

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА
на тему
Приложение химической лаборатории с
использованием технологии виртуальной
реальности

Студент	_____	Казёнкин С.М.
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	Попова Н.А.
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	Такташкин Д.В.
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	Макарычев П.П.
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	Попова Н.А.
	(подпись)	(фамилия, инициалы)

Пенза, 2020

Реферат

Пояснительная записка содержит 106 листов, 16, таблиц, 25 рисунков, 7 использованных источников и 4 приложения.

ВИРТУАЛЬНАЯ РЕАЛЬНОСТЬ, UNITY3D, ВИРТУАЛЬНАЯ ХИМИЧЕСКАЯ ЛАБОРАТОРИЯ, C#.

Объект исследования: лаборатория для изучения химии.

Цель работы: разработка образовательного мобильного приложения для изучения химии путём выполнения заданий по проведению химических реакций.

Средства разработки: Unity3d, C#, Visual Studio 2017, Blender.

Результаты работы: спроектировано и разработано образовательное мобильное приложение для изучения химии.

Область применения: средние общеобразовательные учреждения.

					ПГУ 09.03.02–8ВР131.19 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	«Приложение химической лаборатории с использованием технологии виртуальной реальности» Пояснительная записка	Лит.	Лист	Листов
Разраб.		Казёнкин С.М					4	106
Провер.		Попова Н.А.						
Н. Контр.		Такташкин Д.В.				Группа 16ВИ1		
Утверд.		Макарычев П.П.						

Оглавление

Введение.....	7
1 Анализ требований и постановка задачи на разработку приложения виртуальной химической лаборатории	9
1.1 Анализ рынка соответствующих продуктов.....	9
1.2 Анализ функциональных требований.....	13
1.3 Анализ требований к пользовательскому интерфейсу	19
1.4 Анализ требований к средствам разработки	20
1.5 Постановка задачи	20
2. Проектирование приложения	22
2.1 Проектирование пользовательского интерфейса	22
2.2 Проектирование базы данных	23
2.3 Проектирование структуры программы.....	23
3. Реализация приложения химической лаборатории с использованием технологии виртуальной реальности	28
3.1 Реализация взаимодействия с сервером образовательной платформы Experiment.....	28
3.2 Реализация логики работы программы	29
3.3 Реализация работы приложения с джойстиком.....	31
4. Тестирование приложения виртуальной химической лаборатории.....	34
4.1 Структурное тестирование	34
4.1.1 Метод покрытия операторов	35
4.1.2 Метод покрытия условий	36
4.1.3 Метод покрытия маршрутов	36
4.2 Анализ качества кода	37
4.2.1 Метрики кода.....	37
4.2.2 SLOC-метрика	38
4.2.3 Метрика Холстеда	39

5. Планирование и расчёт бюджета разработки приложения виртуальной химической лаборатории	42
Заключение	49
Список использованных источников	50
Приложение А. Глоссарий	51
Приложение Б. Диаграммы	53
Приложение В. Листинг программы на языке C#	56
Приложение Г. Результаты структурного тестирования	105

Введение

В наше время растёт роль мобильных устройств в различных сферах жизнедеятельности человека, в том числе и в сфере образования. В следствие этого появляется множество различных технологий, одной из которых является виртуальная реальность. Виртуальная реальность – созданный техническими средствами мир, передаваемый человеку через его ощущения: зрение, слух, осязание и другие.

Виртуальная реальность может обеспечить пользователя личным пространством для образования и индивидуальной программой обучения, что может поспособствовать переходу образования на новый уровень. В данный момент, эта технология становится более популярной и востребованной, и, следовательно, возрастает потребность в качественных программных продуктах, использующих все её возможности.

Данная работа направлена на создание химической виртуальной лаборатории с использованием технологии виртуальной реальности «PeriodicTable», в которой пользователи смогут проводить различные реакции смешивая различные вещества, при этом не неся никакого вреда здоровью. Для достижения этой цели необходимо выполнить ряд задач, таких как, анализ требований к разработке приложений, на основе полученных требований спроектировать различные аспекты приложения, после чего произвести разработку программного обеспечения и выполнить его тестирование.

Актуальность выбранной темы исследования заключается в том, что технология виртуальной реальности достаточно широко применяются в различных отраслях экономики и могут помочь решить многие важные задачи науки и образования.

Выполненная работа предоставит возможность превратить скучный процесс изучения химии в увлекательную игру, позволяющую почувствовать себя настоящим химиком.

Все термины, сложные для понимания данной работы, находятся в приложении А.

1 Анализ требований и постановка задачи на разработку приложения виртуальной химической лаборатории

1.1 Анализ рынка соответствующих продуктов

Для поиска продуктов аналогичных тому который будет разработан в последствие, можно обратиться к интернет-магазину мобильных Android приложений, который называется Google Play. Данная площадка позволяет выйти разработчикам приложений на глобальный рынок с быстрым продвижением программного продукта, так как она доступна каждому пользователю смартфона.

После анализа приложений виртуальной реальности по изучению химии, для сравнительного анализа были выбраны следующие приложения: «MEL Chemistry VR уроки химии», «Chemistry simulator VR», «Chemistry VR – Cardboard», которые доступны бесплатно на площадке Google Play.

Приложение «MEL Chemistry VR уроки химии», представленное на площадке Google Play от разработчиков MEL Science, содержит набор курсов по химии, где технология виртуальной реальности используя погружение и метод научной игры, превращает обучение в увлекательный процесс познания химии. Благодаря этому увеличивается показатель заинтересованности предметом и уровень усваиваемости предмета.

Уроки в этом приложении доступны на нескольких языках, для платформ Google Cardboard и Google Daydream. Для охвата всех тем в приложение доступно на момент исследования 39 уроков, разбитых на 5 тем.

Уроки позволяют увидеть, например, как взаимодействуют атомы в различных веществах и объектах, или узнать, сколько нужно молекул газов чтобы создать давление в 1 атмосферу. Интерфейс приложения представлено на рисунке 1.



Рисунок 1 – Интерфейс приложения «MEL Chemistry VR уроки химии»

Приложение «Chemistry Simulator VR», представленное на площадке Google Play от разработчиков Graphic Seven Studio. Оно позволяет наблюдать взаимодействие некоторых химических элементов, таких как $\text{Pb}(\text{NO}_3)_2$, K_2CrO_4 и некоторых других.

Представленный программный продукт имеет ряд недостатков:

- во-первых, весь интерфейс приложения доступен только на английском языке;
- во-вторых, очень мало задействованных химических элементов и соединений;
- в-третьих, в приложение очень мало различных взаимодействий с пользователем.

Данное приложение подходит для изучения химии в средней общеобразовательной школе.

Интерфейс приложения представлен на рисунке 2.

Приложение «Chemistry VR – Cardboard» представленное на площадке Google Play от разработчиков ARLOOPA inc Augmented and Virtual Reality Apps. Оно позволяет проводить различные реакции в игровой форме путём соединения химических элементов.

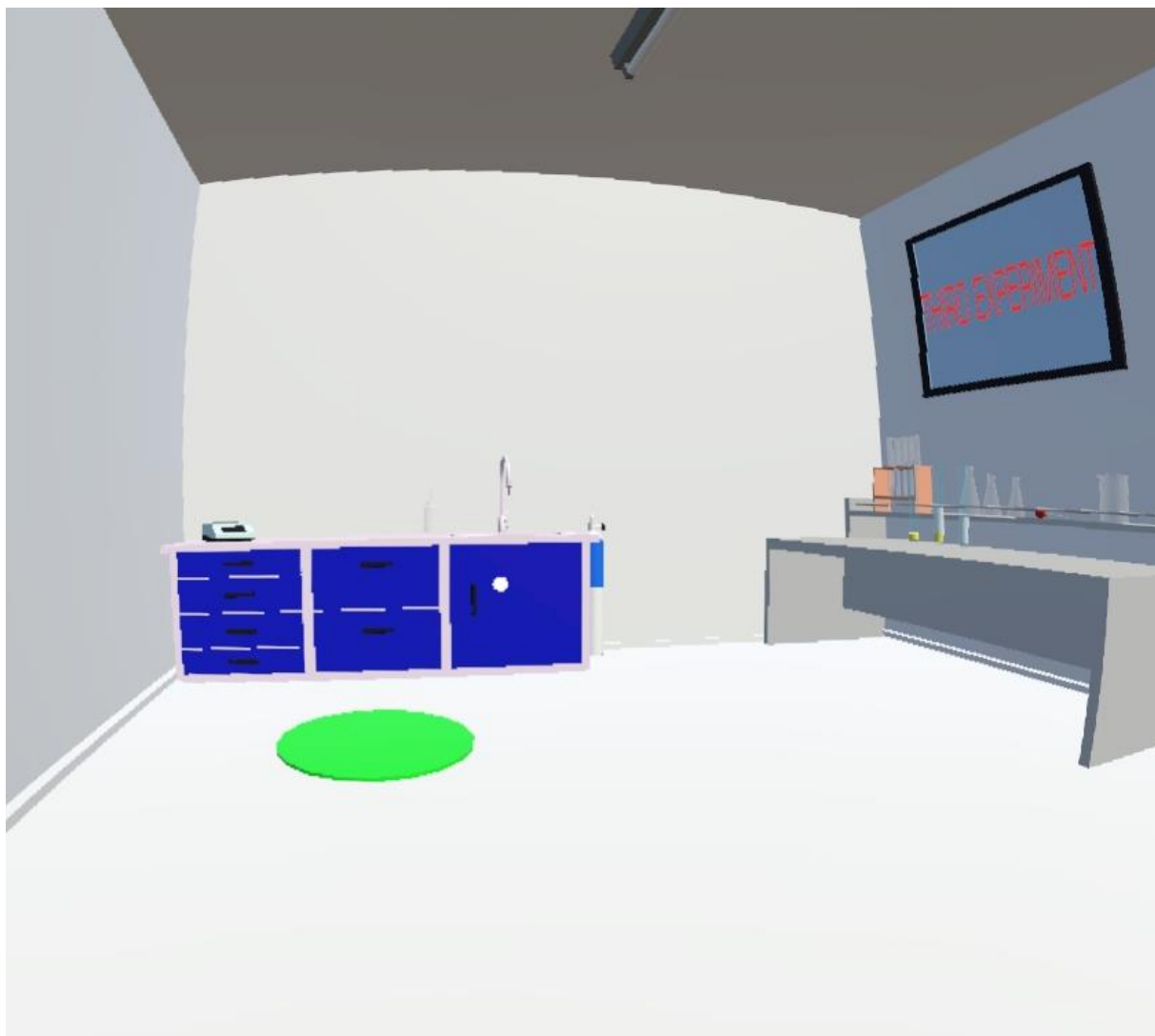


Рисунок 2 – Интерфейс приложения «Chemistry Simulator VR»

Основными недостатком этого приложения являются:

- во-первых, интерфейс приложения доступен только на английском;
- во-вторых, отсутствие эффектов проведения различных химических реакций.

Данное приложение отлично подходит для самого начального этапа изучения химии в общеобразовательных школах.

Пример работы приложения можно увидеть на рисунке 3.

