактивного обучения техническому обслуживанию дорожно-строительной техники

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

ЮУРГУ-090301.2024.405 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____ Ю.Г. Плаксина
«___»_____2024 г.

Автор работы,
студент группы КЭ-405
_____К.А. Житков
«___»_____2024 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Житков Кирилл Алексеевич
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

Тема работы: «Разработка приложения виртуальной реальности для интерактивного обучения техническому обслуживанию дорожно-строительной техники».

Срок сдачи студентом законченной работы: 01 июня 2024 г.

Исходные данные к работе:

1. VR-гарнитура Oculus Quest 2, характеристики:
   - Разрешение на каждый глаз 1832x1920 точек, общее разрешение 3664x1920 точек;
   - Частота обновления экрана 90 Гц;
   - Угол обзора 100°;
   - Объем оперативной памяти 6 ГБ;
   - Объем встроенной памяти 128 ГБ;
   - Поддерживаемые беспроводные интерфейсы: Bluetooth, Wi-Fi;
   - Разъемы на гарнитуре: USB Type-C, 3.5 mm jack.

- Встроенные датчики: акселерометр, гироскоп, датчик отслеживания, датчик приближения.

2. Нефункциональные требования:
   1.1. Использование технологии OpenXR.
   1.2. Минимальные системные требования к персональному компьютеру:
      - Операционная система: Windows 10;
      - Процессор: Intel i5-4590 / AMD Ryzen 5 1500X или лучше;
      - Оперативная память: 8 ГБ или больше;
      - Видеокарта: NVIDIA GeForce GTX 970 / AMD Radeon 400 серии или лучше, поддерживающая DirectX 11;
      - USB 3.0 порт или кабельное соединение с Wi-Fi роутером, работающим в диапазоне частот 5 ГГц.

Перечень подлежащих разработке вопросов:

1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.
2. Проектирование тренажера виртуальной реальности для интерактивного обучения техническому обслуживанию дорожно-строительной техники.
3. Программная реализация приложения.
4. Тестирование разработанного приложения.

Дата выдачи задания: 1 декабря 2023 г.

Руководитель работы _____/ *Ю.Г. Плаксина*/

Студент _____/ *К.А. Житков* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2024	
Проектирование приложения	15.03.2024	
Программная реализация приложения	01.05.2024	
Тестирование разработанного приложения	15.05.2024	
Компоновка текста работы и сдача на нормоконтроль	27.05.2024	
Подготовка презентации и доклада	30.05.2024	

Руководитель работы _____/ *Ю.Г. Плаксина* /

Студент _____/ *К.А. Житков* /

АННОТАЦИЯ

К.А. Житков. Разработка приложения виртуальной реальности для интерактивного обучения техническому обслуживанию дорожно-строительной техники. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 69с., 20 ил., библиогр. список –21 наим.

Выпускная квалификационная работа посвящена разработке приложения виртуальной реальности для интерактивного обучения техническому обслуживанию дорожно-строительной техники с целью его дальнейшего использования в процессе обучения специалистов по ремонту и техническому обслуживанию дорожно-строительной техники.

В первом разделе работы проведен анализ предметной области, рассмотрены современные аналоги разрабатываемого приложения.

Во втором разделе обоснованы функциональные и нефункциональные требования к приложению. Одним из требований является использование технологии OpenXR, которая обеспечивает поддержку приложением широкого спектра гарнитур виртуальной реальности. Были рассмотрены возможные инструменты реализации и из них были выбраны следующие: игровой движок Unity, плагин XR Interaction Toolkit и встроенный в игровой движок Unity инструмент для создания графического пользовательского интерфейса uGUI. Проведено проектирование систем, компонентов приложения и его пользовательского графического интерфейса.

В третьем разделе описана реализация систем приложения и его пользовательского графического интерфейса. Приведено описание демонстрационной обучающей сцены, созданной в процессе реализации приложения.

Четвертый раздел посвящен функциональному и нефункциональному тестированию разработанного приложения.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 8
1. АНАЛИТИЧЕСКИЙ ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ, НОРМАТИВНОЙ И МЕТОДИЧЕСКОЙ ЛИТЕРАТУРЫ ПО ТЕМАТИКЕ РАБОТЫ 10
1.1 ОБЗОР АНАЛОГОВ 13
ВЫВОД ПО РАЗДЕЛУ 1 18
2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 19
2.1. ТРЕБОВАНИЯ К ПРЛОЖЕНИЮ 19
2.1.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ 19
2.1.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ 20
2.2 ВЫБОР ИНСТРУМЕНТОВ РАЗРАБОТКИ 22
2.2.1 ИГРОВОЙ ДВИЖОК 22
2.2.2. XR INTERACTION TOOLKIT 23
2.2.3. ИНСТРУМЕНТЫ СОЗДАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 23
2.3. ПОСТРОЕНИЕ АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ, ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ И СИСТЕМ ПРИЛОЖЕНИЯ, СВЯЗЕЙ МЕЖДУ НИМИ 24
2.3.1. ПРОЕКТИРОВАНИЕ КОМПОНЕНТА ВИРТУАЛЬНОЙ ДЕТАЛИ 26
2.3.2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТА ОБРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА 27
2.3.3. ПРОЕКТИРОВАНИЕ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 27
2.3.3.1. ПРОЕКТИРОВАНИЕ ГЛАВНОГО МЕНЮ 27
2.3.3.2 ПРОЕКТИРОВАНИЕ МЕНЮ ВЫБОРА СЦЕНЫ 28
2.3.3.3. ПРОЕКТИРОВАНИЕ МЕНЮ СПРАВОЧНИКА 29
2.3.3.4. ПРОЕКТИРОВАНИЕ СПИСКА НЕОБХОДИМЫХ ДЕЙСТВИЙ 30
2.3.3.5. ПРОЕКТИРОВАНИЕ ОКНА ПОДСКАЗОК 31
ВЫВОД ПО РАЗДЕЛУ 2 32
3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 33
3.1. СРЕДСТВА РЕАЛИЗАЦИИ 33
3.2. РЕАЛИЗАЦИЯ АРХИТЕКТУРЫ, УПРАВЛЯЕМОЙ СОБЫТИЯМИ 35
3.2. РЕАЛИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ ПОЛЬЗОВАТЕЛЯ С ВИРТУАЛЬНЫМИ ДЕТАЛЯМИ 36
3.2.1. РЕАЛИЗАЦИЯ КОМПОНЕНТА ВИРТУАЛЬНОЙ ДЕТАЛИ 37
3.2.2. РЕАЛИЗАЦИЯ КОМПОНЕНТА ОБРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА 40
3.3. РЕАЛИЗАЦИЯ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 40
3.3.1.РЕАЛИЗАЦИЯ ГЛАВНОГО МЕНЮ, МЕНЮ ВЫБОРА СЦЕНЫ И МЕНЮ СПРАВОЧНИКА 40
3.3.2. РЕАЛИЗАЦИЯ СПИСКА НЕОБХОДИМЫХ ДЕЙСТВИЙ 43
3.3.3. РЕАЛИЗАЦИЯ ОКНА ПОДСКАЗОК 44

3.4. ДОРАБОТКА МГНОВЕННОГО ПЕРЕМЕЩЕНИЯ ПО СЦЕНЕ... 45
3.5. СЦЕНА РАЗБОРКИ БУЛЬДОЗЕРА D12 45
ВЫВОД ПО РАЗДЕЛУ 3 46
4. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ 47
4.1. ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ 47
4.2. НЕФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ 51
ВЫВОД ПО РАЗДЕЛУ 4 52
ЗАКЛЮЧЕНИЕ 53
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 54
ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ АРХИТЕКТУРЫ, УПРАВЛЯЕМОЙ СОБЫТИЯМИ 57
ПРИЛОЖЕНИЕ Б ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ КОМПОНЕНТА ВИРТУАЛЬНОЙ ДЕТАЛИ 59
ПРИЛОЖЕНИЕ В ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ КОМПОНЕНТА ОБРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА 63
ПРИЛОЖЕНИЕ Г ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ ГЛАВНОГО МЕНЮ, МЕНЮ ВЫБОРА СЦЕНЫ И МЕНЮ СПРАВОЧНИКА 64
ПРИЛОЖЕНИЕ Д ИСХОДНЫЙ КОД КОМПОНЕНТА ACTIONSLISTELEMENT 66
ПРИЛОЖЕНИЕ Е ИСХОДНЫЙ КОД КОМПОНЕНТА TOOLTIPCONTROLLER 67
ПРИЛОЖЕНИЕ Ж ИСХОДНЫЙ КОД КОМПОНЕНТА TELEPORTATIONCONTROLLER 68

ВВЕДЕНИЕ

В данной работе рассматривается создание тренажера виртуальной реальности для интерактивного обучения техническому обслуживанию дорожно-строительной техники.

Первые прототипы устройств виртуальной реальности, которые, как и большая часть электронных устройств того времени, были весьма громоздкими и очень дорогими в производстве. Дальнейшее развитие электроники за последние полвека позволило создать компактные, но при этом весьма производительные мобильные устройства, такие как современные смартфоны, без которых уже сложно представить жизнь современного человека. По этой же причине за последние 20 лет стремительно появился и вырос рынок гарнитур виртуальной и дополненной реальностей. Так, по данным немецкой статистической компании Statista рынок устройств дополненной, виртуальной и смешанной реальностей стоит около 27.96 миллиардов долларов США, из которых 21.83 миллиардов долларов составляет рынок именно устройств виртуальной реальности, и непрерывно растет примерно на 15% в год [1]. На сегодняшний день технологии виртуальной реальности могут себе позволить не только крупные предприятия, организации и очень зажиточные люди, но и обычный рабочий, т.к. цены на мобильные гарнитуры колеблется в районе 50-400 долларов, а стоимость более производительных стационарных устройств начинается от 400-500 долларов.

Под терминами дополненной, виртуальной, смешанной и расширенной реальностей подразумевается следующее:

1. Виртуальная реальность (VR) – это созданная с помощью современных технических средств симуляция мира, которая передается человеку через его ощущения (в основном через зрение и слух, реже через осязание и другие чувства).
2. Дополненная реальность (AR) – дополнение визуальной части реального мира различными виртуальными визуальными элементами и данными с

различных датчиков и оборудования при помощи современных технических средств.

3. Смешанная реальность (MR) – она же гибридная реальность, является слиянием реального и виртуального мира. В ней виртуальные и реальные объекты сосуществуют и взаимодействуют друг с другом в реальном времени. Включает в себя как дополненную, так и виртуальную реальность.
4. Расширенная реальность (XR) – общее обозначение для виртуальной, дополненной и смешанной реальности.

Целью выпускной квалификационной работы является создание приложения виртуальной реальности для интерактивного обучения техническому обслуживанию дорожно-строительной техники.

Для достижения поставленной цели необходимо:

- провести аналитический обзор научно-технической, нормативной и методической литературы по теме;
- спроектировать приложение;
- реализовать приложение;
- провести тестирование приложения.

Актуальность данной работы заключается в том, что применение технологий виртуальной реальности в учебном процессе позволяет сэкономить средства, повысить безопасность обучающихся, воссоздать редкие, опасны или дорогие ситуации и оборудование в виртуальной реальности, повысить вовлечение ученика в процесс обучения.

1. АНАЛИТИЧЕСКИЙ ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ, НОРМАТИВНОЙ И МЕТОДИЧЕСКОЙ ЛИТЕРАТУРЫ ПО ТЕМАТИКЕ РАБОТЫ

Основными преимуществами использования технологий виртуальной реальности в учебном процессе являются:

1. Интерактивность и иммерсивность – обучающийся полностью погружается в виртуальную среду и может самостоятельно взаимодействовать с большинством ее объектов весьма реалистичным способом, что позволяет больше сконцентрироваться на обучении, самостоятельно изучать представленные в симуляции объекты и процессы. Трехмерные модели объектов, которые можно осмотреть с любой стороны. Все это улучшает наглядность материала и позволяет лучше усвоить информацию [2, 3].
2. Безопасность – виртуальные объекты не могут нанести человеку вреда, что нельзя сказать о реальных станках, технике, инструментах и многих других вещах. При замещении части практических занятий различными VR-тренажерами можно повысить безопасность обучения в целом. Также, в виртуальной реальности можно создавать напрямую опасные для жизни и здоровья человека ситуации.
3. Возможность представить большинство объектов и их функционал, создать любые ситуации – некоторое оборудование или обучающие ситуации слишком дороги, чтобы их могли себе позволить образовательные учреждения и другие организации. В виртуальной реальности же можно воссоздать большинство объектов, причем затраты на создание их в VR будут многократно меньше, чем создание реальных образцов. Многие же ситуации, которые могут произойти в процессе работы нецелесообразно создавать в реальном мире ввиду их дороговизны или трудоемкости, либо же такие ситуации чрезвычайно редки. Используя современные технические средства их можно воссоздать в виде приближенном к реальному единожды и многократно их воспроизвести их в симуляции [3].

4. Данные о действиях пользователя и результатах обучения могут быть сохранены программой, что позволяет использовать их для дальнейшего анализа, исследования или проведения работы над ошибками [2].

Опыт применения гарнитур на практике выявил и некоторые их недостатки:

1. Головные боли и головокружения при длительных сеансах – отрисовка изображения устройством происходит не мгновенно и случаются задержки, которые пользователь может даже и не заметить, из-за чего мозг вынужден иногда самостоятельно достраивать изображение, что вызывает головную боль. Вследствие сенсорного несоответствия (в реальном мире человек может стоять на месте, а в симуляции перемещаться при этом), особенно при плавном перемещении человека в виртуальной среде, возникает эффект, схожий с «морской болезнью» [3].
2. Возможная повышенная нагрузка на зрение – современные гарнитуры виртуальной реальности устроены так, что их экраны расположены очень близко к глазам, из-за чего при длительном и систематическом их использовании могут возникнуть проблемы со зрением. На данный момент влияние гарнитур виртуальной реальности на зрение исследовано не полностью. Также конструкция гарнитур не учитывает на 100% положение глаз или особенностей зрения, что может вызывать дискомфорт.
3. Стационарные гарнитуры или мобильные гарнитуры, подключенные при помощи кабеля к другим устройствам, стесняют движения пользователя проводами [3].
4. Пользователь не видит реального пространства вокруг. Человек может травмироваться об окружающие объекты, так как попросту их не видит. Эта проблема частично решена в более современных VR-устройствах, в которых при приближении к границе «игровой» области, обозначенной заранее, включаются камеры, расположенные на шлеме, показывая окружение и предупреждая пользователя. Также существуют специальные устройства, представляющие из себя беговую дорожку и кольцо для фиксации пользователя.

5. Внедрение в образовательные учреждения технологий виртуальной реальности несет дополнительную финансовую нагрузку на их закупку и обслуживание. Также нужно будет обучать преподавателей работе с гарнитурами виртуальной реальности и эффективному применению тренажеров и симуляций в учебном процессе [4, 5].

Все преимущества VR-тренажеров, несмотря на имеющиеся недостатки, сделали их популярными в программах обучения работников МЧС, медиков, в частности большой спрос на тренажеры имеется у хирургов, пилотов, операторов и водителей различной техники, в инженерных специальностях. Также весьма распространены симуляции по технике безопасности, особенно для работников, чья деятельность связана с переработкой, добычей и транспортировкой горючих и/или взрывоопасных веществ.

Доступность устройств виртуальной и дополненной реальности позволила поставить вопрос об исследовании эффективности использования устройств расширенной реальности в образовательном процессе в учреждениях высшего, среднего и даже общего образования.

На текущий момент был проведен целый ряд исследований, посвященных влиянию использования технологий виртуальной и дополненной реальностей в образовательном процессе на результаты обучения людей. Эксперимент, проведенный с привлечением 187 учащихся 5-7 в 2014/2015 учебном году в гимназии №1409 города Москвы показал, что школьники, изучавшие устройство жесткого диска с использованием VR-технологий (92 человека), показали лучшие результаты и получили в среднем оценку на 0.612 баллов выше, чем контрольная группа, прошедшая обучение по классической методике [6].

Турецкие исследователи, выдвинули гипотезу, что использование технологий дополненной реальности в обучении повышает эффективность образовательного процесса и позволяет запомнить учащимся больше информации. Был проведен эксперимент в одном из университетов, в котором часть студентов изучали географию с использованием гарнитур дополненной реальности (экспериментальная группа), а другая часть (контрольная группа) обучалась по

традиционной методике. Анализ результатов показал, что студенты экспериментальной группы показали в среднем почти на 10% лучшие результаты, чем контрольная группа, и при этом были способны запомнить больше информации [7].

Авторы одного из мета-анализов по данной теме проанализировали сорок два эксперимента, из которых в пятнадцати пришли к выводу о незначительном влиянии виртуальной реальности на результативность обучения, двадцать шесть показали положительный эффект, и лишь одно исследование выявило более низкие результаты у экспериментальной группы, использовавшей виртуальную реальность в обучении, по сравнению с контрольной группой [8]. По полученным результатам было выявлено, что использование виртуальной реальности в обучении особенно эффективно при подготовке специалистов с техническим, медицинским образованием и работников спасательных служб. Также данный анализ показывает, что технологии виртуальной реальности более эффективны, чем традиционное обучение как механизм развития технических, практических и социальных навыков.

1.1 ОБЗОР АНАЛОГОВ

На данный момент существует множество различных тренажеров виртуальной реальности и симуляторов. Рассмотрим некоторые из них, связанные с темой работы.

На рисунке 1 изображен тренажер для специалистов по ТО Белаз 75313 [9], разработанный ООО «Джэй Эс Эй Групп».

Рисунок 1 – Тренажер для обучения по прохождению ТО Белаз 75313

У данного продукта есть несколько достоинств. Во-первых, взаимодействие с техникой и различными её агрегатами происходит при помощи виртуальных инструментов при их коллизии с определенными областями, при самом взаимодействии могут проигрываться различные анимации. Во-вторых, присутствует инструкция к тренажеру. В-третьих, есть как режим обучения, так и режим тестирования. В-четвертых, в режиме обучения необходимые объекты, области и их контуры подсвечиваются ярким зеленым цветом, что облегчает их поиск.

К недостаткам данного тренажера можно отнести: плавное перемещение, что может вызывать головокружение, отсутствие «всплывающих» подсказок со вспомогательной информацией. Исходя из документации к тренажеру, поддерживается только гарнитура Pico.

Виртуальный тренажер от ООО «Корпоративные системы Плюс» [10] изображенный на рисунке 2 представляет из себя набор сценариев по ремонту и обслуживанию электромобиля.

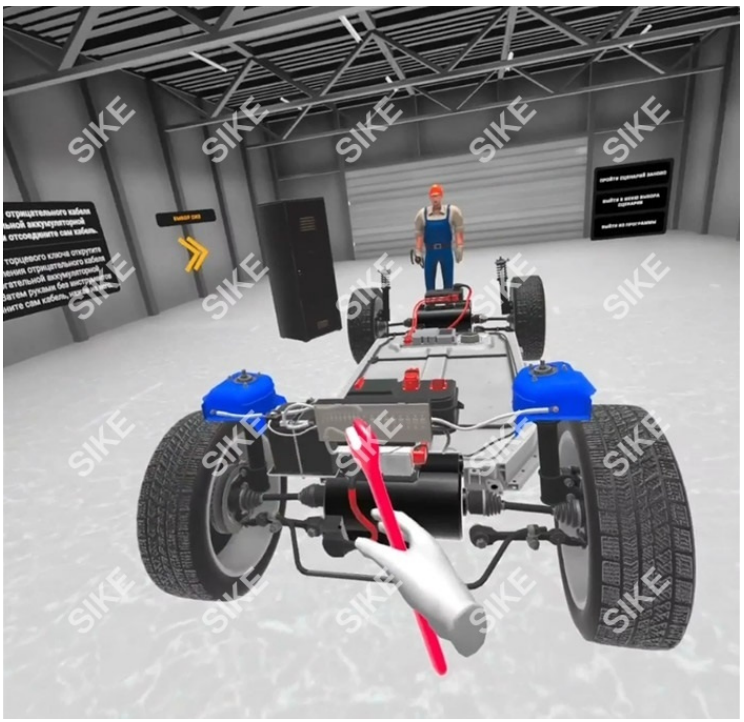

Рисунок 2 – Виртуальный тренажерный комплекс «Ремонт и обслуживание электромобиля VR»

Его преимуществами являются: набор сценариев, между которыми можно переключаться относительно низкие системные требования к ПК, появляющаяся в интерфейсе вспомогательная информация, необходимые объекты выделяются подсветкой, мгновенное перемещение по сцене.

К недостаткам данного продукта можно отнести ограниченный набор сценариев, который не затрагивает многие процессы ремонта и обслуживания техники, специализация на электромобилях. Также подсветка объектов выполнена оранжевым цветом, который не виден на фоне многих других поверхностей. Выбор инструментов осуществляется посредством взаимодействия с меню, никак не привязанному к позиции виртуальной камеры в тренажере, что не удобно. Поддерживаются только гарнитуры Oculus Rift CV1/S, Quest 1/2.

На рисунке 3 изображен еще один тренажер от ООО «Корпоративные системы Плюс» [11]. Отличием от предыдущего продукта является только зеленый цвет подсветки, который облегчает поиск объекта.

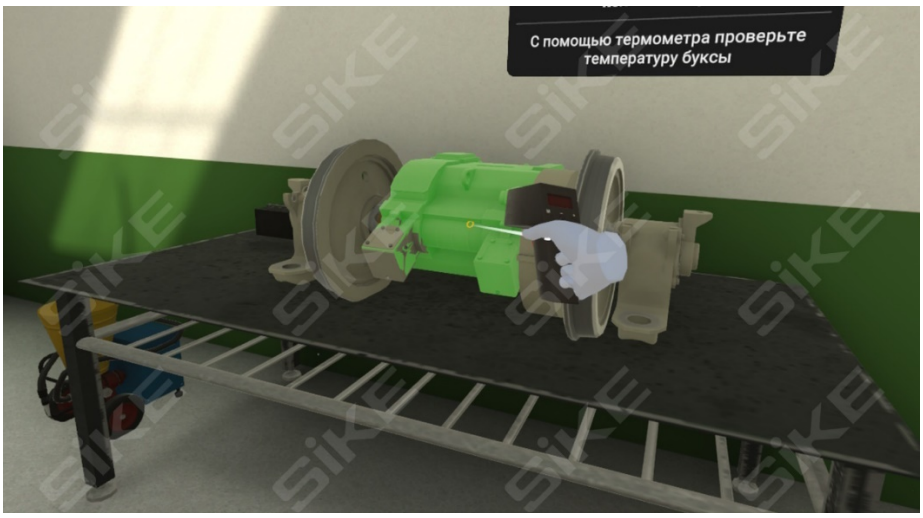

Рисунок 3 – Виртуальный тренажерный комплекс «Ремонт и обслуживание тепловоза 2ТЭ116 VR»

На рисунке 4 изображен тренажер представленный на акселерационной программе КрИЖТ [12].

Рисунок 4 – Тренажер виртуальной реальности по ремонту узлов и агрегатов

К преимуществам данного аналога можно отнести следующее: наличие конструктора и библиотеки элементов узлов, что позволяет самостоятельно на месте создать различные сценарии обучения, текстовый анализатор для помощи в создании цифрового двойника, наличие вспомогательной информации.

Минусами данного тренажера являются: отсутствие подсветки элементов, специализация только на железнодорожной технике. На данный момент продукт находится на ранних стадиях разработки. Информации о поддерживаемых гарнитурах виртуальной реальности нет.

Для сравнения аналогов были выбраны следующие критерии: поддержка широкого спектра гарнитур виртуальной реальности, подсветка объектов в

обучающем режиме, наличие справочной и/или вспомогательной информации в меню или графическом пользовательском интерфейсе, мгновенно перемещение по сцене.

Поддержка широкого спектра гарнитур виртуальной реальности является одним из главных критериев сравнения, так как узкая специализация на определенном устройстве ограничивает пользователя. В России некоторые гарнитуры уже сейчас невозможно купить новыми, а некоторые сняли с производства.

Подсветка объектов в обучающем режиме ярким контрастным цветом позволяет ускорить процесс обучения - обучающийся может не знать внешний вид некоторых деталей или их местоположения, и самостоятельный поиск детали в подобных ситуациях отнимает много времени.

Справочная информация позволяет получить более подробные знания о предмете.

Мгновенное перемещение позволяет повысить комфорт при использовании гарнитуры виртуальной реальности, о чем будет сказано далее в разделе функциональных требований к приложению, и сэкономить время на перемещении по сцене.

В таблице 1 приведено сравнение вышеперечисленных аналогов по данным параметрам.

Таблица 1 – Сравнение аналогов

Название тренажера / **Название критерия**	Тренажер по ТО Белаз 75313	«Ремонт и обслуживание электромобиля VR»	«Ремонт и обслуживание тепловоза VR	Тренажер ВР по ремонту узлов и агрегатов
Поддержка широкого списка VR-гарнитур	-	-	-	?
Подсветка объектов	+	+	+	-
Справочная и/или вспомогательная информация в меню или GUI	-	+	+	+
Мгновенное перемещение по сцене	-	+	+	?

ВЫВОД ПО РАЗДЕЛУ 1

В первом разделе был проведен обзор литературы по тематике работы, который помог определить функциональные и нефункциональные требования к приложению. Были проанализированы несколько аналогов, представленных на рынке и выявлены их сильные и слабые стороны. Одним из главных минусов существующих аналогов является отсутствие поддержки широкого спектра гарнитур.

2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

2.1. ТРЕБОВАНИЯ К ПРЛОЖЕНИЮ

Формирование требований к разрабатываемому продукту является одним из важнейших этапов проектирования. Требования позволяют понять, что продукт должен делать, устанавливают ограничения, дают возможность оценить масштаб работ и расставить приоритеты задач. Правильно разработанные требования уменьшают время на разработку продукта и снижают количество конфликтных ситуаций между исполнителем и заказчиком.

Для разрабатываемого приложения были определены следующие функциональные и нефункциональные требования.

2.1.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1. У пользователя должна быть возможность взаимодействовать с агрегатами и узлами виртуальной техники.

Для создания большей интерактивности необходимо дать пользователю самому взаимодействовать с виртуальным окружением. Данное взаимодействие можно реализовать несколькими способами. Для экономии времени и удобства пользователя можно дистанционно взаимодействовать с виртуальной средой при помощи лучей. Так ему не придется каждый раз перемещаться к объектам на расстояние вытянутой руки. Другим вариантом является взаимодействие при помощи объектов, будь то виртуальный гаечный ключ или другой инструмент, различные детали или агрегаты.

2. Наличие обучающего режима, в котором должны быть: подсветка объектов, с которыми необходимо взаимодействовать; список необходимых действий в графический пользовательский интерфейс; справочная информация по техническому процессу или объекту в графическом пользовательском интерфейсе.

Так как тренажер разрабатывается для дальнейшего его использования в процессе обучения персонала, то в нем как минимум должен быть режим обучения.

Различная справочная информация является неотъемлемой частью любого тренажера. Краткую справку по различным агрегатам, узлам, инструментам, действиям можно вынести в виде различных «всплывающих» подсказок. Более подробную информацию лучше вынести в отдельное меню.

Подсветка объектов необходима для легкого и быстрого обнаружения нужных объектов на сцене, что сильно экономит время обучающегося.

3. Возможность мгновенного перемещения по сцене при помощи «телепортации».

Из-за упомянутого ранее сенсорного несоответствия плавное перемещение пользователя по сцене вызывает головокружение гораздо быстрее, чем мгновенное перемещение, так как у человека будет складываться ощущение, что он находится в непрерывном движении, хотя он на самом деле стоит на месте.

2.1.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1. Использование технологии OpenXR.

OpenXR это бесплатный стандарт доступа к устройствам виртуальной и дополненной реальности с открытым кодом [13]. По сути – это некоторый набор программно-аппаратных интерфейсов для работы с большинством устройств расширенной реальности, который пришел на смену комплектам для разработки программного обеспечения под каждую отдельную платформу расширенной реальности. Именно благодаря данному стандарту один и тот же код будет одинаково работать на разных гарнитурах виртуальной реальности, что сильно облегчает разработку и поддержку приложения, а также не ограничивает разработчиков и пользователей в выборе гарнитуры.

2. Требования к персональному компьютеру.

Любые задержки в отрисовке изображения, падения частоты кадров, визуальные артефакты, которые не так сильно мешают пользователю в обычных приложениях для персональных компьютеров, в виртуальной реальности могут очень быстро вызвать головокружение или головную боль. Поэтому система, на которой тренажер будет использоваться, должна быть достаточно

производительной, чтобы обработать изображение с задержкой не более 20 мс и частотой кадров не менее 72 кадров/сек.

Для создания стереоскопического эффекта системе необходимо визуализировать два изображения сразу (по одному для каждого глаза). Частота обновления экрана у гарнитур виртуальной реальности выше, чем у стандартных мониторов: 90-144 Гц против 59-72 Гц у обычного монитора. Разрешение на каждый отдельный глаз в гарнитуре виртуальной реальности составляет около 2000x2000 точек. Из-за этих трех факторов приложения виртуальной реальности требовательны к частоте работы центрального и графического процессоров, объему видеопамяти и пропускной способности видеопамяти графического процессора.

Гарнитуры виртуальной реальности ввиду своего размера и веса весьма ограничены в вычислительной мощности, если в принципе имеют таковую. Данный факт зачастую делает невозможным их использование отдельно от персонального компьютера или другого достаточно мощного вычислительного устройства. Компьютер, к которому будет подключена подобная гарнитура, должен соответствовать хотя бы минимальным системным требованиям к их совместному использованию:

- Операционная система: Windows 10;
- Процессор: Intel i5-4590 / AMD Ryzen 5 1500X или лучше;
- Оперативная память: 8 ГБ или больше;
- Видеокарта: NVIDIA GeForce GTX 970 / AMD Radeon 400 серии или лучше, поддерживающая DirectX 11;
- USB 3.0 порт или кабельное соединение с Wi-Fi роутером, работающим в диапазоне частот 5 ГГц.

2.2 ВЫБОР ИНСТРУМЕНТОВ РАЗРАБОТКИ

2.2.1 ИГРОВОЙ ДВИЖОК

В качестве среды разработки были рассмотрены следующие игровые движки: Unity, Unreal Engine, Godot Engine, CryEngine, Unigine.