Каждое из приведённых выше приложений имеет свой функционал и различные характеристики, поэтому для того чтобы вывести основные достоинства и недостатки каждого из них необходимо выявить основные критерии сравнения.



Рисунок 3 – Интерфейс приложения «Chemistry VR – Cardboard»

Выбор критериев производится по самым базовым и основным понятиям в химии: возможность проведения реакции; наличие таблицы Менделеева; наличие информации о химических элементах; наличие эффектов химических реакций.

Проведя сравнение программ-аналогов по представленным критериям все полученные результаты сведены в сравнительную таблицу 1.

Таблица 1 – Сравнение по критериям

Название приложения	Проведение реакций	Наличия таблицы Менделеева	Наличия информации о химических элементах	Эффекты проведения реакций
PeriodicTable	+	+	+	+
MEL Chemistry VR	-	+	+	-
Chemistry Simulator VR	+	-	-	+
Chemistry VR – Cardboard	+	-	-	-

По результатам проведённого анализа видно, что разрабатываемое приложение «PeriodicTable», обладает более расширенным функционалом.

1.2 Анализ функциональных требований

Разрабатываемое приложение предназначено для изучения химии путём проведения различных химических реакций и опытов в виртуальной реальности. Анализируя предметную область выделим некоторые функциональные требования:

- приложение должно предоставлять пользователям создавать и получать задания;
- приложения должно позволять взаимодействовать с различными объектами и химическими веществами;
- приложения должно позволять получать информацию о химических элементах;
- приложение должно предоставлять возможность проведение различных химических реакций.

Для описания основных вариантов использования пользователями, представлена диаграмма на рисунке 4.

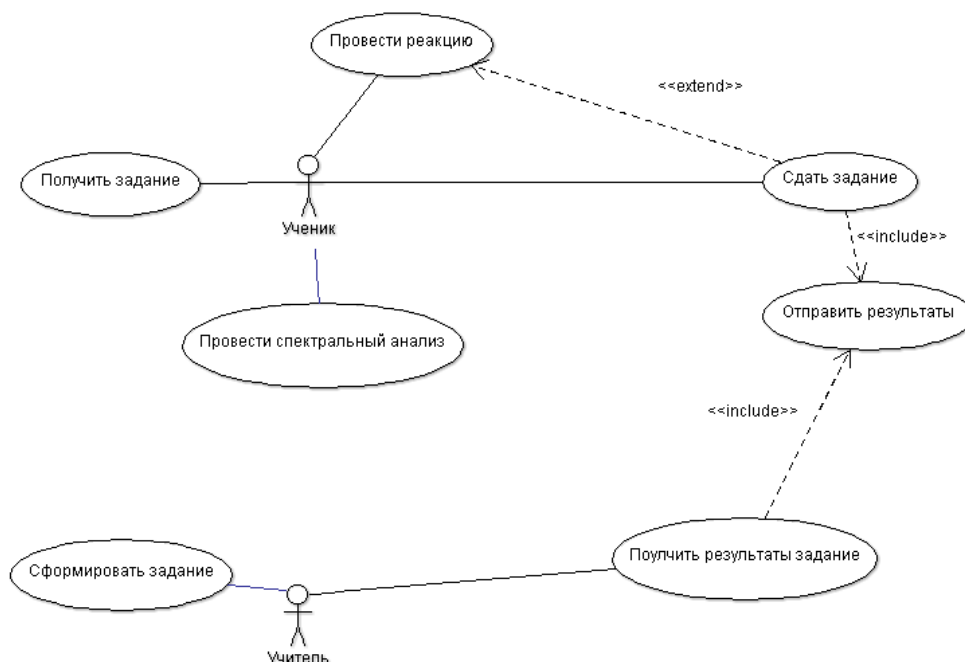


Рисунок 4 – Начальная диаграмма вариантов использования

Описание действующих лиц представлено в таблице 2.

Таблица 2 – Список «Действующих лиц»

Название	Профиль, подготовка и навыки
Ученик	Пользователь системы, который может получать задания, проводить различные реакции, добавляя химические элементы, вносить изменения в условия проведения реакции. Также ученик может отменять проведение реакции, добавлять различные реагенты, проводить спектральный анализ, а после отправлять результаты выполненного задания на сервер для проверки учителем.
Учитель	Пользователь системы, который формирует задания для ученика, и проверяет его выполнение.

Краткое описание вариантов использования представлено в таблице 3.

Таблица 3 – Краткое описание основных вариантов использования

Действующее лицо	Цель	Краткое описание
Ученик	Получить задание	Получить информацию о доступных химических элементах, инструментах, реагентах и описание задания
	Получить описание	Получение информации о том что надо сделать для выполнения работы.
	Получить данные о химических элементах	Получение информации о атомах.
	Получить данные о реагентах	Получение информации о готовых химических соединениях, которые могут ускорить протекание реакций(катализаторы) и/или некоторые промежуточные соединения.
	Получить данные о инструментах	Получение информация о необходимых инструментах для проведения реакции.
	Провести реакцию	Получение химических соединений на основе химических элементов, реагентов и различных условий проведения реакций.
	Добавить химический элемент	Добавление химических элементов в колбу с указанием молярной массы и концентрации.
	Внести изменения условия реакции	Ввод температуры, освещённости, давление и др.
	Добавить реагент	Добавление выданных химических соединений.
	Получить химическое соединение	Отправка внесённых данных на сервер получение готового химического соединения.
	Сдать задание	Отправка данных о химических соединениях для проверки выполнения задания.
	Провести спектральный анализ	Получение информации о химическом элементе или соединении.
Учитель	Сформировать задание	Указание необходимых средств для выполнения реакций и создания описания для выполнения работы.
	Получить результат задания	Получение результатов задания для проверки.

В таблице 4 описывается вариант использования «Получить задание», главный актёр в котором – «ученик».

Таблица 4 – Описание варианта использования «Получить задание»

Название варианта использования	Получить задание
Уровень	Цель пользователя
Область действия	Часть информационной системы для учеников
Участники и интересы	Ученик собирается получить задание с сервера от преподавателя
Основные действующие лица	Ученик
Второстепенные действующие лица	-
Предусловие	Учитель сформировал задание
Постусловие	Ученик получил задание
Гарантия успеха	Ученик зарегистрировался на событие в приложении и получил задание.
Основной сценарий	<ol style="list-style-type: none"> 1. Ученик вошел в систему под своим именем 2. Обнаружил задание 3. Открыл задание для изучения 4. Получил информацию о нужных химических элементах и инструментах для проведения реакции 5. Готовится к выполнению задания
Альтернативные потоки	<ol style="list-style-type: none"> 1. Сервер недоступен 2. Учитель не предоставил доступ к заданию для ученика

В таблице 5 описывается вариант использования «Провести реакцию», главный актёр в котором – «ученик».

Таблица 5 – Описание варианта использования «Провести реакцию»

Название варианта использования	Провести реакцию
Уровень	Цель пользователя
Область действия	Область информационной системы для учеников
Участники и интересы	Ученик начинает проведение реакции, целью которой является получение химического элемента, указанного в задании

Продолжение таблицы 5

Название варианта использования	Провести реакцию
Основные действующие лица	Ученик
Второстепенные действующие лица	-
Предусловие	Ученик получил задание
Постусловие	Получено химическое соединение
Гарантия успеха	При проведении реакции не было допущено ошибок и ученик получил требуемый в задании химическое соединение
Основной сценарий	<ol style="list-style-type: none"> 1. Ученик выбирает химические элементы для реакции 2. Добавляет их в колбу, при этом указывает молярную массу, концентрацию. 3. Выбирает предложенные учителем инструменты 4. Изменяет условия для проведения реакции. 5. Нажимает на кнопку «Провести реакцию» 6. Получает новое химическое соединение.
Альтернативные потоки	<ol style="list-style-type: none"> 1. Реакция протекает неверно, отменить проведение реакции. 2. При необходимости добавить реагенты 3. Сервер не исправен 4. Учитель изменил задание

В таблице 6 описывается вариант использования «Провести спектральный анализ», главный актёр в котором – «ученик».

Таблица 6 – Описание варианта использования «Провести спектральный анализ»

Название варианта использования	Провести спектральный анализ
Уровень	Цель пользователя
Область действия	Область информационной системы для учеников
Участники и интересы	Ученик хочет больше узнать о химическом элементе из таблицы Менделеева или о реагенте

Продолжение таблицы 6

Название варианта использования	Провести спектральный анализ
Основные действующие лица	Ученик
Второстепенные действующие лица	-
Предусловие	Ученик выбрал химические элементы для реакции и хочет о них узнать
Постусловие	Ученик выяснил всю необходимую информацию
Гарантия успеха	Информация об элементе или реагенте получена
Основной сценарий	<ol style="list-style-type: none"> 1. Кликнуть на элемент таблицы Менделеева 2. Поместить выбранный элемент в спектрометр 3. Повернуть камеру на телевизор 4. Получить данные спектрального анализа
Альтернативные потоки	<ol style="list-style-type: none"> 1. Повернуть камеру на колбы с реагентами 2. Выбрать реагент и поместить в спектрометр 3. Получить данные реагента

В таблице 7 описывается вариант использования «Сформировать задание», главный актёр в котором – «учитель».

Таблица 7 – Описание варианта использования «Сформировать задание»

Название варианта использования	Сформировать задание
Уровень	Цель пользователя
Область действия	Область информационной системы для преподавателя
Участники и интересы	Учитель собирается сформировать список возможных инструментов и химических элементов для ученика, что будет являться выбором ученику с помощью чего проводить реакцию. Ставится задача получить заданное учителем соединение.
Основные действующие лица	Учитель
Второстепенные действующие лица	-
Предусловие	Учитель вошел в систему
Постусловие	Учитель сформировал задание для ученика
Гарантия успеха	Задание выложено на сервер

Продолжение таблицы 7

Основной сценарий	1. Учитель нажимает на кнопку «Войти в систему» 2. На главной странице учитель указывает те инструменты, которые будут доступны ученику для выполнения задания 3. Указывает химические элементы, которые будут доступны ученику для выполнения задания 4. Указывает реагенты, которые будут доступны ученику для выполнения задания 5. Создает описание задания с перечисленными выше доступными вещами
Альтернативные потоки	1. Ошибка при входе в систему

Подробная диаграмма вариантов использования приведена в приложении Б.

1.3 Анализ требований к пользовательскому интерфейсу

Требования к пользовательскому интерфейсу являются одними из самых основных, которые определяют частично функционал, а частично внешний вид приложения.

Функциональные требования:

- объекты должны соответствовать физическим ограничениям, действующим в реальном мире;
- обязательная поддержка физического взаимодействия объектов, т.е. использование пространственных отношений [1].

Требования дизайна:

- наличие системы «общения» приложения с пользователем;
- меню должно быть написано на языке доступном пользователю;
- наличие развернутого пользовательского меню;
- наличие визуального представления таблицы Менделеева;
- наличие инструментов для проведения химических реакций.