В первую очередь было обращено внимание на поддержку OpenXR. Судя по информации с официального сайта данного стандарта на данный момент OpenXR интегрирован в Unity версии 2020 LTS и новее, Epic Unreal Engine версии 4.24 и новее, Godot Game engine версии Core 4.0 и новее [13]. Как видно, в Unigine и CryEngine до сих пор не используется OpenXR из-за чего они не подходят для разработки данного приложения.

Разработчики Unigine говорили, что, OpenXR выглядит многообещающе и, возможно, будет интегрирован в их движок, либо будет добавлен аналогичный функционал непосредственно в Unigine [14]. На момент обращения к официальному сайту движка реализована поддержка Oculus Rift, HTC Vive / Vive Pro, но ни о какой поддержке широкого спектра гарнитур расширенной реальности не идет и речи [15].

Остальные три игровых движка в целом предоставляют схожий функционал. В каждом из них есть интегрированная среда разработки, визуальный редактор, инструменты для работы с 3D-моделями, анимацией, ландшафтом [16, 17, 18].

Функционал Unreal Engine 5 избыточен для реализации данного приложения, так как нам не нужно создавать больших открытых пространств на сцене или сложных визуальных эффектов. При своей избыточности Unreal Engine 5 объем памяти, который занимает сам движок и приложение, созданное на его основе, в два-три раза выше в сравнении с Unity или Godot.

Godot наиболее оптимизированный из всех трех, но, в виду большей модульности и простоты поддержки проекта небольшой командой, поддерживающих OpenXR, был выбран Unity.

2.2.2. XR INTERACTION TOOLKIT

Для создания взаимодействия с виртуальными объектами и пользовательским интерфейсом в виртуальной реальности был использован плагин XR Interaction Toolkit – высокоуровневая система, предоставляющая фреймворк, который делает взаимодействия с различными объектами на сцене доступными из событий ввода игрового движка.

2.2.3. ИНСТРУМЕНТЫ СОЗДАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

В качестве инструментов для создания пользовательского интерфейса были рассмотрены UI Toolkit и Unity UI (uGUI).

UI Toolkit на данный момент является плагином для движка Unity версии 2021 LTS и новее, который предоставляет набор инструментов для разработки как пользовательского интерфейса, так и интерфейса для визуального редактора. Он схож с инструментами для создания веб-интерфейсов, такими как HTML-CSS и UXML-USS [19].

uGUI является встроенным в движок основным инструментом для создания именно пользовательского интерфейса. Данная система основана на так называемых GameObject, которые представляют из себя обычный объект, размещаемый на сцене в Unity.

UI Toolkit обладает отдельным окном пользовательского графического интерфейса, который удобнее, чем интерфейс uGUI. Также в нем есть инструмент тестирования и отладки создаваемого интерфейса. В uGUI же для отладки и тестирования интерфейса приходится запускать всю сцену, что увеличивает время на создание интерфейса. Помимо этого, UI Toolkit имеет Unity Style Sheets – формальный язык описания внешнего вида интерфейса, основанного на CSS, что позволяет упростить и ускорить создание однотипных элементов графического интерфейса.

В приложениях виртуальной реальности весь интерфейс представляет из себя некоторые экраны, расположенные на сцене – по сути игровые объекты. Использование же оверлея (в данном случае под оверлеем подразумевается 2D-интерфейс, визуализируемый поверх любых других объектов на сцене), затруднено тем, что в гарнитурах виртуальной реальности зачастую находятся два округлых искривленных дисплея, по одному на каждый глаз соответственно, которые могут сильно отличаться у разных гарнитур. По этой причине интерфейс, находящийся в оверлее, может неправильно отображаться или не отображаться вовсе. A UI Toolkit в первую очередь предназначен именно для оверлея и на данный момент несовместим с интерфейсом в виде игрового объекта на сцене. По данной причине был выбран uGUI.

uGUI совместим с UI Toolkit, и, при решении использовать в дальнейшем UI Toolkit, не обязательно переделывать старый интерфейс [19].

2.3. ПОСТРОЕНИЕ АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ, ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ И СИСТЕМ ПРИЛОЖЕНИЯ, СВЯЗЕЙ МЕЖДУ НИМИ

Хорошо спроектированная архитектура приложения может уменьшить затраты времени и денег как в процессе его реализации, так и в процессе поддержки или добавления нового функционала.

Архитектурные шаблоны представляют из себя шаблон часто используемого решения распространенных задач в архитектуре программ. Они позволяют облегчить проектирование программы, улучшить и упростить саму программу при правильном их применении.

Одним из функциональных требований к приложению является возможность пользователя взаимодействовать с виртуальным окружением. При взаимодействии пользователя с отдельными частями окружения нам может понадобиться функционал сразу нескольких компонентов или систем приложения. Некоторые системы и компоненты могут быть сильно связаны, например система ввода и перемещения пользователя по виртуальной сцене. В подобных случаях мы можем

использовать нужный функционал, обратившись напрямую к связанной системе. Но если системы связаны слабо, то у нас могут возникнуть проблемы с использованием нужного нам функционала. Для решения данной проблемы и обеспечения взаимодействия между множеством компонентов программы в реальном времени было принято решение использовать шаблон архитектуры, управляемой событиями (сокращенно EDA – event-driven architecture).

Основной идеей архитектуры, управляемой событиями, является разделение системы на независимые компоненты, которые будут общаться между собой при помощи событий. При этом каждый компонент будет обрабатывать только нужные ему события.

Архитектура, управляемая событиями, состоит из трех логических частей:

- генератор событий, который вызывает событие;
- обработчик событий – место, где событие идентифицируется и выбирается соответствующая реакция на него, которая затем выполняется;
- канал или шина событий – механизм, по которому передается информация от генератора событий к обработчику.

Преимуществами использования данного шаблона являются: хорошая масштабируемость, гибкость и отказоустойчивость. Недостатками использования являются возможные проблемы с производительностью и сложности в отладке программы.

На рисунке 5 представлена схема взаимодействия компонентов и систем приложения.

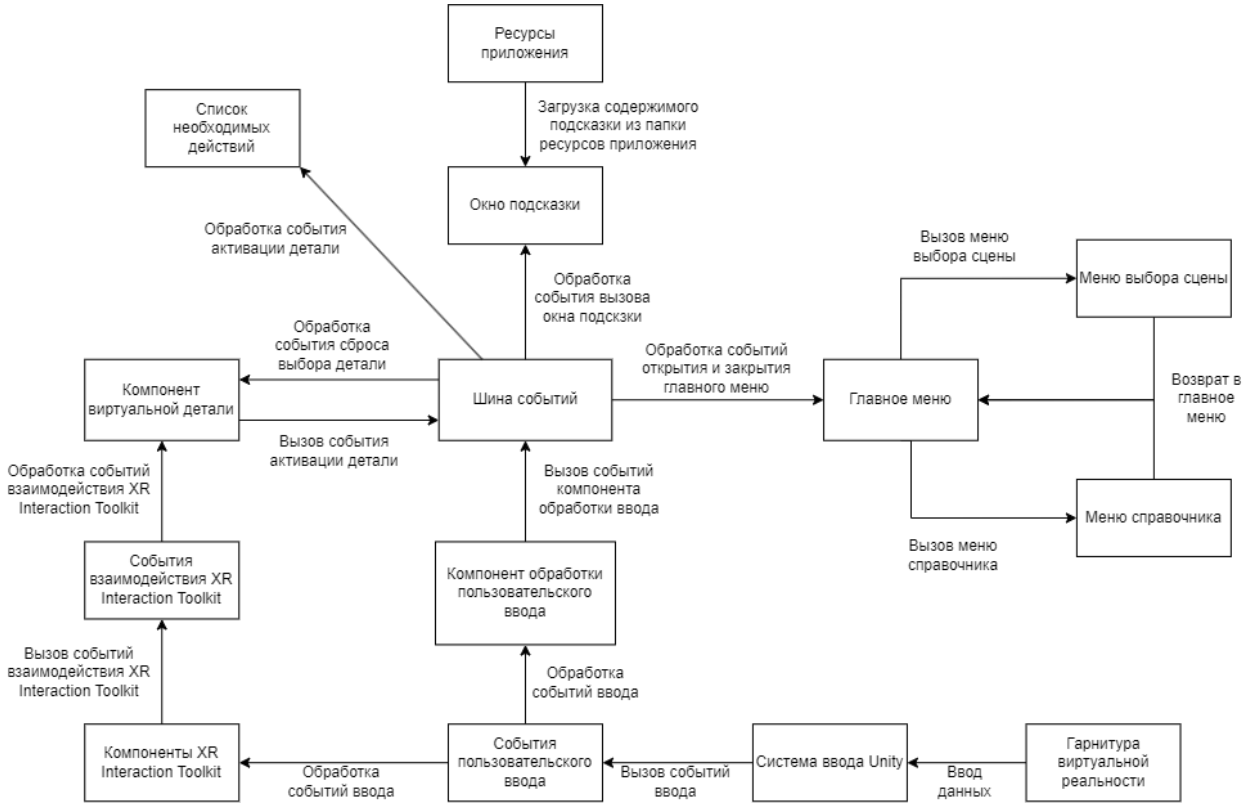

Рисунок 5 – Схема взаимодействия компонентов и систем приложения

2.3.1. ПРОЕКТИРОВАНИЕ КОМПОНЕНТА ВИРТУАЛЬНОЙ ДЕТАЛИ

Компонент виртуальной детали отвечает за взаимодействие с деталью и хранит информацию о ней.

Функциональные требования к компоненту виртуальной детали следующие:

- должна быть предусмотрена возможность выбора детали, чтобы исключить возможность случайной активации событий, связанных с деталью;
- выбранная деталь должна быть выделена подсветкой зеленого цвета;
- одновременно может быть выбрана только одна деталь;
- должна быть предусмотрена возможность сброса выбора детали;
- должна быть предусмотрена возможность заблокировать деталь;
- заблокированную деталь нельзя выбрать;
- должна быть предусмотрена возможность разблокировать деталь;

- выбрать можно только разблокированную деталь;
- должна быть предусмотрена возможность выбора детали в качестве текущей цели;
- должна быть возможность «активировать» выбранную деталь, тем самым запустив некоторую последовательность событий по типу проигрывания анимации, смены состояния детали, изменения материала 3D-модели и пр.

2.3.2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТА ОБРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА

В данном компоненте должна происходить обработка событий ввода с контроллеров виртуальной гарнитуры, которые не связаны с выбором или активацией детали: вызов главного меню, вызов подсказок, сброс выбора виртуальной детали.

2.3.3. ПРОЕКТИРОВАНИЕ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Спроектируем визуальную часть интерфейса в соответствии с функциональными требованиями к нему.

2.3.3.1. ПРОЕКТИРОВАНИЕ ГЛАВНОГО МЕНЮ

Главное меню, представленное на рисунке 6, должно включать в себя:

- кнопка перехода в меню выбора сцены;
- кнопка перехода в меню справочника;
- кнопка перезапуска сцены, которая доступна только непосредственно на обучающих сценах;
- кнопка закрытия приложения;
- кнопка закрытия окна меню.

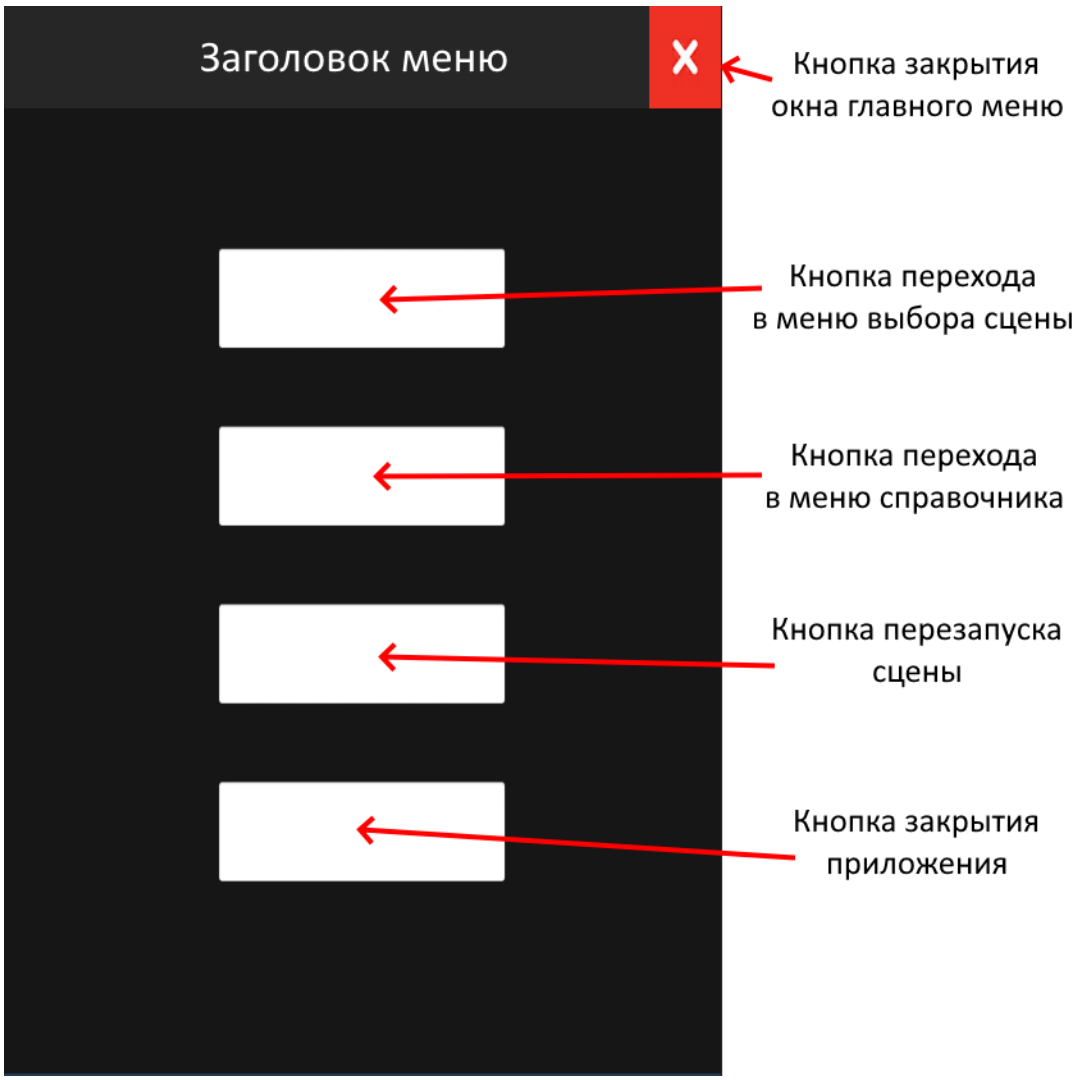

Рисунок 6 – Внешний вид главного меню и расположение элементов в нем

2.3.3.2 ПРОЕКТИРОВАНИЕ МЕНЮ ВЫБОРА СЦЕНЫ

Меню выбора сцены, представленное на рисунке 7, должно включать в себя:

- кнопку закрытия окна меню;
- кнопку возврата в предыдущее меню;
- кнопки выбора конкретных обучающих сцен, расположенные сеткой;
- полосу прокрутки.

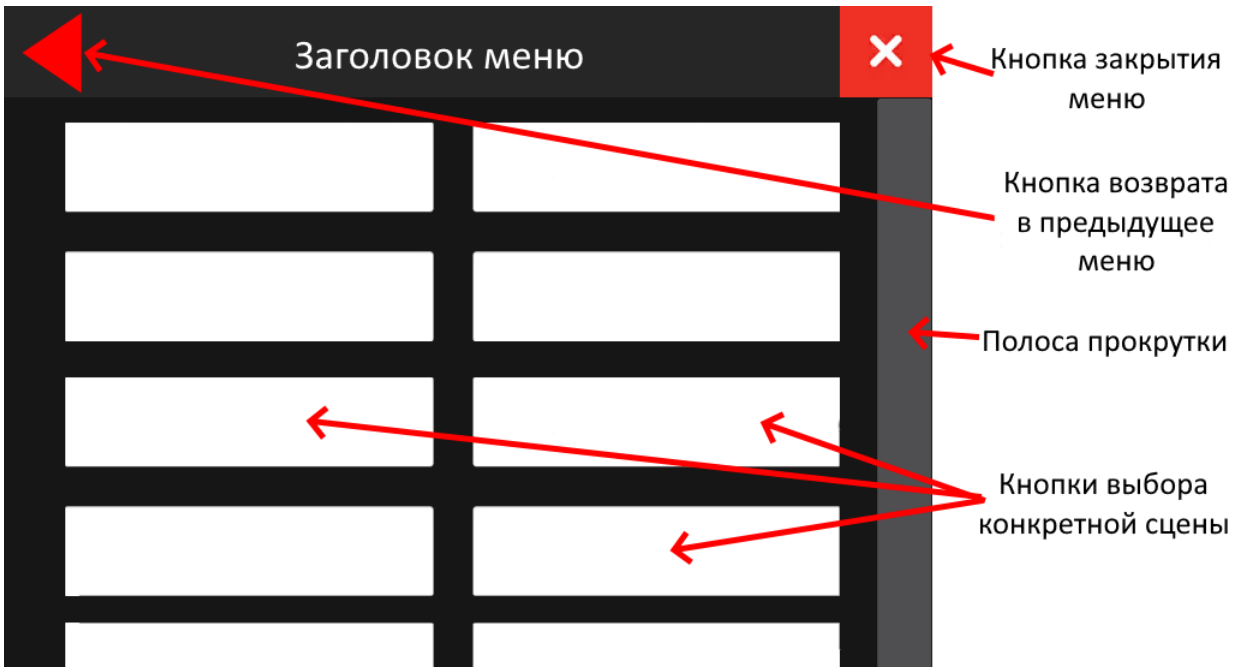

Рисунок 7 – Внешний вид меню выбора сцены и расположение элементов в нем

2.3.3.3. ПРОЕКТИРОВАНИЕ МЕНЮ СПРАВОЧНИКА

Справочный материал необходим для более глубоко понимания предмета изучения. Справка по техническим процессам, узлам и агрегатам техники является необходимым элементом подобного обучающего приложения.

Интерфейс справочника должен содержать:

- кнопку закрытия окна меню;
- кнопку возврата в предыдущее меню;
- справочник в виде многоуровневого списка;
- у каждого элемента многоуровневого списка должна быть кнопка, позволяющая свернуть/развернуть его содержание.

На рисунке 8 приведены внешний вид меню справочника и расположение элементов в нем.

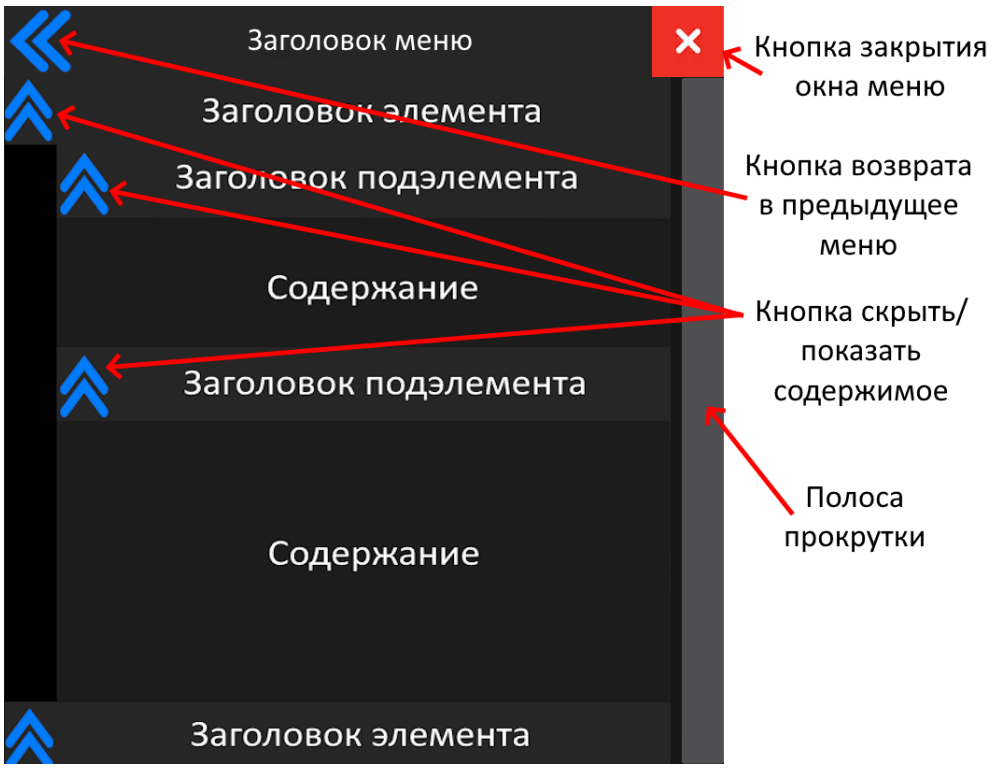

Рисунок 8 – Внешний вид меню справочника и расположения элементов в нем

2.3.3.4. ПРОЕКТИРОВАНИЕ СПИСКА НЕОБХОДИМЫХ ДЕЙСТВИЙ

Обучающемуся необходимо понять, что и в какой последовательности ему нужно делать для прохождения сценария. Для этого нужно вывести в графический пользовательский интерфейс список необходимых действий.

Интерфейс должен содержать:

- многоуровневый список;
- у каждого элемента списка должна быть кнопка, позволяющая свернуть/развернуть вложенные элементы, название и содержание;
- полосу прокрутки.

Выполнение действия должно как-то визуально отображаться в списке, например изменением цвета текста содержания, заголовка элемента, зачеркиванием текста и т.д.

На рисунке 9 приведены внешний вид списка необходимых действий и расположение элементов в нем.

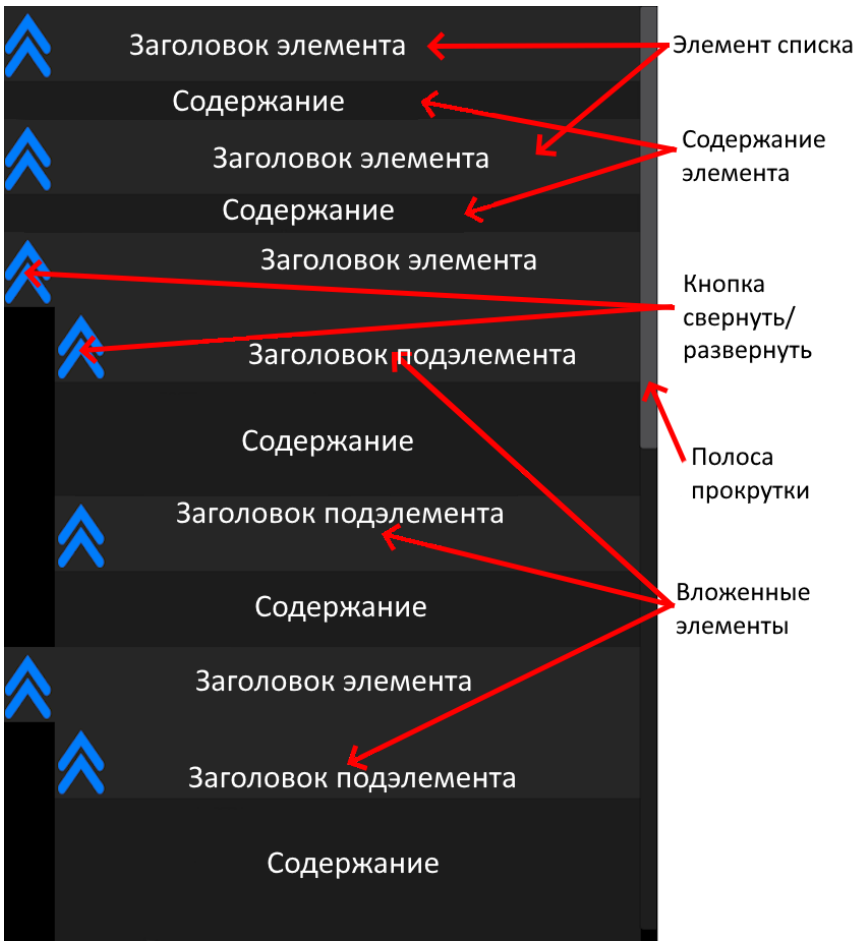

Рисунок 9 – Внешний вид списка необходимых действий и расположение элементов в нем

2.3.3.5. ПРОЕКТИРОВАНИЕ ОКНА ПОДСКАЗОК

Во время выполнения обучающего сценария ученику важно понимать с чем он работает, что и как ему нужно делать. Некоторый минимум необходимой информации можно дать пользователю через подсказки. Их можно реализовать в виде появляющихся над объектами окон или в виде окна или объекта, каким-либо образом привязанного к позиции камеры пользователя на сцене.

Интерфейс должен содержать:

- краткое название подсказки;
- содержание подсказки;
- полосу прокрутки;
- кнопку закрытия окна.

Внешний вид и расположение элементов представлены на рисунке 10.

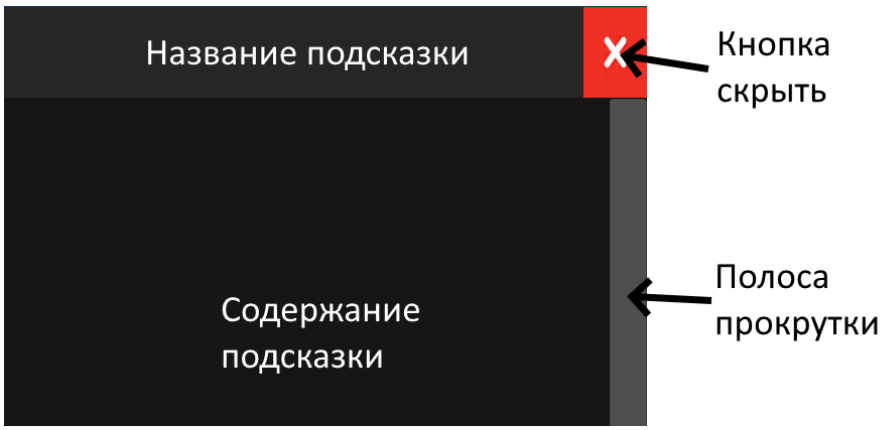

Рисунок 10 – Внешний вид окна подсказки и расположения элементов в нем

ВЫВОД ПО РАЗДЕЛУ 2

Во втором разделе были обоснованы функциональные и нефункциональные требования к разрабатываемому приложению. Рассмотрены игровые движки и другие инструменты реализации. В ходе их анализа были выбраны игровой движок Unity, плагин XR Interaction Toolkit и встроенный в Unity инструмент для создания пользовательского графического интерфейса uGUI.

Выбрана архитектура, управляемая событиями. Спроектирована схема взаимодействия систем и компонентов приложения. Спроектированы компоненты приложения и визуальная часть следующих элементов пользовательского графического интерфейса:

- главного меню;
- меню выбора сцены;
- меню справочника;
- окна подсказок;
- списка необходимых действий.

3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1. СРЕДСТВА РЕАЛИЗАЦИИ

Для реализации приложения был использован игровой движок Unity, плагин XR Interaction Toolkit, встроенный инструмент создания пользовательского интерфейса uGUI, текстовый редактор Visual Studio Code и язык программирования C#.

Архитектура проекта в Unity основана на архитектурном шаблоне Entity-Component, который несколько отличается от шаблона Entity-Component-System, но в целом очень на него похож.

Согласно шаблону Entity-Component, существуют сущности (Entity) и компоненты (Component). Сущности по своей сути являются контейнерами для компонентов, которые не обладают никакими свойствами и хранят в себе очень мало информации (в основном идентификационный номер и название). Компоненты же хранят в себе различные данные и логику.

Сцена (Scene) в Unity состоит из игровых объектов (GameObject), которые и представляют из себя сущность, с прикрепленными к ним компонентами. Любой игровой объект может быть сохранен в виде префаба (Prefab).

Префаб – это особый тип файла, который хранит в себе игровой объект, со всеми его дочерними игровыми объектами, прикрепленными к ним компонентами и конкретными данными, хранящимися в компонентах. Префаб представляет из себя некоторый шаблон объекта, из которого можно создавать конкретные экземпляры на сцене [20]. Любые изменения в префабе применяются ко всем его экземплярам на сцене.

Программирование в Unity заключается в создании пользовательских классов и классов-компонентов. Все пользовательские классы-компоненты должны наследоваться от класса MonoBehaviour. Для их обозначения на диаграмме классов к их имени будет добавлен модификатор «component».

В классе MonoBehaviour есть методы Update(), Start(), OnDisable(), OnEnable(), OnValidate(), Awake() и пр., которые могут быть переопределены в наследниках.

Update() вызывается каждый кадр, если данный компонент активен.

Start() вызывается во время первого кадра, когда компонент стал активным в первый раз, перед вызовом метода Update(). Вызывается только один раз за весь жизненный цикл компонента. Обычно используется для инициализации объектов и компонентов.

OnDisable() и OnEnable() вызываются, когда компонент становится неактивным или активным соответственно. В них в основном будут происходить подписка на события и отписка от них.

OnValidate() вызывается во время загрузки скрипта или изменении данных компонента в редакторе сцен.

Awake() вызывается при загрузке сцены и только один раз за весь жизненный цикл компонента.

Плагин XR Interaction Toolkit позволяет создавать взаимодействия с объектами в смешанной реальности. В нем есть три основных компонента: субъект (Interactor), объект (Interactable) и менеджер взаимодействий (Interaction Manager) [21].

Пара из субъекта и объекта имеет три состояния: Hover, Select и Activate. В изменение состояния вовлечены субъект и объект, и оба из них будут уведомлены о смене состояния одного из них.

Состояние Hover означает, что субъект готов и может начать какое-то взаимодействие с объектом, но, обычно, никак не меняет их поведение. Если представить субъект в виде курсора мыши, а объект в виде окна интерфейса, то в состояние Hover они входят, когда курсор мыши находится внутри окна. Для простоты будем говорить, что в состоянии Hover субъект указывает на объект.

Для перехода в состояние Select нужно произвести действие, например, нажать на клавишу или курок контроллера.

Чтобы перейти в состояние Activate тоже нужно произвести действие, как и для перехода в состояние Select. Данное состояние используется для дальнейшего взаимодействия с объектом, и, обычно, при переходе в данное состояние, вызываются какие-либо события.

Пара из объекта и субъекта уведомляются о смене состояния друг друга. Уведомление происходит посредством событий. Так как один субъект может взаимодействовать только с одним объектом одновременно, но с одним объектом могут взаимодействовать сразу несколько субъектом, то объект имеет несколько большее количество событий. В приложении используются в основном события объекта, причем следующие: Hover Entered, Last Hover Exited и Select Entered.