1.4 Анализ требований к средствам разработки

На основе изучения различных средств для разработки приложения с технологией виртуальной был выведен ряд требований:

- обеспечение высокого качества графического представления информации;
- библиотека 3D объектов имеет возможность работать с современными форматами трехмерной графики – *.3ds, *.dae, *.fbx, *.flt;
- поддержка языков программирования высокого уровня для обеспечения эффективного процесса разработки;
- наличие редактора программной и графической разработки объектов;
- возможность подключения сторонних библиотек объектов [2].

Исходя из требований в качестве движка был выбран Unity3d, который доступен бесплатно для некоммерческих целей. В качестве расширения функционала и подключения дополнительных библиотек выбран Visual Studio 2017.

Для разработки 3D объектов был выбран редактор трехмерной графики Blender. Основными достоинствами этой программы служат большое количество инструментов для моделирования различных объектов, простота использования и большое количество инструкций использования.

1.5 Постановка задачи

Для разработки химической лаборатории с использованием технологии виртуальной реальности «PeriodicTable», необходимо выделить ряд задач:

- изучение различной литературы для разработки приложения;
- проектирование приложения виртуальной химической лаборатории;
- создание различных моделей для создания лаборатории;
- создание структуры приложения;
- разработки приложения виртуальной реальности;
- тестирование приложения.

После выполнения поставленных задач будет создана и оформлена документация по приложению «PeriodicTable».

2 Проектирование приложения

2.1 Проектирование пользовательского интерфейса

Пользовательский интерфейс – это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы программной системы [3].

Основываясь на требования к пользовательскому интерфейсу были разработаны следующие модели в Blender:

- комната;
- химический стол;
- периодическая система Менделеева;
- шкафы;
- колбы;
- штатив с пробирками;
- спиртовка;
- спектрометр.

Концепт пользовательского интерфейса с использованием разработанных моделей представлен на рисунке 5.

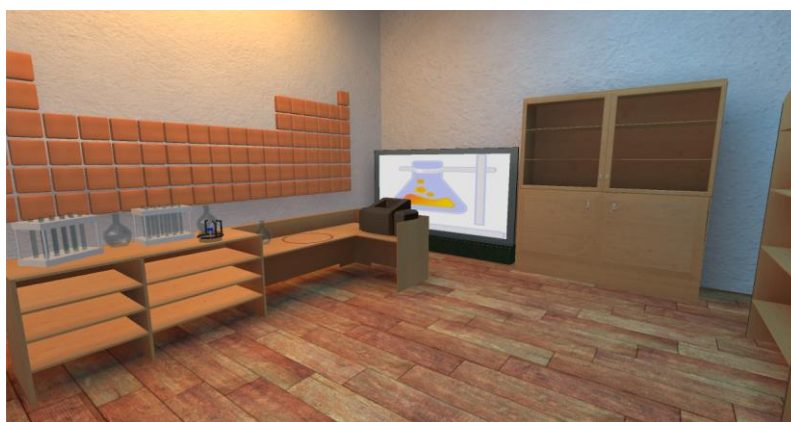


Рисунок 5 – Концепт пользовательского интерфейса

Так же были разработаны модели для каждого элемента таблицы Менделеева.

Для добавления материалов кнопок и их корректного отображения согласно периодической таблице Менделеева, в последствие будет добавлен скрипт, выполняющийся при запуске программы. Он автоматически задаст материал каждой кнопки, добавит информацию об элементах в зависимости от данных из базы данных.

2.2 Проектирование базы данных

Основываясь на функциональных требованиях приложения можно выделить следующие требования к структуре базы данных:

- наличие информации о химических элементах;
- наличие информации о различных реакциях, то есть в базе данных должна храниться информация о химических веществах, участвующих в реакции, условия проведения реакции и эффекты проходящие при прохождении реакции;
- необходимо хранить информацию о заданиях, которые необходимо выполнить.

В соответствие с этими требованиями была построена логическая модель данных представленная на рисунке 6.

Физическая модель данных приведена в приложении Б.

2.3 Проектирование структуры программы

На основе составленных требований была составлена диаграмма классов химической лаборатории с использованием технологии виртуальной реальности. Она представлена на рисунке 7.

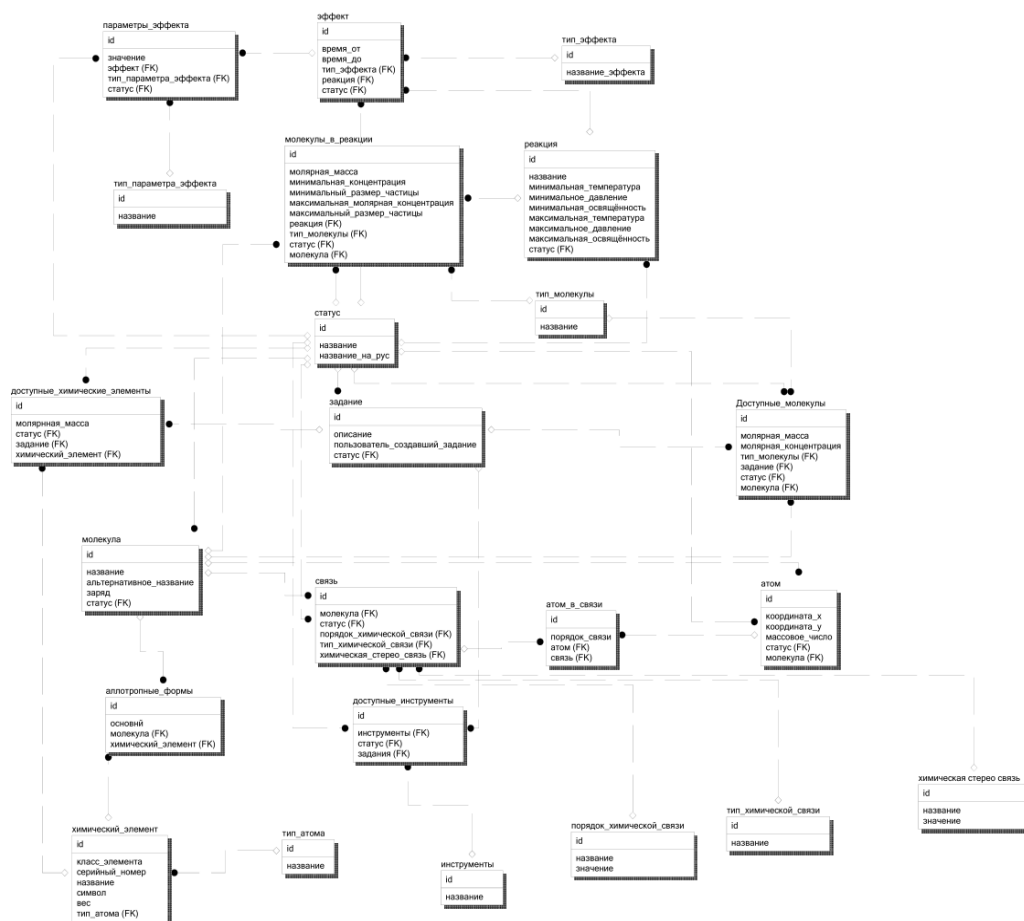


Рисунок 6 – Логическая модель данных

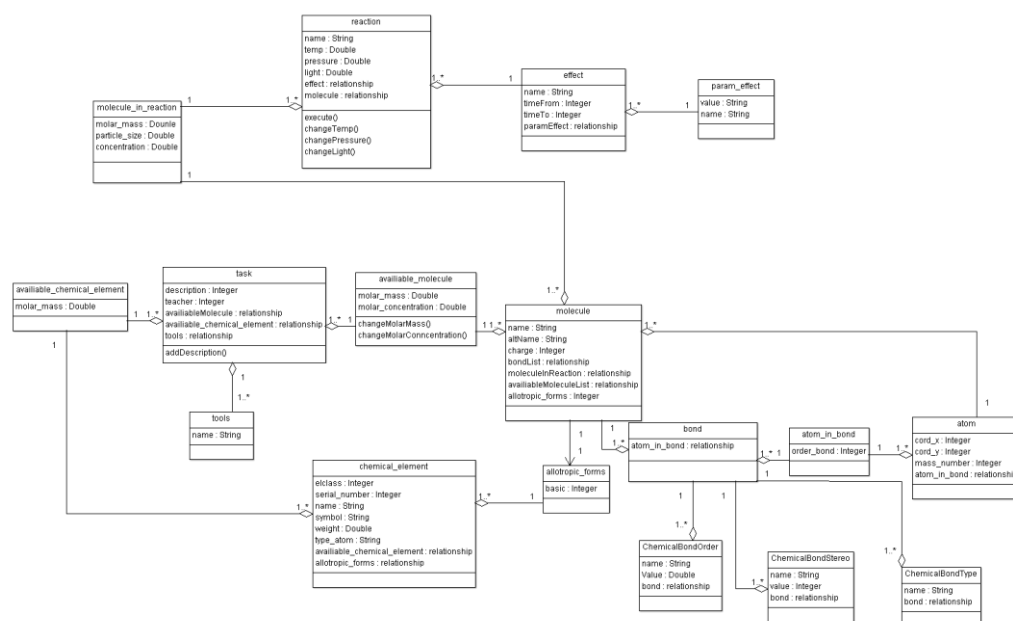


Рисунок 7 – Диаграмма классов химической лаборатории

Диаграмма деятельности отображает последовательность действий необходимых для создания задания учителем, получения, выполнения и сдачи задания учеником и последующей проверки правильности выполнения. Деятельность начинается с формирования задания учителем. После чего ученик может получить это задание и выполнить его проводя различные реакции. При получении готового соединения ученик может сдать задание на проверку для проверки учителем. Диаграмма представлена на рисунке 8.