Событие Hover Entered вызывается, когда какой-либо субъект начинает указывать на объект. Last Hover Exited вызывается, когда последний субъект, указывающий на объект, перестает на него указывать. Select Entered вызывается, когда пара объект – субъект входит в состояние Select.

3.2. РЕАЛИЗАЦИЯ АРХИТЕКТУРЫ, УПРАВЛЯЕМОЙ СОБЫТИЯМИ

Исходный код реализации архитектуры, управляемой событиями приведен в листинге А.1 приложения А.

Событие представлено интерфейсом IEvent, классы PartActivationEvent, ShowTooltipEvent, ShowHideMainMenu и PartDeselectionEvent, реализующие данный интерфейс, являются конкретными типами событий и содержат в себе данные, необходимые для своей обработки.

Обработчик события в общем виде представлен интерфейсом IBaseEventReciever. Конкретные типы обработчиков представлены интерфейсами наследниками от IBaseEventReciever. В данном приложение тип обработчика всего один – IEventReciever<T>, который в качестве T принимает IEvent. Классы, реализующие интерфейс IEventReciever<T>, являются обработчиками событий. Метод OnEvent(T @event) данных классов является методом обработки события.

Шину данных реализует класс EventBus. Он хранит в себе списки обработчиков событий в словаре recievers, ключом для которого является тип события, и словарь со ссылками на конкретные обработчики событий recieversHashToReference, ключом для которого является хэш-код обработчика.

Для представления шины событий на сцене будем использовать пустой игровой объект, с прикрепленным к нему компонентом EventBusHolder, являющимся классом заглушкой, который будет создавать объект EventBus в методе Awake() и хранить его в себе.

Рассмотрим методы класса Event Bus.

Метод Register<T>(IEventReciever<T> receiver) подписывает обработчики событий на события. Он принимает на вход обработчик события и добавляет его в список обработчиков конкретного события и в словарь ссылок на конкретные обработчики.

Метод Unregister<T>(IEventReciever<T> receiver) совершает отписку обработчиков от событий.

Метод RaiseEvent<T>(T @event) вызывает конкретное событие, передавая информацию о нем от генератора события к его обработчикам и вызывая обработку события.

Классы, вызывающие события при помощи метода RaiseEvent<T>(T @event) шины событий, являются генераторами событий.

3.2. РЕАЛИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ ПОЛЬЗОВАТЕЛЯ С ВИРТУАЛЬНЫМИ ДЕТАЛЯМИ

Для реализации взаимодействия пользователя с виртуальными деталями был использован плагин XR Interaction Toolkit. Был выбран способ взаимодействия с помощью луча, так как при его использовании не обязательно приближаться к объекту вплотную, что экономит время на перемещении по сцене. В качестве Interactor был выбран Ray Interactor, который работает на встроенном в Unity методе обнаружения объектов на прямой в трехмерном пространстве Raycast.

К игровым объектам детали, был добавлены компоненты XR Simple Interactable, являющийся одной из реализаций Interactable плагина XR Interaction Toolkit, и MeshCollider – компонент коллайдера (фигуры для проверки на столкновения), без которого Ray Interactor не сможет взаимодействовать с XR SimpleInteractable.

3.2.1. РЕАЛИЗАЦИЯ КОМПОНЕНТА ВИРТУАЛЬНОЙ ДЕТАЛИ

UML-диаграмма классов компонента виртуальной детали изображена на рисунке 11, а его реализация представлена в листинге Б.1 приложения Б.

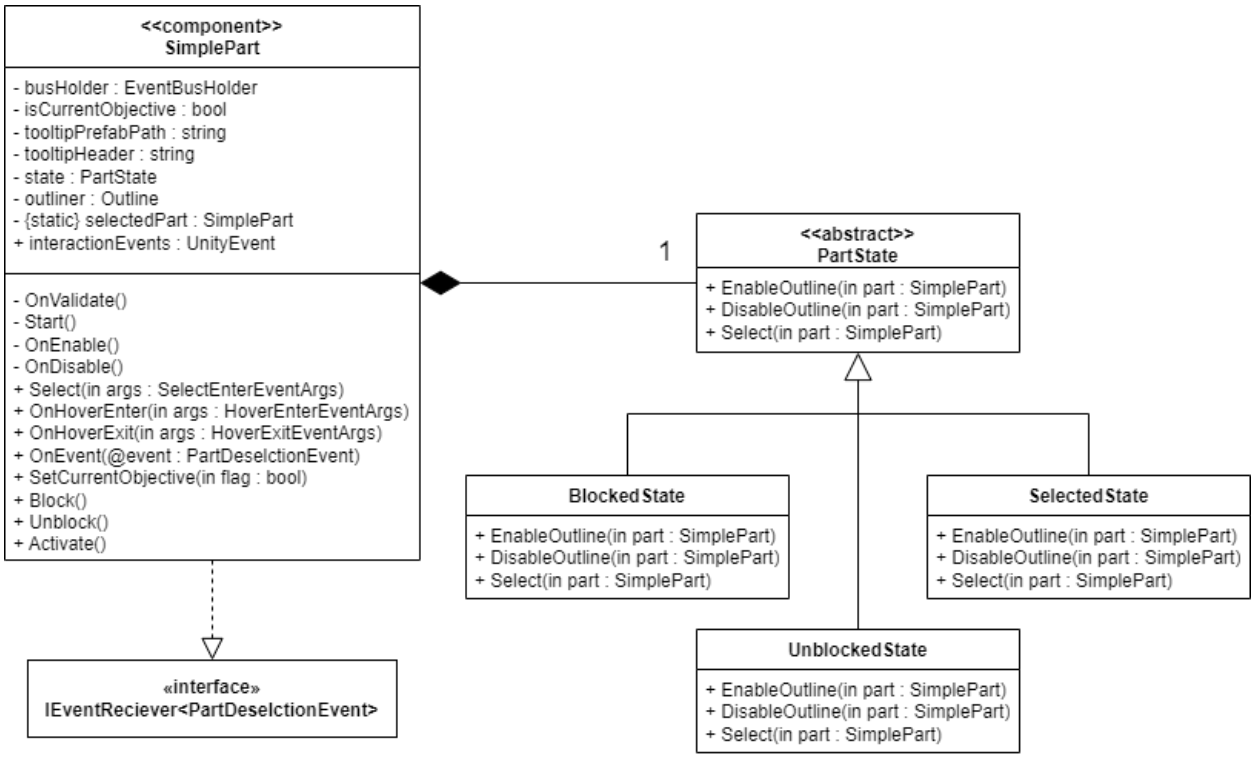

Рисунок 11 – UML-диаграмма классов компонента виртуальной детали

Компонент виртуальной детали реализует поведенческий шаблон «Состояние». В зависимости от текущего класса состояния SimplePart, представленного наследниками абстрактного класса PartState, будет меняться поведение методов EnableOutline(SimplePart part), DisableOutline(SimplePart part) и Select(SimplePart part).

Класс SimplePart представляет сам компонент, который добавляется к игровому объекту. Он реализует интерфейс IEventReciever<PartDeselectionEvent>,

таким образом являясь обработчиком события PartDeselectionEvent, вызываемого компонентом обработки пользовательского ввода и отвечающего за сброс выбора детали.

Разберем методы класса SimplePart.

OnValidate() добавляет к игровому объекту детали компоненты XR Simple Interactable и Mesh Renderer, если таковые не присутствуют у данного игрового объекта.

В методе Start() происходит инициализация компонента SimplePart и подписка методов Select, OnHoverEnter и OnHoverExit на события Select Entered, First Hover Entered и Last Hover Exited компонента XR Simple Interactable соответственно.

В методах OnEnable() и OnDisable() происходит подписка на событие PartDeselectionEvent и отписка от него соответственно.

Select(SelectEnterEventArgs args) обрабатывает событие перехода компонента XR Simple Interactable в состояние Selected. Переход компонента в данное состояние осуществляется наведением на виртуальную деталь луча XR Ray Interactable и нажатием на кнопку выбора детали на контроллере виртуальной гарнитуры. Данный метод вызывает метод Select(SimplePart part) текущего класса состояния детали.

Метод OnHoverEnter(HoverEnterEventArgs atgs) вызывает метод EnableOutline(SimplePart part) класса состояния детали при вызове события On First Hover Entered. Данное событие вызывается, когда мы наводим первый луч XR Ray Interactor на виртуальную деталь. Пока первый луч наведен на деталь, наведение других лучей на данную деталь не будет вызывать данное событие.

Метод OnHoverExit(HoverExitEventArgs atgs) вызывает метод DisableOutline(SimplePart part) класса текущего состояния детали при вызове события On Last Hover Exited. Данное событие вызывается, когда мы перестаем наводить последний луч XR Ray Interactor, который был наведен на данную виртуальную деталь.

OnEvent(PartDeselectionEvent @event) является обработчиком события сброса выбора детали PartDeselectionEvent. Отвечает за сброс выбора детали и вызывает метод DisableOutline(SimplePart part) текущего класса состояния детали.

Метод SetCurrentObjective(bool flag) отвечает за установку значения переменно IsCurrentObjective, отвечающей за выбор детали в качестве текущей целит. В зависимости от значения данной переменной меняется поведение методов DisableOutline(SimplePart part) классов состояний детали.

Методы Block() и Unblock() меняют класс состояния компонента детали на BlockedState и UnblockedState соответственно.

Activate() отвечает за «активацию» детали. Генерирует события PartActivationEvent.

Разберем методы класса BlockedState, представляющего заблокированную деталь.

EnableOutline(SimplePart part) включает у детали подсветку синего цвета.

DisableOutline(SimplePart part) выключает подсветку, если переменная IsCurrentObjective объекта класса SimplePart имеет значение false, и включает подсветку красного цвета, если переменная имеет значение true.

Select(SimplePart part) ничего не делает. Отвечает за невозможность выбора заблокированной детали.

Разберем методы класса UnblockedState, представляющего разблокированную деталь.

Методы EnableOutline(SimplePart part) и DisableOutline(SimplePart part) аналогичны одноименным методам класса BlockedState().

Select(SimplePart part) меняет текущий класс состояния детали на SelectedState. Отвечает за возможность выбора детали.

Разберем методы класс SelectedState, представляющего выбранную деталь.

Методы EnableOutline(SimplePart part) и Disable(SimplePart part) включают у детали подсветку зеленого цвета.

Метод Select(SimplePart part) активирует блокирует деталь, вызывает метод Activate() компонента SimplePart, выключает подсветку детали, сбрасывает выбор детали.

3.2.2. РЕАЛИЗАЦИЯ КОМПОНЕНТА ОБРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА

Компонент обработки пользовательского ввода реализован в класс InteractionManager, исходный код которого приведен в листинге В.1 приложения В.

Метод Deselect(InputAction.CallbackContext context) обрабатывает событие нажатия на кнопку сброса выбора детали контроллера гарнитуры виртуальной реальности, генерируя событие PartDeselectionEvent.

Метод ShowTooltip(InputAction.CallbackContext context) обрабатывает событие нажатие на кнопку вызова окна подсказки контроллера гарнитуры виртуальной реальности, генерирует событие ShowTooltipEvent.

Метод ShowHideMainMenu(InputAction.CallbackContext context) обрабатывает нажатие на кнопку открытия/закрытия главного меню контроллера гарнитуры виртуальной реальности, генерируя событие ShowHideMainMenu.

3.3. РЕАЛИЗАЦИЯ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

3.3.1.РЕАЛИЗАЦИЯ ГЛАВНОГО МЕНЮ, МЕНЮ ВЫБОРА СЦЕНЫ И МЕНЮ СПРАВОЧНИКА

Главное меню, меню выбора сцены и меню выбора справочника реализованы в одном окне. На рисунке 12 изображена UML-диаграмма классов системы, управляющей данным окном.

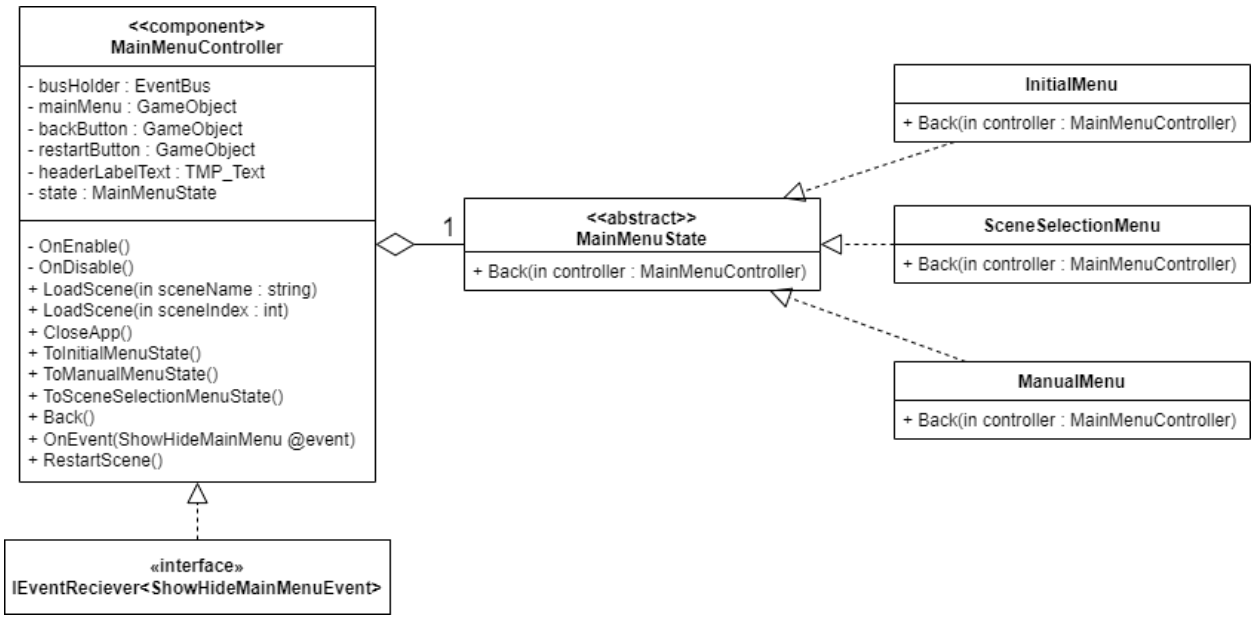

Рисунок 12 – UML-диаграмма классов контроллера окна главного меню

Компонент MainMenuController, абстрактный клас MainMenuState и его дочерние классы InitialMenu, SeceneSelectionMenu и ManualMenu реализуют паттерн проектирования «Состояние». Их исходный код представлен в листинге Г.1 приложения Г. В данном случае каждое отдельное меню данного окна представлены в виде классов состояний. На данный момент они определяют логику поведения кнопки возврата к предыдущему меню. Класс MainMenuController реализует интерфейс IEventReciever<ShowHideMainMenuEvent>, то есть является обработчиком данного события.

Методы LoadScene(int sceneIndex) и LoadScene(string sceneName) нужны для загрузки соответствующей сцены по нажатию соответствующей кнопки в меню выбора сцены.

CloseApp() закрывает приложение.

ToInitialMenuState(), ToManualMenuState() и ToSceneSelectionState() отвечают за изменение текущего класса состояния MainMenuController.

В методах OnEnable() и OnDisable() происходит подписка метод OnEvent(ShowHideMainMenu @event) на событие ShowHideMainMenu и отписка от него соответственно. Также OnEnable() отвечает за отображение только на обучающих сценах кнопки перезапуска сцены в главном меню.

OnEvent (ShowHideMainMenu @event) является обработчиком события ShowHideMainMenuEvent и отвечает за открытие/закрытие окна меню.

RestartScene() перезапускает текущую сцену.

На рисунках 13-15 изображены получившиеся варианты главного меню, меню выбора сцены и меню справочника.

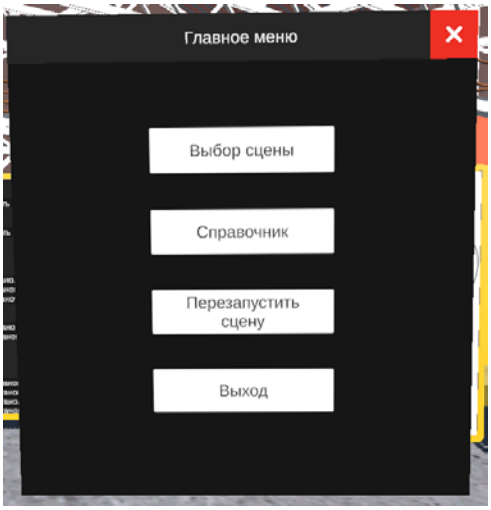

Рисунок 13 – Реализация интерфейса главного меню

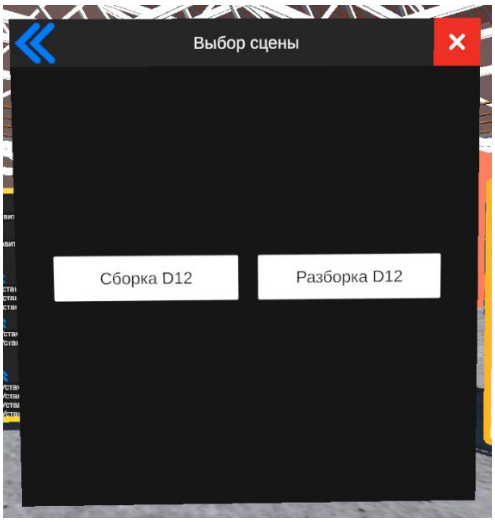

Рисунок 14 – Реализация интерфейса меню выбора сцены

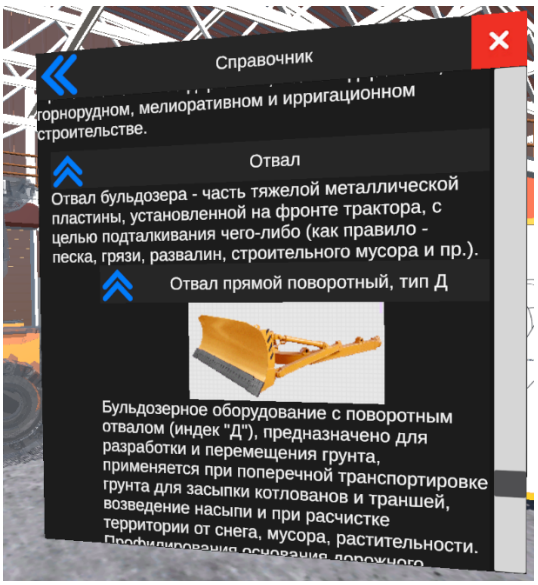

Рисунок 15 – Реализация интерфейса меню справочника

На данный момент в меню выбора сцены доступны лишь две сцены. Сцена под названием «Сборка D12» пустая и создана для тестирования функционала приложения.

Справочник содержит как общую информацию о деталях, узлах и агрегатах техники, так и более детальную. На данный момент в нем приведены характеристики бульдозера D12, информация о ходовой части, трансмиссии и допустимом навесном оборудовании данной модели, часть информации об устанавливаемых на данную технику двигателях и признаках, указывающих на неисправность вышеперечисленных частей. Также в данное меню была добавлена вспомогательная информация об управлении и взаимодействии в самом приложении.

3.3.2. РЕАЛИЗАЦИЯ СПИСКА НЕОБХОДИМЫХ ДЕЙСТВИЙ

Список необходимых действий выполнен в отдельном окне в виде экрана. В нем приведена последовательность действий для выполнения сценария. Реализация данного элемента интерфейса изображена на рисунке 16.

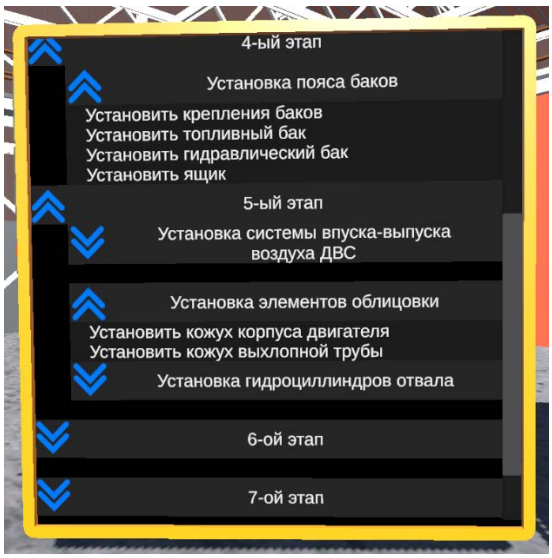

Рисунок 16 – Реализация интерфейса списка необходимых действий

К элементам и содержанию списка необходимых действий был добавлен компонент ActionsListElement, исходный код которого приведен в листинге Д.1 приложения Д. Данный компонент реализует интерфейс IEventReciever<PartActivationEvent>, таким образом являясь обработчиком события PartActivationEvent. Подписка обработчика на данное событие и отписка от него выполняются в методах OnEnable() и OnDisable() соответственно. Обработчиком события является метод OnEvent(PartActivationEvent @event), который зачеркивает текст, соответствующий выполненному действию.

3.3.3. РЕАЛИЗАЦИЯ ОКНА ПОДСКАЗОК

Реализация окна подсказок представлена на рисунке 17.

Рисунок 17 – Реализация окна подсказок

К игровому объекту окна подсказок был добавлен компонент TooltipController, исходный код которого приведен в листинге Е.1 приложения Е. Он реализует интерфейс IEventReciever<ShowTooltipEvent>, тем самым являясь обработчиком событий ShowTooltipEvent. В методах OnEnable() и OnDisable() происходит подписка обработчика на событие ShowTooltipEvent и отписка от него соответственно. Обработчик OnEvent(ShowTooltipEvent @event) вызывает окно подсказок, очищает его содержимое и заголовок, загружает префаб содержимого окна из папки ресурсов приложения, и устанавливает заголовок окна.

3.4. ДОРАБОТКА МГНОВЕННОГО ПЕРЕМЕЩЕНИЯ ПО СЦЕНЕ

XR Interaction Toolkit предлагает готовые компоненты, реализующие мгновенное перемещение по сцене на основе XR Ray Interactor, но оно нуждается в доработке. По умолчанию на телепортацию и взаимодействие с объектами назначена одна и та же кнопка на контроллерах. Помимо этого, луч для телепортации постоянно отображается, что не очень удобно, так как его можно перепутать с лучом для взаимодействия. Для решения этих проблем был написан компонент TeleportationController. Он переназначает телепортацию на другую кнопку контроллеров гарнитур виртуальной реальности. При нажатии на эту кнопку он включает соответствующий XR Ray Interactor. При отпускании этой клавиши он производит перемещение пользователя по сцене и выключение компонента луча. Исходный код компонента, отвечающего за данный функционал приведен в листинге Ж.1 приложения Ж.

3.5. СЦЕНА РАЗБОРКИ БУЛЬДОЗЕРА D12

В ходе реализации приложения была создана сцена, демонстрирующая финальные этапы сборки бульдозера на заводе. Для данной сцены была переведена в формат .FBX и упрощена модель бульдозера D12. Из модели были убраны лишние для данной сцены элементы, например внутренняя часть кабины, проводка

и внутренние части двигателя. Были созданы анимации установки деталей, узлов и агрегатов на бульдозер при помощи встроенного в Unity аниматора и C# кода.

Сцена демонстрирует процессы установки ходовой части, трансмиссии, двигателя, системы охлаждения, системы впуска-выпуска воздуха ДВС, кабины, пояса баков и навесного оборудования. На рисунке 18 изображен внешний вид данной сцены.

Рисунок 18 – Сцена разборки бульдозера D12

ВЫВОД ПО РАЗДЕЛУ 3

В третьей главе были описаны некоторые из выбранных средств реализации и сама программная реализация приложения. Были реализованы:

- компонент виртуальной детали;
- компонент обработки пользовательского ввода
- главное меню;
- меню справочника;
- меню выбора сцены;
- список необходимых действий.

Доработано мгновенное перемещение по сцене.

Также описана, созданная в ходе выполнения работы, сцена сборки бульдозера D12.

4. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Для проверки работоспособности приложения были проведены функциональное и нефункциональное тестирования приложения.

Эти тестирования довольно просты, но позволяют выявить отличия между требуемыми и реально существующими свойствами приложения. Функциональное тестирование позволяет убедиться, что все функции приложения работают корректно и правильно реагируют на входные данные. Нефункциональное тестирование позволяет оценить производительность приложения.

4.1. ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

В таблице 2 приведены результаты функционального тестирования компонентов детали и пользовательского ввода.

Таблица 2 – результаты функционального тестирования компонентов детали и пользовательского ввода

Название теста	**Действия**	**Результат**	**Тест пройден?**
Проверка работы подсветки	На демонстрационной сцене навести луч для взаимодействия на любую разблокированную деталь, выбрать деталь, сбросить выбор детали, отвести луч от детали	При наведении луча деталь подсветилась синим цветом, при выборе детали подсветка сменила цвет на зеленый, при сбросе выбора подсветка сменила цвет на синий, при отведении луча подсветка выключилась	Да
Проверка возможности активации детали	Навести луч на деталь, выбрать деталь, выбрать деталь повторно	Деталь была активирована, вызвав соответствующие события	Да

Продолжение таблицы 2

Название теста	**Действия**	**Результат**	**Тест пройден?**
Проверка возможности одновременного выбора только одной детали	Навести луч на деталь, выбрать деталь, перевести луч на другу деталь, попытаться выбрать деталь	Первая деталь была выбрана. При попытке выбора второй детали ничего не произошло	Да
Вызов меню подсказок	Навести луч на деталь, нажать кнопку вызова меню подсказок	Подсказка, соответствующая детали, была вызвана	Да
Проверка возможности выбора заблокированной детали	Навести луч на заблокированную деталь, попытаться выбрать деталь	Деталь не была выбрана	Да
Вызов и закрытие окна главного меню	Вызвать главное меню, нажать на кнопку закрытия главного меню	Окно главного меню было успешно вызвано, после чего закрыто	Да
Проверка возможности сброса выбора детали	Выбрать деталь, нажать на кнопку сброса выбора детали	Сброс детали произошел успешно	Да

В таблице 3 приведены результаты функционального тестирования главного меню.

Таблица 3 – результаты функционального тестирования главного меню

Название теста	**Действия**	**Результат**	**Тест пройден?**
Переход в меню выбора сцены	Нажать на кнопку перехода в меню выбора сцены	Было открыто меню выбора сцены	Да
Переход в меню справочника	Нажать на кнопку перехода в меню справочника	Было открыто меню справочника	Да

Продолжение таблицы 3

Название теста	**Действия**	**Результат**	**Тест пройден?**
Проверка правильности работы кнопки перезапуска сцены	Вызвать главное меню в пустой сцене, перейти на сцену сборки D12, вызвать главное меню, нажать на кнопку перезапуска сцены	В пустой сцене не отображается кнопка перезапуска, на демонстрационной сцене кнопка перезапуска отображается. Сцена перезагружается при нажатии кнопки	Да
Проверка функционала кнопки закрытия приложения	Вызвать главное меню, нажать на кнопку закрытия приложения	Приложение закрылось	Да

В таблице 4 приведены результаты функционального тестирования меню выбора сцены.

Таблица 4 – результаты функционального тестирования меню выбора сцены

Название теста	**Действия**	**Результат**	**Тест пройден?**
Переход на другую сцену	Нажать на любую кнопку перехода на сцену	Был осуществлен переход на соответствующую сцену	Да
Возврат в главное меню	Нажать на кнопку возврата в главное меню	Был осуществлен переход в главное меню	Да
Проверка работы полосы прокрутки	Использовать полосу прокрутки	Содержимое меню прокручивается	Да

В таблице 5 приведены результаты функционального тестирования меню справочника.

Таблица 5 – результаты функционального тестирования меню справочника.

Название теста	**Действия**	**Результат**	**Тест пройден?**
Возврат в главное меню	Нажать на кнопку возврата в главное меню	Был осуществлен переход в главное меню	Да
Проверка работы полосы прокрутки	Использовать полосу прокрутки	Содержимое меню прокручивается	Да
Проверка работы кнопок скрытия/показа содержимого элемента справочника	Нажать на кнопку показа содержимого, нажать на кнопку скрытия содержимого	Содержимое элемента справочника было показано, после чего скрыто	Да

В таблице 6 приведены результаты функционального тестирования меню подсказок.

Таблица 6 – результаты функционального тестирования меню подсказок

Название теста	**Действия**	**Результат**	**Тест пройден?**
Проверка работы полосы прокрутки	Использовать полосу прокрутки	Содержимое меню прокручивается	Да
Проверка работы кнопки скрытия меню подсказок	Нажать на кнопку скрытия меню подсказок	Меню подсказок скрыто	Да

В таблице 7 приведены результаты функционального тестирования списка необходимых действий.

Таблица 7 – результаты функционального тестирования списка необходимых действий.

Название теста	**Действия**	**Результат**	**Тест пройден?**
Возврат в главное меню	Нажать на кнопку возврата в главное меню	Был осуществлен переход в главное меню	Да
Проверка работы полосы прокрутки	Использовать полосу прокрутки	Содержимое меню прокручивается	Да
Проверка работы кнопок скрытия/показа содержимого элемента справочника	Нажать на кнопку показа содержимого, нажать на кнопку скрытия содержимого	Содержимое элемента справочника было показано, после чего скрыто	Да
Проверка изменения списка необходимых действий при выполнении действия	Выполнить указанное в списке действие	Текст, отвечающий за указанное действие, был зачеркнут	Да

4.2. НЕФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

Были установлены минимальные системные требования для приложения. Мы можем отследить производительность приложения при помощи встроенных инструментов Unity.

Тестирование проводилось на персональном компьютере со следующей конфигурацией:

- процессор AMD Ryzen 5 2600 Six-Core Processor, 3400 МГц;
- 32 ГБ оперативной памяти DDR4;
- видеокарта NVIDIA GeForce GTX 1650 Super, 4 ГБ видеопамяти, частота видеопамяти 12000 МГц, разрядность шины памяти 128 бит, поддержка DirectX 12;
- жесткий диск объемом 1 Тб и скоростью вращения 7200 об./мин.

В ходе тестирования, результаты которого представлены на рисунках 19 и 20, можно увидеть, что количество кадров в секунду находится в диапазоне от 100 до 200.