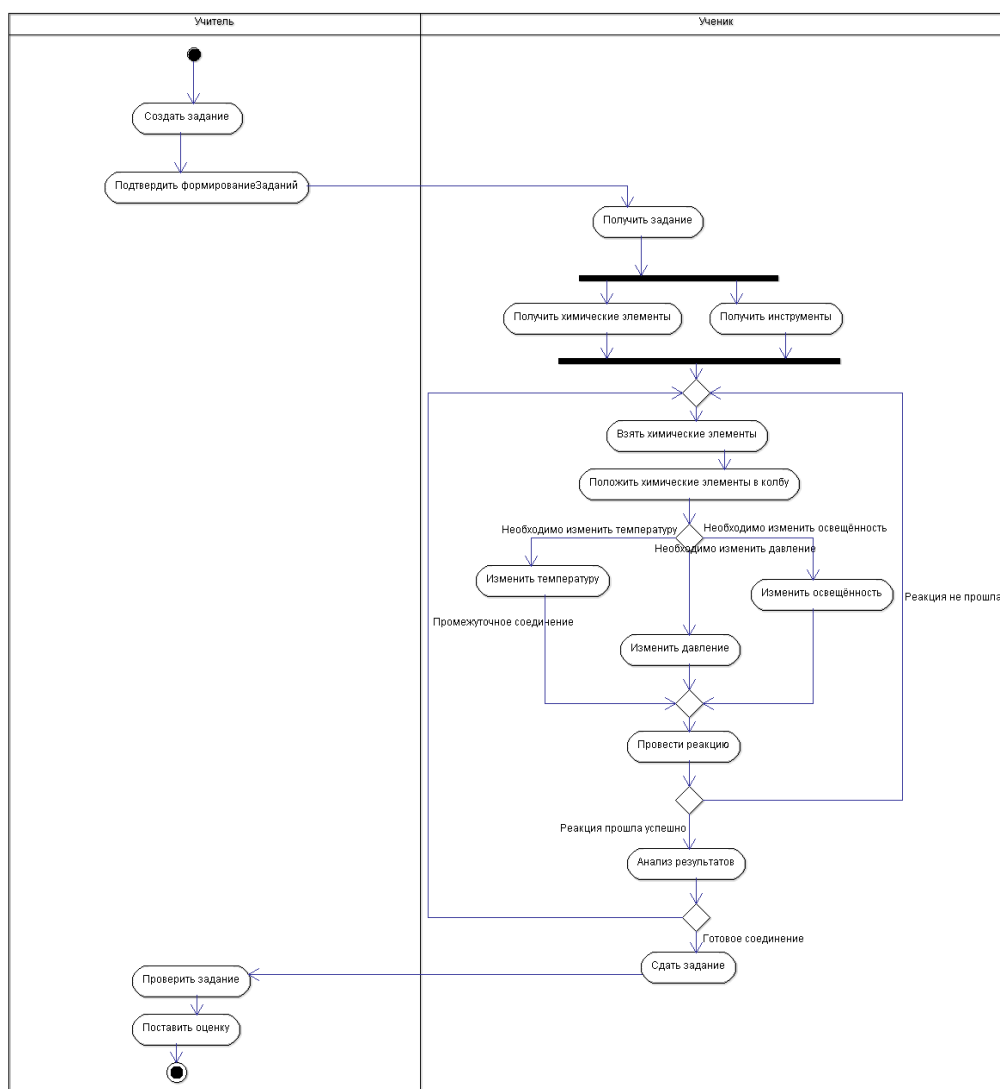


Рисунок 8 – Диаграмма деятельности системы

Теперь составим диаграмму состояний, которая покажет нам все возможные состояния объекта класса «reaction». Для этого определим основные состояния объекта.

1. Создан объект класса. В этом состоянии объект только инициализирован конструктором.
2. Заполнены молекулы в реакции. Объект переходит в это состояние после внесения различной информации о молекулах в реакции, таких как молярная масса и концентрация вещества в реакции.
3. Заполнены условия проведения реакции. Объект находится в этом состоянии, когда пользователь изменяет условия проведения реакции, такие как температура, давление и уровень освещённости.
4. Реакция запущена. В этом состоянии объект находится тогда, когда ученик запускает реакцию и происходит анимации различных эффектов.

Диаграмма состояния представлена на рисунке 9.

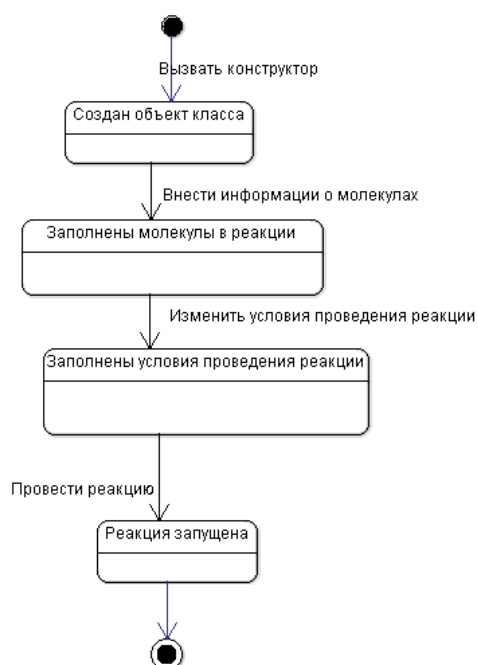


Рисунок 9 – Диаграмма состояния объекта класса «reaction»

Теперь давайте посмотрим принцип сдачи задания. Для этого построим диаграмму кооперации, в которой наглядно разберём взаимодействие различных объектов системы при сдаче задания. Диаграмма кооперации представлена на рисунке 10.

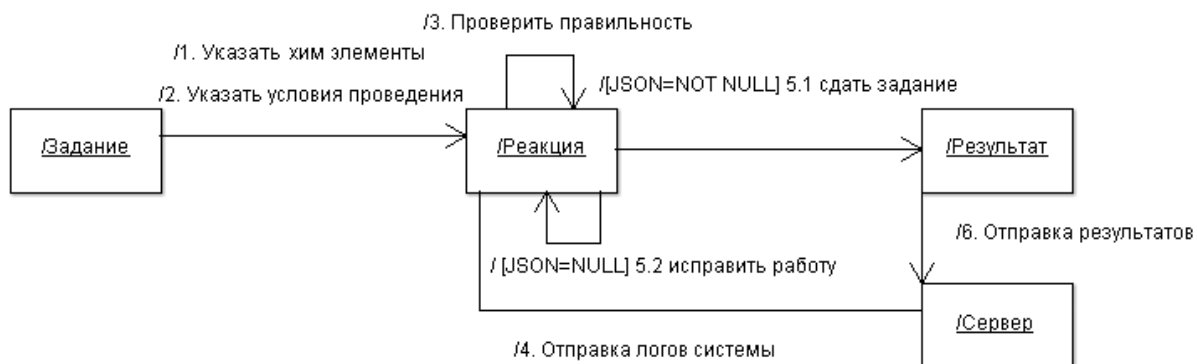


Рисунок 10 – Диаграмма кооперация для сдачи задания.

Рассмотрим основные различными взаимодействия при сдаче задания, сведённые в таблицу 8.

Таблица 8 – Взаимодействия объектов в диаграмме кооперации

№	Описание взаимодействия объектов
1	Из задания выбираются доступные химические элементы и помещаются в объект реакция.
2	На основе выданного задания и доступных инструментов в нём выставляются условия проведения реакции.
3	Делается запрос к БД и проверяется существования реакции.
4	Отправляются сообщение о проведённой реакции на сервер платформы Experiment.
5.1	Если результат представленный в виде JSON не является пустым, то формируется результат для сдачи задания.
5.2	Если результат представленный в виде JSON является пустым, то об этом сообщается пользователю который исправляет реакцию.
6	После формирования результата идёт отправка результатов на сервер платформы Experiment

Таким образом мы можем достаточно просто отправить результат задания на проверку.

3. Реализация приложения химической лаборатории с использованием технологии виртуальной реальности

3.1 Реализация взаимодействия с сервером образовательной платформы Experiment

Для начала реализации приложения необходимо рассмотреть взаимодействие с сервером Experiment, так как с ним связана авторизация пользователя, и соответственно получение заданий, получение информации о химических элементах, проведение реакции и сдача задания. Для этого рассмотрим процесс авторизации пользователя в приложении на примере диаграммы последовательности, изображённой на рисунке 11.

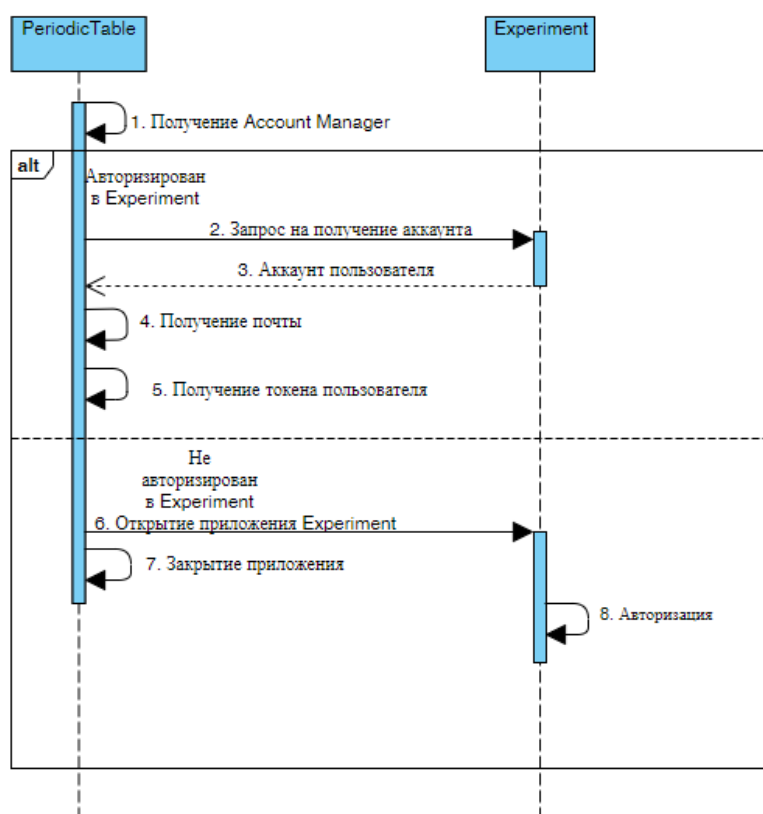


Рисунок 11 – Авторизация пользователя в системе

В данной диаграмме показано, что авторизация происходит с использованием AccountManager, с его помощью мы можем сделать запрос на получение аккаунта пользователя из мобильного приложения Experiment,

и соответственно выбрать из полученной информации почту и токен пользователя, которые необходимы для HTTP-запросов на сервер. Также если пользователь не авторизован в системе Experiment, то мы можем перенаправить пользователя на авторизацию.

3.2 Реализация логики работы программы

Для реализации возможностей VR в приложении используются технология Cardboard SDK для GVR от Google. Эта технология предоставляет всё необходимое для создания приложений виртуальной реальности. Cardboard SDK поддерживает функции VR, такие как:

- отслеживание движения;
- стереоскопическая визуализация;
- взаимодействие с пользователем [4].

Все эти возможности реализуются в приложение путём подключения различных компонентов к внутриигровому персонажу.

MonoBehaviour – это базовый класс от которого наследуются все скрипты [5]. На основе этого класса формируются все взаимодействия объектов в приложении.

Рассмотрим реализацию логики работы на примере скрипта «SpiritLamp», представленным на рисунке 12.

Метод «Start» вызывается когда скрипт включён первым вызовом любого из методов обновления. Данный метод используется для инициализации свойства отвечающее за смещения координаты колбы при помещении её на спиртовку.