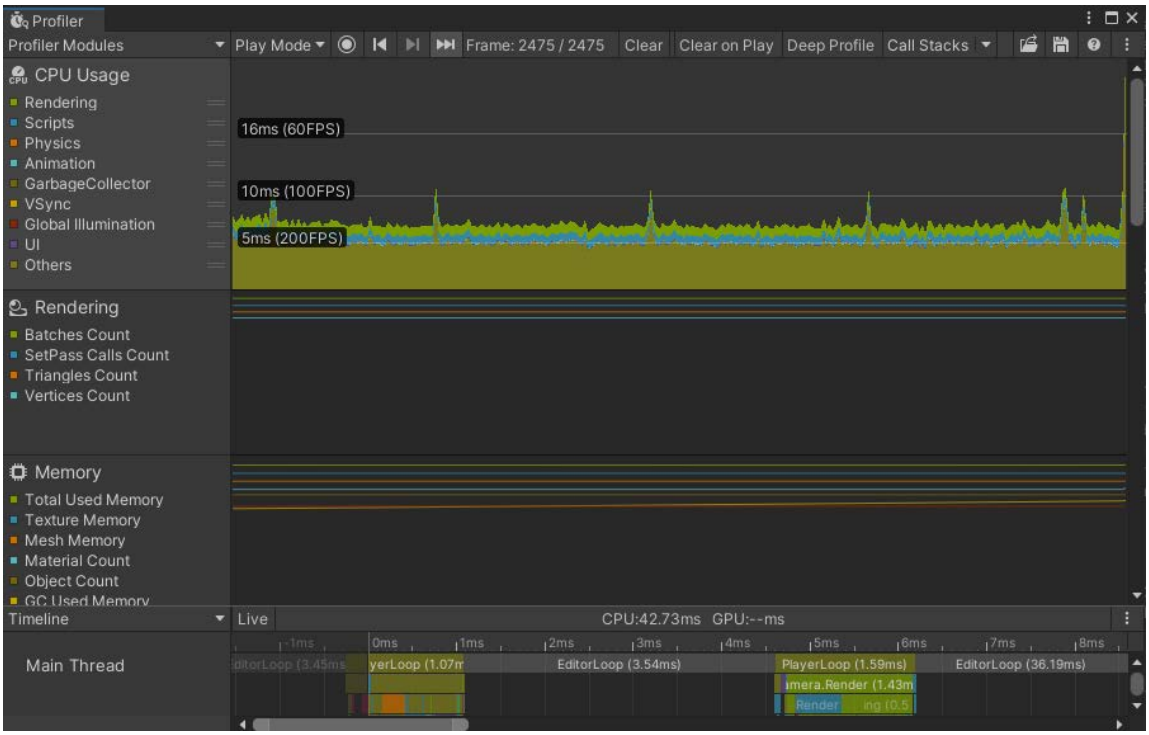

Рисунок 19 – Вывод окна Profiler игрового движка Unity, во время работы приложения

Рисунок 20 – Вывод встроенного инструмента статистики Unity в один из моментов работы приложения

ВЫВОД ПО РАЗДЕЛУ 4

В четвертом разделе были проведены функциональное и нефункциональное тестирование приложения. В рамках тестирования был создана демонстрационная сцена, показывающая финальные этапы сборки бульдозера на производстве.

Результаты тестирования показали, что приложение соответствует выдвинутым к нему требованиям.

ЗАКЛЮЧЕНИЕ

В ходе аналитического обзора научно-технической, нормативной и методической литературы по тематике работы было выяснено, что использование приложений виртуальной реальности в процессе обучения, если не улучшает результаты обучения, то увеличивает вовлеченность ученика в обучающий процесс. Были выявлены следующие преимущества применения технологий виртуальной реальности в обучении: повышение вовлеченности ученика в процесс обучения; повышение безопасности ученика; возможность воссоздания в виртуальной реальности сценариев, воспроизведение которых в реальном мире нецелесообразно; возможность сбора данных о процессе обучения, что позволит использовать их для дальнейшего анализа.

Был проведен сравнительный обзор аналогов, в результате которого было выяснено, что основным недостатком существующих аналогов является поддержка малого количества гарнитур виртуальной реальности.

Были рассмотрены инструменты реализации приложения. Было проведено их сравнение, в итоге которого были выбраны следующие инструменты: игровой движок Unity, плагин XR Interaction Toolkit и система создания пользовательского графического интерфейса uGUI.

При проектировании приложения были сформированы функциональные и нефункциональные требования к нему. Была построена архитектура приложения с использованием архитектуры, управляемой событиями. Данный архитектурный шаблон позволяет улучшить масштабируемость, гибкость и производительность приложения. Были спроектированы системы и компоненты приложения.

Реализация приложения была рассмотрена работе. В ходе реализации была создана демонстрационная обучающая схема, показывающая финальные этапы сборки бульдозера на заводе.

Были проведены функциональное и нефункциональное тестирования, согласно результатам которых, приложение соответствует выдвинутым к нему требованиям.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Extended reality (XR) market size worldwide from 2021 to 2026 : Statista. – https://www.statista.com/statistics/591181/global-augmented-virtual-reality-market-size/. Дата обращения: 21.12.2023.
2. Данилова А.С. VR как инструмент развития отраслевого образования: опыт железнодорожного вуза / А.С. Данилова // Сборник трудов Всероссийской научно-практической конференции с международным участием «Драйверы развития общего и профессионального образования». – Павлово, 2021. – С. 189-194.
3. Андрушко, Д.Ю. Применение технологий виртуальной и дополненной реальности в образовательном процессе: проблемы и перспективы / Д.Ю. Андрушко // Научное обозрение. Педагогические науки. – 2018. - № 6 – С. 5-10.
4. Effectiveness of virtual reality-based instruction on students' learning outcomes in K-12 and higher education: A meta-analysis / Zahira Merchant, Ernest T. Goetz, Lauren Cifuentes, Wendy Keeny-Kennicutt, Trina J. Davis // Computers & Education. – 2014. - №70. – С. 29-40.
5. Challenges and Prospects of Virtual Reality and Augmented Reality Utilization among Primary School Teachers: A Developing Country Perspective / Nasser Alalwan, Lim Cheng, Hoasm Al-Samarraie, Reem Yousef, Ahmed Ibrahim Alzahranim, Samer Muthana Sarsam // Studies in Educational Evaluation. – 2020. - №66.
6. Гриншкун А.В. Об эффективности использования технологий дополненной реальности при обучении школьников информатике / Гриншкун А.В. // Вестник МГПУ. Серия «Информатика и информатизация образования». – 2016. - №35. – С. 98-103.
7. The impact of mobile augmented reality in geography education: achievements, cognitive loads and views of university students / Zeynep Turan, Elif Meral,

Ibrahim Fevzi Sahin // Journal of Geography in Higher Education. – 2018. - №42 – С. 427-441.

8. Meta-Analysis Assessing the Effects of Virtual Reality Training on Student Learning and Skills Development / Diego F. Angel-Urdinola, Catalina Castillo-Castro, Angela Hoyos. – 2021.
9. Промышленные VR-Тренажеры. – https://jsa-group.ru/vr#!/tab/575683464-2 Дата обращения: 28.02.2024.
10. Ремонт и обслуживание электромобиля VR – Виртуальный тренажерный комплекс (VR ТОиР) SIKE. – https://shop.sike.ru/vr-trenazher-remont-ehlektromobilya. Дата обращения: 28.02.2024.
11. Ремонт и обслуживание тепловоза VR – Виртуальный тренажерный комплекс (VR ТОиР) SIKE. – https://shop.sike.ru/vr-trenazher-teplovoz. Дата обращения 28.02.2024.
12. Тренажер виртуальной реальности по ремонту узлов и агрегатов / Д.Н. Кравец, В.М. Лаер, В.Е. Офицеров // Молодежная наука : труды XXVII Всероссийской студенческой научнопрактической конференции КрИЖТ ИрГУПС (г. Красноярск, 20.04.2023 г.) : Т. 1: Секция «Транспортные системы»; секция «Инфраструктура железных дорог» / редкол. : В.А. Поморцев (отв. ред.) [и др.] ; КрИЖТ ИрГУПС. – Красноярск: КрИЖТ ИрГУПС, 2023. – С. 163-167.
13. OpenXR Overview – The Khronos Group Inc. – https://www.khronos.org/openxr/. Дата обращения: 24.04.24.
14. OpenXR – C++ Programming – UNIGINE Developers Community. – https://developer.unigine.com/forum/topic/5846-openxr/. Дата обращения: 24.04.24.
15. Начало работы с виртуальной реальностью – Документация – Unigine Developer. – https://developer.unigine.com/ru/docs/latest/start/vr/?rlang=cpp. Дата обращения: 24.04.24
16. Платформа и редактор для разработки RT3D-контента. – https://unity.com/ru/products/unity-engine. Дата обращения: 24.04.24.

17. List of features – Godot Engine (stable) documentation in English. – https://docs.godotengine.org/en/stable/about/list_of_features.html#scripting. Дата обращения: 24.04.24.
18. Features – Unreal Engine. – https://www.unrealengine.com/en-US/features. Дата обращения: 24.04.24
19. Разработка интерактивных интерфейсов | Unity UI Toolkit. – https://www.simplilearn.com/entity-component-system-introductory-guide-article Дата обращения: 26.05.2024.
20. Unity – Manual: Prefabs. – https://docs.unity3d.com/Manual/Prefabs.html. Дата обращения: 26.05.2024.
21. XR Interaction Toolkit | XR Interaction Toolkit | 2.0.4. – https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit%402.0/manual/index.html. Дата обращения 26.05.2024.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ АРХИТЕКТУРЫ, УПРАВЛЯЕМОЙ СОБЫТИЯМИ

Листинг А.1 – Исходный код EventBus.cs

```
using System;
using System.Collections.Generic;

public interface IEvent{}

public interface IBaseEventReciever{}

public interface IEventReciever<T> : IBaseEventReciever where T : IEvent{
    void OnEvent(T @event){}
}

//Класс шины событий
public class EventBus{
    //Словарь со списками ссылок на обработчики событий
    private Dictionary<Type, List<WeakReference<IBaseEventReciever>>> recievers;
    //Словарь со ссылками на обработчики
    private           Dictionary<int,           WeakReference<IBaseEventReciever>>
recieverHashToReference;

    public EventBus(){
        recievers               =               new               Dictionary<Type,
List<WeakReference<IBaseEventReciever>>>();
        recieverHashToReference           =           new           Dictionary<int,
WeakReference<IBaseEventReciever>>();
    }

    //Метод подписки обработчика на событие
    public void Register<T>(IEventReciever<T> reciever) where T : IEvent{
        Type eventType = typeof(T);

        if(!recievers.ContainsKey(eventType)){
            recievers[eventType] = new List<WeakReference<IBaseEventReciever>>();
        }

        WeakReference<IBaseEventReciever>          reference          =          new
WeakReference<IBaseEventReciever>(reciever);
        recievers[eventType].Add(reference);
        recieverHashToReference[reciever.GetHashCode()] = reference;
    }

    //Метод отписки обработчика
    public void Unregister<T>(IEventReciever<T> reciever) where T : IEvent{
        Type eventType = typeof (T);
        int recieverHash = reciever.GetHashCode();

        if(!recievers.ContainsKey(eventType)                                    ||
recieverHashToReference.ContainsKey(recieverHash)){return;}

        WeakReference<IBaseEventReciever>                reference                =
recieverHashToReference[recieverHash];
        recievers[eventType].Remove(reference);
        recieverHashToReference.Remove(recieverHash);
    }
```

Окончание листинга А.1

```
    public void RaiseEvent<T>(T @event) where T : IEvent{
        Type eventType = typeof(T);
        if(!recievers.ContainsKey(eventType)){return;}

        foreach(WeakReference<IBaseEventReciever>           reference           in
recievers[eventType]){
            if(reference.TryGetTarget(out IBaseEventReciever reciever)){
                ((IEventReciever<T>)reciever).OnEvent(@event);
            }
        }
    }
}

//Далее идут 4 класса событий с необходимой для них информацией
public class PartActivationEvent : IEvent{
    public readonly SimplePart part;

    public PartActivationEvent(SimplePart part){
        this.part = part;
    }
}

public class ShowTooltipEvent : IEvent{
    public readonly string tooltipContentPrefabPath, tooltipHeader;

    public ShowTooltipEvent(string contentPrefabPath, string header){
        tooltipContentPrefabPath = contentPrefabPath;
        tooltipHeader = header;
    }
}

public class ShowHideMainMenuEvent : IEvent{}

public class PartDeselectionEvent : IEvent{}
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ КОМПОНЕНТА ВИРТУАЛЬНОЙ ДЕТАЛИ

Листинг Б.1 – Исходный код SimplePart.cs

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using cakeslice;
using UnityEngine.XR.Interaction.Toolkit;

//Абстрактный класс состояния детали
public abstract class PartState{
    //Метод включения подсветки детали
    public abstract void EnableOutline(SimplePart part);

    //Метод выключения подсветки детали
    public abstract void DisableOutline(SimplePart part);

    //Метод, отвечающий за выбор детали
    public abstract void Select(SimplePart part);
}

//Класс состояния заблокированной детали
public class BlockedState : PartState{
    public override void EnableOutline(SimplePart part){
        Outline outline = part.Outliner;
        outline.color = 3;
        outline.enabled = true;
    }
    public override void DisableOutline(SimplePart part){
        if(part.IsCurrentObjective){
            part.Outliner.color = 2;
        }
        else{part.Outliner.enabled = false;}
    }
    public override void Select(SimplePart part){}
}

//Класс состояния разблокированной детали
public class UnblockedState : PartState{
    public override void EnableOutline(SimplePart part){
        Outline outline = part.Outliner;
        outline.color = 3;
        outline.enabled = true;
    }
    public override void DisableOutline(SimplePart part){
        if(part.IsCurrentObjective){
            part.Outliner.color = 2;
        }
        else{part.Outliner.enabled = false;}
    }
    public override void Select(SimplePart part){
        if(SimplePart.SelectedPart == null){
            SimplePart.SelectedPart = part;
            part.State = new SelectedState();
            part.State.EnableOutline(part);
        }
```

Продолжение листинга Б.1

```
    }
}
//Класс состояния выбранной детали
public class SelectedState : PartState{
    public override void EnableOutline(SimplePart part){
        Outline outline = part.Outliner;
        outline.color = 1;
        outline.enabled = true;
    }
    public override void DisableOutline(SimplePart part){
        Outline outline = part.Outliner;
        outline.color = 1;
        outline.enabled = true;
    }
    public override void Select(SimplePart part){
        part.Block();
        part.Activate();
        part.IsCurrentObjective = false;
        part.State.DisableOutline(part);
        SimplePart.SelectedPart = null;
    }
}

public class SimplePart : MonoBehaviour, IEventReciever<PartDeselectionEvent>
    [SerializeField] private EventBusHolder busHolder;
    [SerializeField] private bool isCurrentObjective = false;
    [SerializeField]    private    List<SimplePart>    partsToUnblock    =    new
List<SimplePart>();
    [SerializeField] private PartState state = new UnblockedState();
    [SerializeField] private string tooltipPrefabPath, tooltipHeader;
    private static SimplePart selectedPart = null;
    private Outline outliner;
    public UnityEvent interactionEvents;

    public static SimplePart SelectedPart{
        get{ return selectedPart; }
        set{ selectedPart = value; }
    }
    public PartState State{
        get{ return state; }
        set{ state = value; }
    }
    public bool IsCurrentObjective{
        get{ return isCurrentObjective; }
        set{ isCurrentObjective = value;}
    }
    public string TooltipPrefabPath{
        get{ return tooltipPrefabPath; }
    }
    public string TooltipHeader{
        get{ return tooltipHeader; }
    }
    public Outline Outliner{
        get{ return outliner; }
    }

    //Метод вызывает метод Select(SimplePart part) текущего класса состояния
    public void Select(SelectEnterEventArgs args){
        state.Select(this);
    }
```