Метод «OnTriggerEnter», вызывается когда сталкиваются 2 игровых объектов. Этот метод используется для того, чтобы смещать и фиксировать положение колбы при взаимодействии со спиртовкой.

```

public class SpiritLamp : MonoBehaviour {
    private bool spiritLamp;
    public Transform transformFlask;
    private Vector3 position = new Vector3();
    // Use this for initialization
    void Start () {
        GameObject Flask = GameObject.Find("ReactionFlask");
        position = Flask.GetComponent<Transform>().position;
        position.y = 1.256f;
    }
    private void OnTriggerEnter(Collider other){
        if (other.tag == "ReactionFlask" && !spiritLamp){
            if (!gameObject.GetComponent<DragableObject>().drag){
                GameObject Flask = GameObject.Find("ReactionFlask");
                GameObject.Find("MenuReaction").transform.Find("Condition").Find("Heat").gameObject.SetActive(true);
                spiritLamp = true;
                gameObject.GetComponent<Transform>().position = position;
                gameObject.transform.rotation = Flask.transform.rotation;

                transformFlask.position = position;
                Flask.GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeAll;
                gameObject.GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeRotation | RigidbodyConstraints.FreezePosition;
                gameObject.GetComponent<Rigidbody>().freezeRotation = true;
            }
        }
    }
    private void OnTriggerStay(Collider other){
        if (other.tag == "ReactionFlask"){
            if (!gameObject.GetComponent<DragableObject>().drag){
                spiritLamp = true;
                position.y -= 0.23f;
                gameObject.transform.position = position;
                position.y += 0.23f;}}}
    // Update is called once per frame
    void Update () {
        if(gameObject.GetComponent<DragableObject>().drag){
            if (spiritLamp){
                GameObject.Find("ReactionFlask").GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeRotation | RigidbodyConstraints.FreezePositionX | RigidbodyConstraints.FreezePositionZ;
                gameObject.GetComponent<Rigidbody>().constraints = RigidbodyConstraints.None;
                spiritLamp = false;
                gameObject.transform.Find("Fire").gameObject.SetActive(false);
                Reactions reactions = GameObject.Find("MenuReaction").GetComponent<StartReaction>().reactions;
                GameObject.Find("MenuReaction").transform.Find("Condition").Find("Heat").gameObject.SetActive(false);
                if (reactions != null){
                    reactions.temp = 0;}}}
            if (spiritLamp == false){
                GameObject.Find("ReactionFlask").GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeRotation | RigidbodyConstraints.FreezePositionX | RigidbodyConstraints.FreezePositionZ;
                gameObject.GetComponent<Rigidbody>().constraints = RigidbodyConstraints.None;
            }
        }
    }
}

```

Рисунок 12 – Реализация скрипта «SpiritLamp»

Метод «OnTriggerStay» который вызывается практически каждый кадр для каждого объекта, которые соприкасаются с триггером. Этот метод работает для фиксации колбы относительно спиртовки.

Метод «Update» вызывается каждый кадр, если MonoBehaviour включен. Этот метод позволяет отслеживать нахождение спиртовки под колбой и добавление в условия проведения реакции процесс нагревания. Также он позволяет отслеживать что при условии пользователь взял спиртовку для переноса в другое место, снималась фиксация колбы и прекращение действия меню для изменения температуры в условии проведения реакции.

Все эти методы унаследованны от базового класса MonoBehaviour. Вся логика работы программы строится на использование таких методов, и некоторых глобальных переменных, например, переменные типа GameObject которые позволяют обращаться к различным компонентам, свойствам, таким как, свойство Transform отвечающего за положение объекта в игровом пространстве, каждого объекта в приложении.

3.3 Реализация работы приложения с джойстиком

Для реализации управления в приложении и взаимодействия с интерфейсом в разрабатываемом приложении используется джойстик. В данном приложении используются 2 кнопки и аналоговый стик для управления передвижения в виртуальной химической лаборатории, представленные на рисунке 13.

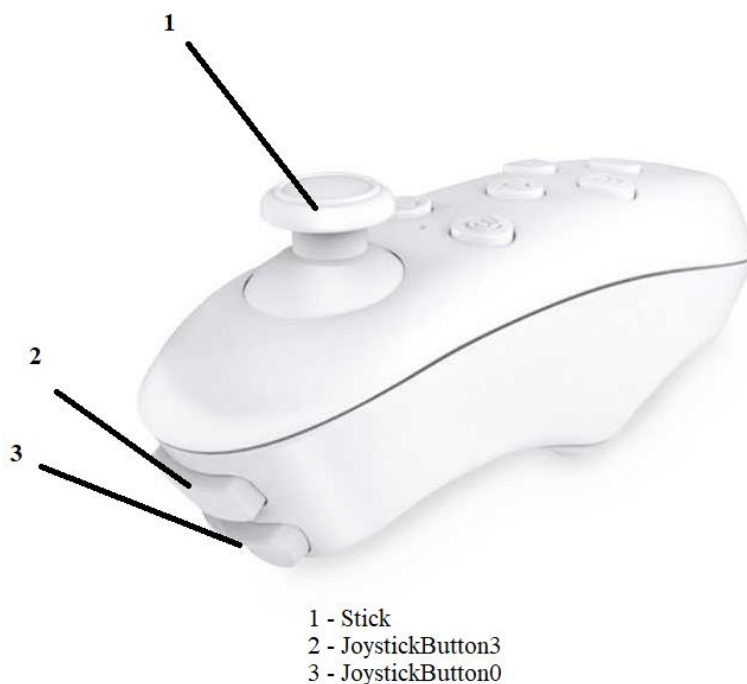


Рисунок 13 – Используемые кнопки на джойстике

Обработчик «JoystickButton3» выполняет роль основной кнопки для взаимодействия с пользовательским интерфейсом, также позволяет переносить объекты, у которых есть компонент RigidBody, который отвечает за физику.

Обработчик «JoystickButton0» необходим для выполнения дополнительных действий, например, взятие пробы какого-либо химического элемента, и в последствие добавление его в реакцию.

Рассмотрим логику обработчика на примере «JoystickButton3» представленный на рисунке 14.

```

private bool cursor;
// Use this for initialization
void Start()
{
    cursor = false;
}

// Update is called once per frame
void Update()
{
    if ((Input.GetKeyDown(KeyCode.Mouse2) || Input.GetKeyDown(KeyCode.JoystickButton3)) && cursor)
    {
        try
        {
            gameObject.GetComponentInChildren<Button>().onClick.Invoke();
            if (!gameObject.activeInHierarchy)
            {
                cursor = false;
            }
        }
        catch (Exception e)
        {
            try
            {
                gameObject.GetComponentInChildren<CreateElements>().CreateElement();
            }
            catch (Exception ex)
            {
                Debug.Log(ex.Message);
            }
        }
    }
}

public void OnPointerExit(PointerEventData eventData)
{
    cursor = false;
}

public void OnPointerEnter(PointerEventData eventData)
{
    cursor = true;
}

```

Рисунок 14 – Обработчик «JoystickButton3»

Переменная `cursor` имеет булевский тип со значением по умолчанию `false`, и отвечает за наведение на объект на котором находится курсор, то есть если курсор наводится на объект, с которым можно взаимодействовать срабатывает метод `OnPrinterEnter` переводя эту переменную в значение `true`, и выведения с объекта срабатывает метод `OnPrinterExit`, переводя переменную в значение `false`. Также эту переменную можно перевести в значения `false`, когда выполняется условие, что один из объектов родителей и соответственно сам объект не активен. Это сделано для того чтобы не было случайных взаимодействий с объектами которых нет в химической лаборатории в момент срабатывания обработчика. Соответственно обработчик будет срабатывать только при наведении на активные объекты курсора и при выполнении условия что нажата кнопка `KeyCode`, которой равен `JoystickButton3` или `Mouse2`.

Обработчик «Stick» предназначен для передвижения учеником по виртуальной лаборатории. Он работает аналогично другим обработчикам, только с одним отличием он генерирует четыре кода клавиша в зависимости от направления наклона, за счёт этого у нашего персонажа меняется координаты по двум осям.

4 Тестирование приложения виртуальной химической лаборатории

4.1 Структурное тестирование

Структурное тестирование или тестирование по принципу белого ящика характеризуется тем, что разработчик теста имеет доступ к коду программы [6]. Данная методология предполагает несколько методов тестирования: метод покрытия условий, метод покрытия операторов, метод покрытия решений и метод покрытия маршрутов.

Проведём тестирование на основе алгоритма «открытия меню для проведения реакции». Блок-схема алгоритма представлена на рисунке 15.

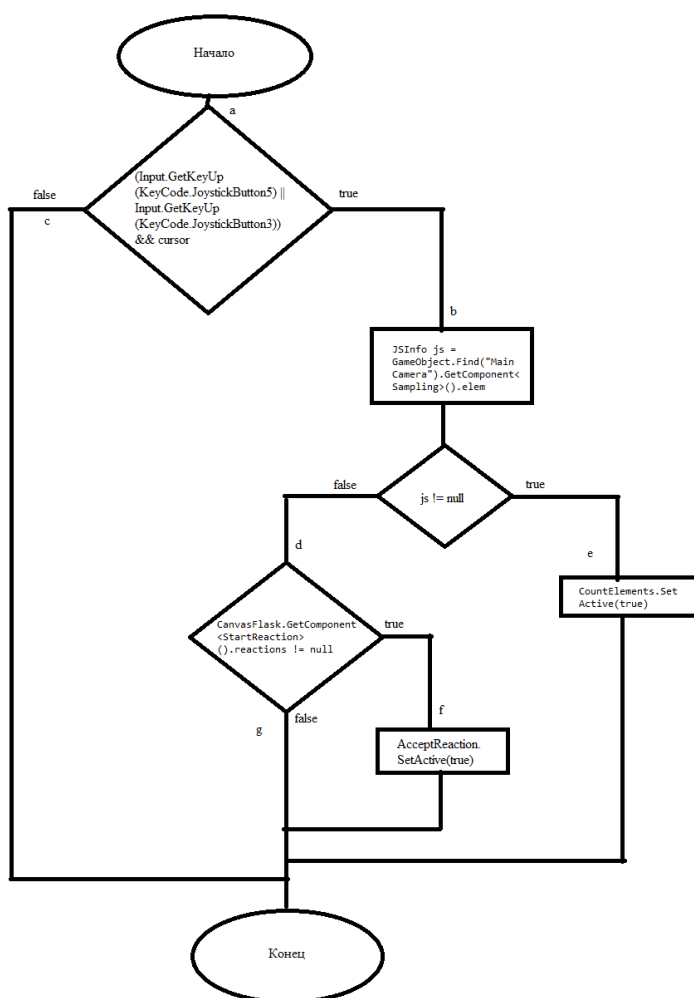


Рисунок 15 – Блок-схема открытия меню для проведения реакций

Управляющий граф алгоритма, который отображает последовательность операторов, представлен на рисунке 16.

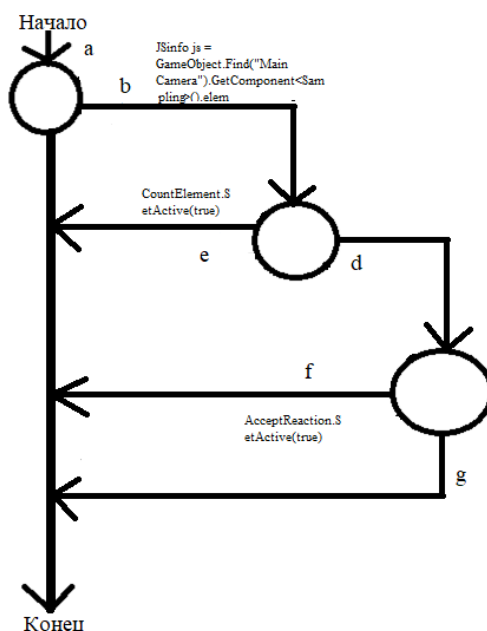


Рисунок 16 – Управляющий граф алгоритма

Управляющий отображает поток управления программы.

4.1.1 Метод покрытия операторов

Метод покрытия операторов подразумевает выполнение каждого оператора хотя бы один раз.

Для покрытия операторов необходимо 2 теста представленных в таблице 9.

Таблица 9 – Тесты метода покрытия операторов

Маршрут	Тест	Фактический результат	Результат тестирования
a,b,e	Пользователь взял пробу элемента и добавляет её в колбу.	Открывается меню с добавлением вещества.	Успешно
a,b,d,f	Пользователь не взял пробу и пытается вызвать меню проведения реакция с не пустой колбой.	Появляется меню проведения реакции.	Успешно

Все тесты прошли успешно, при этом каждый оператор выполнен, по крайней мере, один раз.

4.1.2 Метод покрытия условий

Для выполнения метода покрытие условий необходимо такое количество тестов, чтобы все возможные результаты каждого условия выполнялись по крайней мере один раз.

Для этого необходимо 2 теста представленных в таблице 10.

Таблица 10 – тесты метода покрытия условий

Маршрут	Входные данные	Фактический результат	Результат тестирования
a,c	Input.GetKeyUp(KeyCode.JoystickButton5)== true, Input.GetKeyUp(KeyCode.JoystickButton3) == false, cursor==false, js!=null, CanvasFlask.GetComponent<StartReaction>(). reactions == null	Не отображается ни одно меню.	Успешно
a,b,d,f	Input.GetKeyUp(KeyCode.JoystickButton5)== false, Input.GetKeyUp(KeyCode.JoystickButton3) == true, cursor==true, js==null, CanvasFlask.GetComponent<StartReaction>(). reactions != null	Отображается меню проведения реакции	Успешно

Все тесты прошли успешно, при этом каждое условие выполнилось хотя бы 1 раз, по крайней мере, один раз.

4.1.3 Метод покрытия маршрутов

Для выполнения метода покрытия маршрутов необходимо составить тесты, которые будут проходить по каждому пути в алгоритме хотя бы один раз.

Для покрытия маршрутов было составлено 4 теста, представленных в таблице 11.

Таблица 11 – Тесты для метода покрытия маршрутов

Маршрут	Тест	Фактический результат	Результат тестирования
a,c	Пользователь не пытается взаимодействовать с колбой для открытия меню.	Не отображается ни одно меню.	Успешно
a,b,e	Пользователь взял пробу элемента и добавляет её в колбу.	Открывается меню с добавлением вещества.	Успешно
a,b,d,f	Пользователь не взял пробу и пытается вызвать меню проведения реакция с не пустой колбой.	Появляется меню проведения реакции.	Успешно
a,b,d,g	Пользователь не взял пробу и пытается взаимодействовать с пустой колбой	Не появляется ни одно меню	Успешно

Все тесты прошли успешно, при этом были пройдены все маршруты.

4.2 Анализ качества кода

4.2.1 Метрики кода

Метрики кода позволяют оценить качество и другие свойства определенного участка программного кода системы или приложения. Метрики кода являются совокупностью оценок ПО, служащей рекомендательными характеристиками и представляющую более полную информацию о программном коде [6].

Расчёт метрик для данной работы будут производиться на основе алгоритма получения заданий.

Метрики программного кода делятся на несколько групп:

- количественные метрики(Source Lines of Code - SLOC-метрика);
- метрики сложности потока управления программы (метрики Холстеда, цикломатическая сложность Маккейба);
- метрики сложности потока данных (метрика Чепина).

Рассчитаем некоторые из этих метрик на конкретных примерах.

4.2.2 SLOC-метрика

SLOC-метрика – это метрика программного кода используемая для измерения его объёма, путём расчёта количества строк. Все строки исходного кода разделяют на 2 типа: физические и логические строки.

Физические строки – это общее количество строк кода, включая комментарии и пустые строки.

Логические строки – это число команд в исходном коде, оно зависит от языка программирования.

Результаты подсчёта данной метрики представлены в таблице 12.

Таблица 12 – Расчёт SLOC-метрики

Физические строки	27
Логические строки	16
Пустые строки	4
Строки комментариев	1

Расчёт SLOC-метрики производился для класса «Tasks» представленного в приложение В.

4.2.3 Метрика Холстеда

Метрика Холстеда состоит из четырёх измеряемых характеристик программы:

- число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
- число уникальных операндов программы (словарь операндов);
- общее число операторов в программе;
- общее число операндов в программе.

На основе этих характеристик можно рассчитать следующие оценки:

- словарь программы;
- длина программы;
- объём программы;
- сложность программы;
- оценка усилия программиста.

Рассчитаем метрику Холстеда для программного кода получения задания (приложение В). Результат данной метрики продемонстрирован в таблице 13.

Таблица 13 – Операторы и операнды для метрики Холстеда

Операторы	Число вхождений	Операнды	Число вхождения
Public	2	Start	1
Private	1	UnityEngine	1
Class	1	Tasks	1
Static	1	MonoBehaviour	1
If	2	string	1
Void	1	task	2
=	8	taskId	3
==	1	int	2
New	1	Application.platform	1

Продолжение таблицы 13

Операторы	Число вхождений	Операнды	Число вхождения
<>	5	RuntimePlatform.Android	1
UnityPlayer.GetStatic	1	AndroidJavaClass	1
currentActivity.Call	1	AndroidJavaObject	6
intent.Call	2	currentActivity	2
extras.Call	1	intent	3
SendQuery.GET	1	UnityPlayer	1
Application.LoadLevel	1	Bool	2
()	11	hasExtra	2
		extras	2
		SendQuery	1
		Application	2
		0	1
		12-уникальных строковых констант	12
n1=17	N1=41	n2=33	N2=49

На основании представленных характеристик можно рассчитать следующие оценки.

Длина программы рассчитывается по формуле:

$$N = N1 + N2.$$

Словарь программы рассчитывается по формуле:

$$n=n1+n2.$$

Вычислительная сложность рассчитывается по формуле:

$$V = N * \log_2(n).$$

Объём программы по Холстеду определяет объём запрограммированного кода. Размер метрики для одной функции должен находиться в пределах от 20 до 1000.

В результате расчёта оценок по метрики Холстеда получены следующие значения:

- 1) $N = 90$;
- 2) $n = 50$;
- 3) $V = 508$.

В нашем случае объём программы по Холстеду находится в пределах норм.

Теоретическая длина программы рассчитывается по формуле:

$$N' = n_1 * \log_2(n_1) + n_2 * \log_2(n_2).$$

Результат вычисления теоретической длины программы составил 235,951.

5 Планирование и расчёт бюджета разработки приложения виртуальной химической лаборатории

Управление проектом – это приложение знаний, опыта, методов и средств работы проекта для удовлетворения требований, предъявляемых к проекту, планирование, организация, контроль всех направлений [7].

В качестве продукта управления проектами был выбран Microsoft Project, позволяющей оптимизировать планирование проекта, а также наглядно показать весь процесс разработки на сетевом графе и диаграмме Ганта. Для реализации планирования в среде Microsoft Project необходимо сначала определить список задач. Список задач проекта их длительность, и данные о предшественниках приведены в таблице 14.

Таблица 14 – Список задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало реализация	-	-
2	Анализ требований и постановка задачи	-	-
3	Анализ рынка соответствующих продуктов	3	1
4	Анализ функциональных требований	1	3
5	Анализ требований к пользовательскому интерфейсу	2	3
6	Анализ требований к средствам разработки	1	4;5
7	Постановка задачи	1	3
8	Анализ предметной области и постановка задачи завершены	-	7
9	Проектирование приложения	-	2
10	Проектирование пользовательского интерфейса	10	7
11	Проектирование базы данных	8	10
12	Проектирование структуры программы	5	11
13	Проектирование завершено	-	12
14	Реализация приложения	-	9
15	Реализация структур данных	2	11
16	Реализация пользовательского интерфейса	5	15

Продолжение таблицы 14

№	Название задачи	Длительность	Предшественники
17	Реализация программных средств	20	16
18	Реализация завершена	-	17
19	Отладка и тестирование	-	14
20	Отладка	4	17
21	Структурное тестирование	5	17
22	Анализ качества кода	4	20
23	Отладка и тестирование завершены	-	19
24	Составление программной документации	10	22
25	Сдача в эксплуатацию	1	24
26	Проект завершён	-	25
27	Конец проекта	-	26

После определения списка работ требуется определить ресурсы, необходимые для выполнения задач проекта. Список ресурсов, их типов и соответствующих им затрат приведены в таблице 15.

Таблица 15 – Ресурсы и затраты проекта

№	Название ресурса	Тип ресурса	Затраты		Начисления
			Стандартная ставка	Ставка сверхурочных	
1	Программист	Трудовой	16800 р./мес	200 р./ч	Пропорциональное
2	Аналитик	Трудовой	15000 р./мес	150 р./ч	Пропорциональное
3	Тестировщик	Трудовой	16000 р./мес	200 р./ч	Пропорциональное
4	Бумага	Материальный	300,00 р.		Пропорциональное
5	Флеш-накопитель	Материальный	300,00 р.		
6	Телефон поддерживающий VR	Материальный	20000 р.		Пропорциональное
7	Компьютер	Материальный	60000 р.		Пропорциональное
8	Интернет	Затраты	-	-	Пропорциональное

Ресурсы, приведённые в таблице 15, необходимо распределить между задачами в соответствии с таблицей 16.

Таблица 16 – Распределение ресурсов проекта по задачам

№	Название задачи	Ресурсы	Единицы(затраты)
1	Начало реализация	Компьютер	1 шт.
2	Анализ требований и постановка задачи		
3	Анализ рынка соответствующих продуктов	Аналитик Интернет	100% 470р.
4	Анализ функциональных требований	Программист Интернет Аналитик	50% 470р. 50%
5	Анализ требований к пользовательскому интерфейсу	Программист	25%
6	Анализ требований к средствам разработки	Аналитик Программист	100% 25%
7	Постановка задачи	Программист	50%
8	Анализ предметной области и постановка задачи завершены		
9	Проектирование приложения		
10	Проектирование пользовательского интерфейса	Программист	100%
11	Проектирование базы данных	Программист	100%
12	Проектирование структуры программы	Программист	100%
13	Проектирование завершено		
14	Реализация приложения		
15	Реализация структур данных	Программист	100%
16	Реализация пользовательского интерфейса	Программист	100%

Продолжение таблицы 16

№	Название задачи	Ресурсы	Единицы(затраты)
17	Реализация программных средств	Программист Интернет	100% 470р.
18	Реализация завершена		
19	Отладка и тестирование		
20	Отладка	Программист Тестировщик	50% 50%
21	Структурное тестирование	Тестировщик	100%
22	Анализ качества кода	Тестировщик	100%
23	Отладка и тестирование завершены		
24	Составление программной документации	Бумага Аналитик Программист Тестировщик Флеш-карта Интернет	1 уп. 100% 100% 100% 1 шт. 470р.
25	Сдача в эксплуатацию	Аналитик Программист Тестировщик	100% 100% 100%
26	Проект завершён		
27	Конец проекта		

На рисунке 17 представлена таблица из Microsoft Project, которая содержит полную информацию о списке задач, их длительности и распределения ресурсов.