Продолжение листинга Б.1

```
    // Метод вызывает метод EnableOutline(SimplePart part) текущего класса состояния
    public void OnHoverEnter(HoverEnterEventArgs args){
        state.EnableOutline(this);
    }

    // Метод вызывает метод DisableOutline(SimplePart part) текущего
    //класса состояния

    public void OnHoverExit(HoverExitEventArgs args){
        state.DisableOutline(this);
    }

    //Обработчик события PartDeselectionEvent, сбрасывает выбор детали
    public void OnEvent(PartDeselectionEvent @event){
        if(state.GetType() == typeof(SelectedState)){
            selectedPart = null;
            state = new UnblockedState();
            if(outliner.enabled){
                state.DisableOutline(this);
                state.EnableOutline(this);
            }
            else { state.DisableOutline(this); }
        }
    }
    public void SetCurrentObjective(bool flag){
        IsCurrentObjective = flag ? true : false;
    }

    //Метод меняет класс состояния на UnblockedState
    public void Unblock(){
        if(state.GetType() == typeof(BlockedState)){
            state = new UnblockedState();
            if(outliner.enabled){
                state.DisableOutline(this);
                state.EnableOutline(this);
            }
            else { state.DisableOutline(this); }
        }
    }

    //Метод меняет класс состояния на BlockedState
    public void Block(){
        state = new BlockedState();
        if(outliner.enabled){
                state.DisableOutline(this);
                state.EnableOutline(this);
            }
        else { state.DisableOutline(this); }
    }

    //Метод отвечающий за активацию детали
    public void Activate()
    {
        if (partsToUnblock != null) foreach (var part in partsToUnblock) if (part !=
null)              {part.Unblock();              part.state.DisableOutline(part);
part.SetCurrentObjective(true);}
        Deselect();
        interactionEvents?.Invoke();
        busHolder.EventBus.RaiseEvent(new PartActivationEvent(this));
    }
```

Окончание листинга Б.1

```
    private void OnValidate() {
        if(!gameObject?.GetComponent<Collider>())
gameObject.AddComponent<MeshCollider>();
        if(!gameObject?.GetComponent<Outline>())
gameObject.AddComponent<Outline>();
        if(!gameObject?.GetComponent<XRBaseInteractable>())
gameObject.AddComponent<XRSimpleInteractable>();
    }

    //Метод отвечает за инициализацию компонента
    private void Start()
    {
        outliner = GetComponent<Outline>();
        XRSimpleInteractable interactable = GetComponent<XRSimpleInteractable>();
        interactable.hoverEntered.AddListener(OnHoverEnter);
        interactable.lastHoverExited.AddListener(OnHoverExit);
        interactable.selectEntered.AddListener(Select);
        outliner.enabled = true;
        state.DisableOutline(this);
    }

    private void OnEnable(){
        busHolder.EventBus.Register(this as IEventReciever<PartDeselectionEvent>);
    }
    private void OnDisable(){
        busHolder.EventBus.Unregister(this                                    as
IEventReciever<PartDeselectionEvent>);
    }
}
```

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ КОМПОНЕНТА ОБРАБОТКИ ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА

Листинг В.1 – Исходный код InteractionManager.cs

```
using UnityEngine;
using System;
using UnityEngine.InputSystem;

public class InteractionManager : MonoBehaviour
{
    [SerializeField] private GameObject rightController;
    [SerializeField] private InputActionAsset inputActions;
    [SerializeField] private EventBusHolder busHolder;
    private InputAction deselect, mainMenu, showTip;

    //Генератор события сброса выбора детали PartDeselectionEvent
    public void Deselect(InputAction.CallbackContext context){
        busHolder.EventBus.RaiseEvent(new PartDeselectionEvent());
    }
    //Генератор события вызова подсказки ShowTooltipEvent
    //Получает путь хранения префаба содержимого подсказки и заголовок окна
    //подсказки из компонента виртуальной детали при помощи Raycast
    public void ShowTooltip(InputAction.CallbackContext context){
        Ray ray = new Ray(rightController.transform.position,
rightController.transform.TransformDirection(Vector3.forward));

        if(Physics.Raycast(ray, out RaycastHit hit, 100, 30)){
            SimplePart part = hit.transform.gameObject.GetComponent<SimplePart>();
            busHolder.EventBus.RaiseEvent(new
ShowTooltipEvent(part.TooltipPrefabPath, part.TooltipHeader));
        }
    }

    //Генератор события открытия/закрытия окна главного меню ShowHideMainMenuEvent
    public void ShowHideMainMenu(InputAction.CallbackContext context){
        busHolder.EventBus.RaiseEvent(new ShowHideMainMenuEvent());
    }

    private void OnEnable(){
        deselect = inputActions.FindActionMap("XRI RightHand
Interaction").FindAction("Deselect");
        showTip = inputActions.FindActionMap("XRI RightHand
Interaction").FindAction("Show Tooltip");
        mainMenu = inputActions.FindActionMap("XRI LeftHand
Interaction").FindAction("Show/Hide Menu");
        deselect.Enable();
        showTip.Enable();
        mainMenu.Enable();
        deselect.performed += Deselect;
        showTip.performed += ShowTooltip;
        mainMenu.performed += ShowHideMainMenu;
    }
    private void OnDisable(){
        deselect.performed -= Deselect;
        showTip.performed -= ShowTooltip;
        mainMenu.performed -= ShowHideMainMenu;
    }}
```

ПРИЛОЖЕНИЕ Г
ИСХОДНЫЙ КОД РЕАЛИЗАЦИИ ГЛАВНОГО МЕНЮ, МЕНЮ ВЫБОРА СЦЕНЫ И МЕНЮ СПРАВОЧНИКА

Листинг Г.1 – Исходный код MainMenuController.cs

```
using UnityEngine;
using TMPro;
using UnityEngine.SceneManagement;

//Абстрактный класс состояния окна главного меню
public abstract class MainMenuState{
    //Отвечает за функционал кнопки перехода в предыдущее меню
    public abstract void Back(MainMenuController controller);
}

public class MainMenuController : MonoBehaviour,
IEventReciever<ShowHideMainMenuEvent>
{
    [SerializeField] private EventBusHolder eventBus;
    [SerializeField] private GameObject mainMenu, backButton, restartButton;
    [SerializeField] private TMP_Text headerLabelText;
    [SerializeField] private MainMenuState state = new InitialMenu();
    public GameObject MainMenu { get { return mainMenu;}}
    public TMP_Text HeaderLabelText { get { return headerLabelText;} }
    public GameObject BackButton { get { return backButton;}}
    public MainMenuState State { get { return state;} set { state = value; } }

    //В OnEnable() включается отображение кнопки перезапуска сцены, если сцена
    //не является стартовой
    private void OnEnable(){
        if(SceneManager.GetActiveScene() != SceneManager.GetSceneByBuildIndex(0)){
            restartButton.SetActive(true);
        }
        else{
            restartButton.SetActive(false);
        }

        eventBus.EventBus.Register(this as IEventReciever<ShowHideMainMenuEvent>);
    }
    private void OnDisable(){
        eventBus.EventBus.Unregister(this as
IEventReciever<ShowHideMainMenuEvent>);
    }

    //Методы LoadScene загружают необходимую сцену
    public void LoadScene(string sceneName){
        SceneManager.LoadSceneAsync(sceneName);
    }
    public void LoadScene(int sceneIndex){
        SceneManager.LoadSceneAsync(sceneIndex);
    }

    //Закрывает приложение
    public void CloseApp(){
        Application.Quit();
    }

    //Следующие 3 метода отвечают за смену текущего класса состояния окна
```

Окончание листинга Г.1

```
    public void ToInitialMenuState(){
        state = new InitialMenu();
        headerLabelText.text = "Главное меню";
    }
    public void ToManualMenuState(){
        state = new ManualMenu();
        headerLabelText.text = "Справочник";
    }
    public void ToSceneSelectionMenuState(){
        state = new SceneSelectionMenu();
        headerLabelText.text = "Выбор сцены";
    }

    //Вызывает метод Back текущего класса состояния
    public void Back(){
        state.Back(this);
    }

    //Перезапускает сцену
    public void RestartScene(){
        LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    //Обработчик события ShowHideMainMenuEvent. Открывает/закрывает окно меню
    public void OnEvent(ShowHideMainMenuEvent @event){
        if(MainMenu.activeSelf) MainMenu.SetActive(false);
        else MainMenu.SetActive(true);
    }

//Далее идут 3 класса состояния окна главного меню
public class InitialMenu : MainMenuState{
    public override void Back(MainMenuController controller){

    }
}

public class ManualMenu : MainMenuState{
    public override void Back(MainMenuController controller){

controller.MainMenu.transform.Find("InitialMenu").gameObject.SetActive(true);

controller.MainMenu.transform.Find("ManualMenu").gameObject.SetActive(false);
        controller.BackButton.SetActive(false);
        controller.ToInitialMenuState();
    }
}

public class SceneSelectionMenu : MainMenuState{
    public override void Back(MainMenuController controller){

controller.MainMenu.transform.Find("InitialMenu").gameObject.SetActive(true);

controller.MainMenu.transform.Find("ScenesSelectionMenu").gameObject.SetActive(fal
se);
        controller.BackButton.SetActive(false);
        controller.ToInitialMenuState();
    }
}
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД КОМПОНЕНТА ACTIONSLISTELEMENT

Листинг Д.1 – Исходный код ActionsListElement.cs

```
using UnityEngine;
using TMPro;
using System.Collections.Generic;

public class ActionsListElement : MonoBehaviour,
IEventReciever<PartActivationEvent>
{
    [SerializeField] private EventBusHolder busHolder;
    [SerializeField] private SimplePart part;
    [SerializeField] private List<TMP_Text> text;


    //Обработчик события PartActivationEvent. При активации детали, указанной в поле
    //SimplePart part стиль всего текста, приведенного в списке text меняется
    //на зачеркнутый
    public void OnEvent(PartActivationEvent @event){
        if(part == @event.part){
            foreach(TMP_Text txt in text){
                txt.fontStyle = FontStyles.Strikethrough;
            }
        }
    }
    private void OnEnable(){
        busHolder.EventBus.Register(this as IEventReciever<PartActivationEvent>);
    }
    private void OnDisable(){
        busHolder.EventBus.Unregister(this as IEventReciever<PartActivationEvent>);
    }
}
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД КОМПОНЕНТА TOOLTIPCONTROLLER

Листинг Е.1 – Исходный код TooltipController.cs

```
using UnityEngine;
using TMPro;

public class TooltipController : MonoBehaviour, IEventReciever<ShowTooltipEvent>
{
    [SerializeField] private GameObject tooltipMenu, tooltipUIViewportContent,
tooltipHeaderText;
    [SerializeField] private EventBusHolder busHolder;

    //Обработчик события ShowTooltipEvent. Вызывает окно подсказки, удаляет его
    //содержимое, загружает содержимое из префаба, меняет заголовок окна
    public void OnEvent(ShowTooltipEvent @event){
        if(!tooltipMenu.activeInHierarchy) {tooltipMenu.SetActive(true);}
        foreach(Transform t in
tooltipUIViewportContent.GetComponentsInChildren<Transform>()){
            if(t != null && !t.Equals(tooltipUIViewportContent.transform))
Destroy(t.gameObject);
        }
        GameObject tooltip =
Instantiate(Resources.Load<GameObject>(@event.tooltipContentPrefabPath),
tooltipUIViewportContent.transform);
        tooltip.transform.SetParent(tooltipUIViewportContent.transform);
        tooltipHeaderText.GetComponent<TMP_Text>().text = @event.tooltipHeader;
    }

    private void OnEnable(){
        busHolder.EventBus.Register(this as IEventReciever<ShowTooltipEvent>);
    }
    private void OnDisable(){
        busHolder.EventBus.Register(this as IEventReciever<ShowTooltipEvent>);
    }
}
```

ПРИЛОЖЕНИЕ Ж
ИСХОДНЫЙ КОД КОМПОНЕНТА TELEPORTATIONCONTROLLER

Листинг Ж.1 – Исходный код компонента TeleportationController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.InputSystem;
using UnityEngine.XR.Interaction.Toolkit;

public class TeleportationController: MonoBehaviour
{
    static private bool _teleportIsActive = false;

    public enum ControllerType
    {
        RightHand,
        LeftHand
    }

    public ControllerType targetController;
    public InputActionAsset inputAction;
    public XRRayInteractor rayInteractor;
    public TeleportationProvider teleportationProvider;
    private InputAction thumbstickInputAction;
    private InputAction teleportActivate;
    private InputAction teleportCancel;

    void Start()
    {
        rayInteractor.enabled = false;

        Debug.Log("XRI " + targetController.ToString());
        teleportActivate      =      inputAction.FindActionMap("XRI      "      +
targetController.ToString() + " Locomotion").FindAction("Teleport Mode Activate");
        teleportActivate.Enable();
        teleportActivate.performed += OnTeleportActivate;

        teleportCancel      =      inputAction.FindActionMap("XRI      "      +
targetController.ToString() + " Locomotion").FindAction("Teleport Mode Cancel");
        teleportCancel.Enable();
        teleportCancel.performed += OnTeleportCancel;

        thumbstickInputAction     =     inputAction.FindActionMap("XRI     "     +
targetController.ToString() + " Locomotion").FindAction("Move");
        thumbstickInputAction.Enable();

    }

    private void OnDestroy()
    {
        teleportActivate.performed -= OnTeleportActivate;
        teleportCancel.performed -= OnTeleportCancel;
    }

    //В данном методе луч, отвечающий за телепортацию, включается по нажатию
    //соответствующей кнопки, после чего вызывается телепортация и луч выключается
```

Окончание листинга Ж.1

```
    void Update()
    {
        if (!_teleportIsActive)
        {
            return;
        }
        if (!rayInteractor.enabled)
        {
            return;
        }
        if (thumbstickInputAction.triggered)
        {
            return;
        }
        if (!rayInteractor.TryGetCurrent3DRaycastHit(out RaycastHit raycastHit))
        {
            rayInteractor.enabled = false;
            _teleportIsActive = false;
            return;
        }

        TeleportRequest teleportRequest = new TeleportRequest()
        {
            destinationPosition = raycastHit.point,
        };

        teleportationProvider.QueueTeleportRequest(teleportRequest);

        rayInteractor.enabled = false;
        _teleportIsActive = false;
    }

    //Метод обработки события ввода включения «телепорта»
    private void OnTeleportActivate(InputAction.CallbackContext context)
    {
        if (!_teleportIsActive)
        {
            rayInteractor.enabled = true;
            _teleportIsActive = true;
        }

    }

    //Метод обработки события ввода выключения «телепорта»
    private void OnTeleportCancel(InputAction.CallbackContext context)
    {
        if (_teleportIsActive && rayInteractor.enabled == true)
        {
            rayInteractor.enabled = false;
            _teleportIsActive = false;
        }

    }
}
```