	Название задачи	Длительн	Начало	Окончание	Предше	Названия ресурсов
1	Начало реализации	1 день	Вс 02.02.20	Вс 02.02.20		Телефон поддерживающий VR[1 шт.];Компьютер[1 шт.]
2	Анализ требований и постановка задачи	9 дней	Пн 03.02.20	Чт 13.02.20		
3	Анализ рынка соответствующих продуктов	3 дней	Пн 03.02.20	Ср 05.02.20	1	Аналитик;Интернет[470,00р.]
4	Анализ функциональных требований	1 день	Чт 06.02.20	Чт 06.02.20	3	Аналитик[50%];Интернет;Програмист[50%]
5	Анализ требований к пользовательскому интерфейсу	2 дней	Пт 07.02.20	Пн 10.02.20	3	Програмист[25%]
6	Анализ требований к средствам разработки	1 день	Вт 11.02.20	Вт 11.02.20	4;5	Аналитик;Програмист[25%]
7	Постановка задачи	1 день	Ср 12.02.20	Ср 12.02.20	3	Програмист[50%]
8	Анализ предметной области и постановка задачи завершены	0 дней	Чт 13.02.20	Чт 13.02.20	7	
9	Проектирование приложения	23 дней	Ср 12.02.20	Пн 16.03.20	7	
10	Проектирование пользовательского интерфейса	10 дней	Чт 13.02.20	Ср 26.02.20	7	Програмист
11	Проектирование базы данных	8 дней	Чт 27.02.20	Пн 09.03.20	10	Програмист
12	Проектирование структуры программы	5 дней	Пн 09.03.20	Пн 16.03.20	11	Програмист
13	Проектирование завершено	0 дней	Пн 16.03.20	Пн 16.03.20	12	
14	Реализация приложения	28 дней	Пн 16.03.20	Чт 23.04.20	12	
15	Реализация структур данных	2 дней	Вт 17.03.20	Ср 18.03.20	11	Програмист
16	Реализация пользовательского интерфейса	5 дней	Чт 19.03.20	Ср 25.03.20	15	Програмист
17	Реализация программных средств	20 дней	Чт 26.03.20	Ср 22.04.20	16	Програмист;Интернет[470,00р.]
18	Реализация завершена	0 дней	Чт 23.04.20	Чт 23.04.20	17	
19	Отладка и тестирование	14 дней	Чт 23.04.20	Вт 12.05.20	14	
20	Отладка	4 дней	Чт 23.04.20	Вт 28.04.20	17	Програмист[50%];Тестировщик[50%]
21	Структурное тестирование	5 дней	Ср 29.04.20	Вт 05.05.20	17	Тестировщик
22	Анализ качества кода	4 дней	Ср 06.05.20	Пн 11.05.20	20	Тестировщик
23	Отладка и тестирование завершены	0 дней	Вт 12.05.20	Вт 12.05.20	22	
24	Завершение проекта	31 дней	Ср 13.05.20	Ср 24.06.20	19	
25	Составление программной документации	30 дней	Вт 12.05.20	Пн 22.06.20	22	Аналитик;Бумага[1 уп.];Интернет[470,00р.];Програмист;Т
26	Сдача в эксплуатацию	1 день	Вт 23.06.20	Вт 23.06.20	25	Аналитик;Програмист;Тестировщик
27	Проект завершён	0 дней	Вт 23.06.20	Вт 23.06.20	26	
28	Конец проекта	0 дней	Вт 23.06.20	Вт 23.06.20	27	

Рисунок 17 – Перечень задач и ресурсов проекта

После формирования списка задач с ресурсами проекта, Microsoft Project автоматически строит диаграмму Ганта, которая отражает календарный план выполнения проекта со всеми задачами и требующимися для них ресурсами. Диаграмма ганта представлена на рисунке 18.

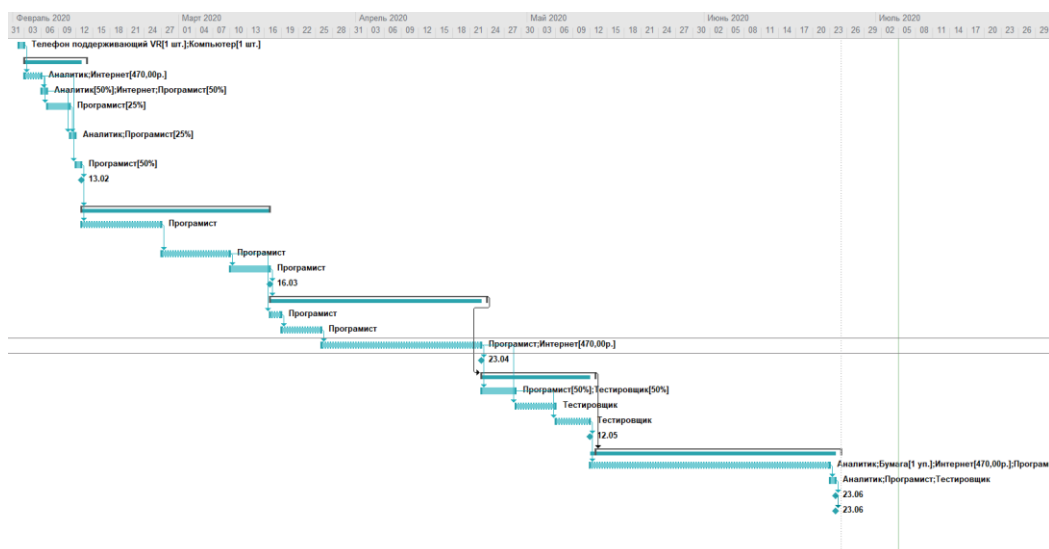


Рисунок 18 – Диаграмма Ганта

После построения диаграммы Ганта в Microsoft Project можно создать отчёты которые позволяют оценить затраты на разработку проекта. На рисунке 19 представлен отчёт по движению денежных средств.

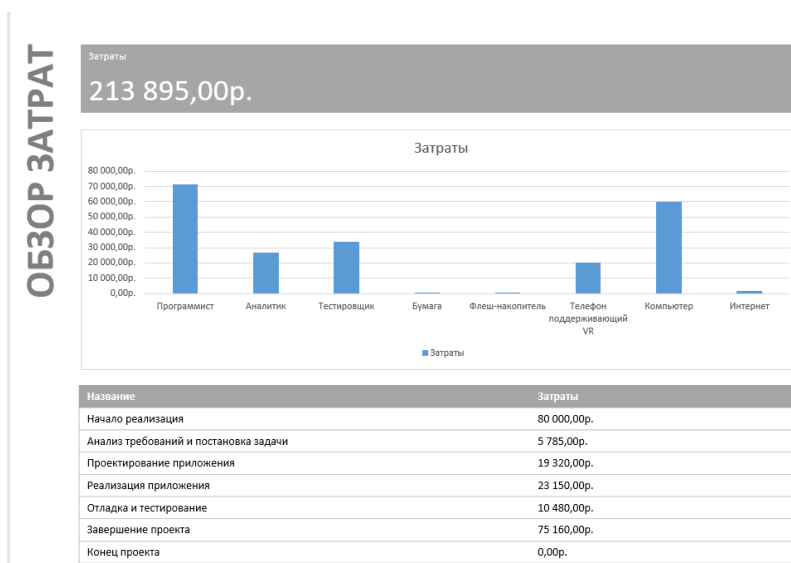


Рисунок 19 – Затраты денежных средств на реализацию проекта

Также Microsoft Project предоставляет возможность построить отчет о трудозатратах, позволяя оценить временные на весь проект. Данный отчет представлен на рисунке 20.

ОБЗОР ТРУДОЗАТРАТ

СТАТИСТИКА РЕСУРСОВ

Состояние трудозатрат для всех трудовых ресурсов.



СОСТОЯНИЕ РЕСУРСОВ

Оставшиеся трудозатраты для всех трудовых ресурсов

Название	Начало	Окончание	Оставшиеся трудозатраты
Программист	Чт 06.02.20	Вт 23.06.20	678 часов
Аналитик	Пн 03.02.20	Вт 23.06.20	284 часов
Тестировщик	Чт 23.04.20	Вт 23.06.20	336 часов

Рисунок 20 – Трудозатраты на проект

По результатам проведённых отчётов видно, что бюджет проекта составил 213895 рублей, а срок выполнения проекта 54 дня. Благодаря оценке сроков и бюджета проекта можно сделать вывод о том, что процесс разработки приложения химической лаборатории с использованием технологии виртуальной реальности является экономически выгодным.

Заключение

В результате выполнения выпускной квалификационной работы, с помощью игрового движка Unity3d, а также программ Visual Studio и Blender, было реализовано приложение химической лаборатории с использованием технологии виртуальной реальности.

Был проведен анализ аналогов на рынке интернет-магазина GooglePlay. Также было проведено проектирование пользовательского интерфейса, данных и структуры программы, и разработаны в соответствие с этим: пользовательский интерфейс, взаимодействия пользователя и приложения с использованием джойстика, а также написаны алгоритмы для взаимодействия с пользователем и сервером образовательной платформы Experiment. Проведено структурное тестирование, которое показало корректность работы программы. Также было выполнено планирование и расчёт бюджета на разработку приложения виртуальной химической лаборатории.

Список использованных источников

1. Като, Х. Манипуляция виртуальными объектами в настольной среде AR / Х. Като, М. Биллингхерст, И. Поупырев, К. Имамото, К. Тачибана // ISAR'00. – 2000. – С.111-119.
2. Козловский, Е.О. Мультимедийная виртуальная лаборатория по физике в системе дистанционного обучения / Е.О. Козловский, Г.М. Кравцов// Магистерская студия – 2016. – Выпуск №16. – С. 58-66.
3. Гультяев, А.К. Проектирование и дизайн пользовательского интерфейса / А.К. Гультяев, В.А. Машин: учебное пособие. – СПб.: Корона–Принт, 2007. – 239 с.
4. Документация для разработчиков по Cardboard SDK [Электронный ресурс]. – URL: <https://developers.google.com/cardboard/develop> (дата обращения 10.06.20).
5. Документация для разработчиков Unity [Электронный ресурс]. - <https://docs.unity3d.com/ru/530/ScriptReference/MonoBehaviour.html> (дата обращения 10.06.20).
6. Виды тестирования программного обеспечения // Тестирование программного обеспечения [Электронный ресурс]. – URL: <http://www.protesting.ru/testing/testtypes.html> (дата обращения: 12.06.2020).
7. Четфилд, К. Microsoft Project 2016 / Четфилд К., Джонсон Т. – М.: ЭКОМ Паблишерз, 2018. – 594 с.

Приложение А
Глоссарий
(справочное)

Виртуальная реальность (virtual reality, VR) – это созданный техническими средствами мир, передаваемый человеку через органы чувств.

Дополненная реальность (augmented reality, AR) – это среда, в реальном времени дополняющая физический мир, каким мы его видим, цифровыми данными с помощью каких-либо устройств – планшетов, смартфонов или других, и программной части.

Google Play (Play Market, Android Market) – магазин приложений, игр, книг, музыки и фильмов компании Google и других компаний, позволяющий владельцам устройств с операционной системой Android устанавливать и приобретать различные приложения.

JSON – текстовый формат обмена данных основанный на JavaScript.

Unity3d – это инструмент для разработки двух- и трехмерных приложений и игр, работающий под операционными системами Windows, Linux и OS X.

Microsoft Visual Studio 2017 (VS17) – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств.

Blender – профессиональное свободное и открытое программное обеспечение для создания трёхмерной компьютерной графики, включающее в себя средства моделирования, анимации, рендеринга, постобработки и монтажа видео со звуком, компоновки с помощью «узлов», а также для создания интерактивных игр.

Стереоскопическая визуализация – это вид визуализации, при котором синтезируются 2 изображения с 2х виртуальных камер, одно изображение – для левого глаза, другое – для правого.

Токен – строка символов, являющаяся уникальным идентификатором пользователя, которая выдаётся пользователю после успешной авторизации и является ключом для доступа к различным службам системы.

Приложение Б
Диаграммы
(рекомендуемое)

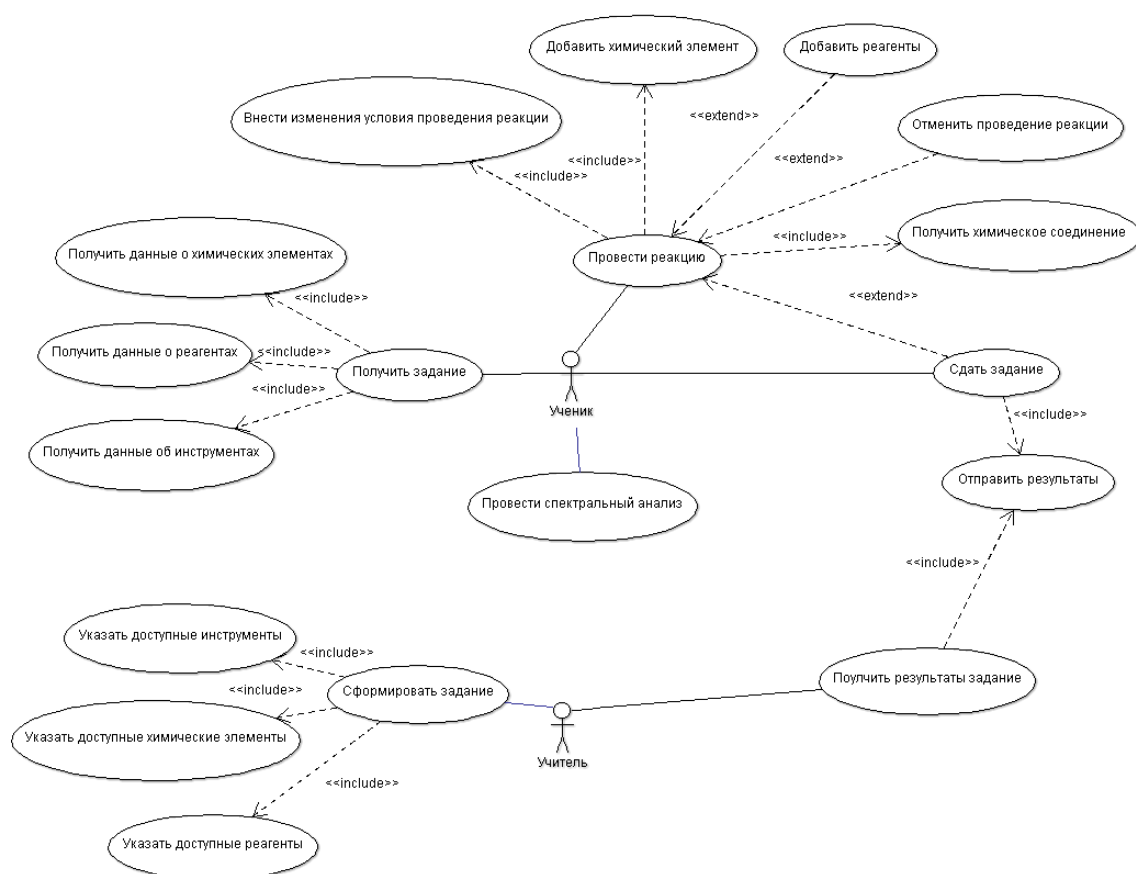


Рисунок Б.1 – Детализированная диаграмма вариантов использования

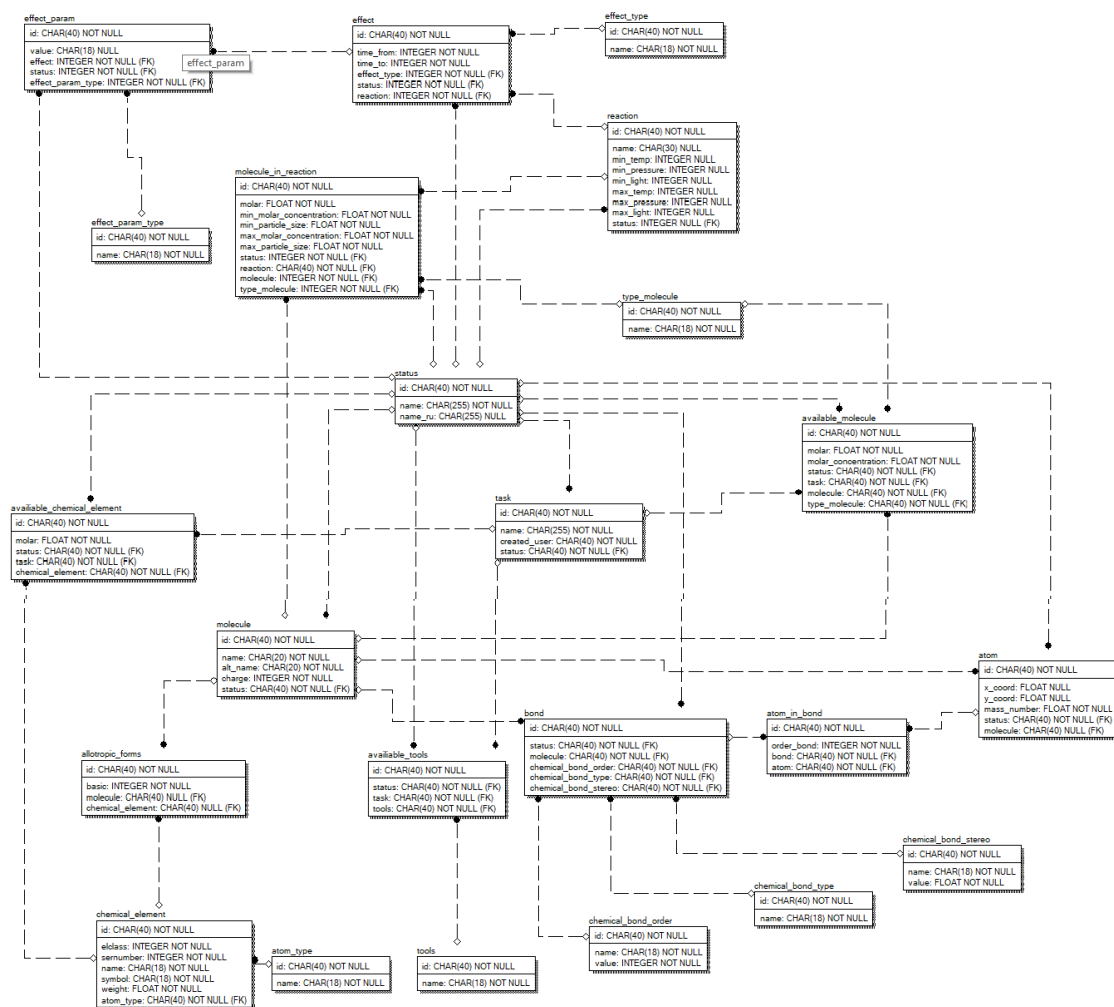


Рисунок Б.2 – Физическая модель данных

Приложение В
Листинг программы на язык C#
(рекомендуемое)

```

public class AcceptTask : MonoBehaviour {

    public GameObject task;
    private GameObject element;
    // Use this for initialization
    void Start () {

    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Element")
        {
            element=other.gameObject;
            task.SetActive(true);
        }
    }

    public void acceptTask()
    {
        Debug.Log(element.GetComponent<ElementInfo>().JSInfo.element.name);
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.tag == "Element")
        {
            task.SetActive(false);
        }
    }

    // Update is called once per frame
    void Update () {

    }
}

```

```

public class ButtonClick : MonoBehaviour, IPointerEnterHandler,
IPointerExitHandler
{

    private bool cursor;
    // Use this for initialization
    void Start()
    {
        cursor = false;
    }
}

```

```

    }

    // Update is called once per frame
    void Update()
    {
        if ((Input.GetKeyDown(KeyCode.Mouse2) ||
Input.GetKeyDown(KeyCode.JoystickButton3)) && cursor)
        {
            try
            {

gameObject.GetComponentInChildren<Button>().onClick.Invoke();
                if (!gameObject.activeInHierarchy)
                {
                    cursor = false;
                }
            }
            catch (Exception e)
            {
                try
                {

gameObject.GetComponentInChildren<CreateElements>().CreateElement();
                }
                catch (Exception ex)
                {

                    Debug.Log(ex.Message);
                }
            }
        }
    }

    public void OnPointerExit(PointerEventData eventData)
    {
        cursor = false;
    }

    public void OnPointerEnter(PointerEventData eventData)
    {
        cursor = true;
    }
}

```

```

public class ButtonGenerator : MonoBehaviour
{
    //Text on buttons
    private Text myText;

    string Canvas = "Material";

    public Material[] m_Material;
    private Renderer rend;
    private JsonReader info;

    // Use this for initialization
    void Start()
    {
        info =
GameObject.Find("Controller").GetComponent<JsonReader>();
        foreach (Pelement pelements in info.json.PElements)
        {
            //Debug.Log("Зашёл в Цикл");
            SetName(pelements.symbol, pelements.name);
            SetLabel(pelements.symbol);
            SetNumber(pelements.symbol,
pelements.sernumber.ToString());
            SetWeight(pelements.symbol,
pelements.weight.ToString());
            SetMaterial(pelements.symbol, pelements.elclass);
        }
    }

    // Set material of our buttons
    private void SetMaterial(string label, int elclass)
    {
        rend = GameObject.Find("PeriodicTable/" + Canvas + "/" +
label).GetComponent<Renderer>();
        rend.enabled = true;
        switch (elclass)
        {
            case 1:
                rend.sharedMaterial = m_Material[5];
                break;
            case 2:
                rend.sharedMaterial = m_Material[0];
                break;
            case 3:
                rend.sharedMaterial = m_Material[1];
                break;
            case 6:
                rend.sharedMaterial = m_Material[2];
                break;
        }
    }
}

```

```

        case 5:
            rend.sharedMaterial = m_Material[3];
            break;
        case 4:
            rend.sharedMaterial = m_Material[4];
            break;
        case 7:
            rend.sharedMaterial = m_Material[6];
            break;
        case 8:
            rend.sharedMaterial = m_Material[7];
            break;
        default:
            rend.sharedMaterial = m_Material[3];
            break;
    }

}

// Set name of element on button
private void SetName(string label, string name)
{
    myText = GameObject.Find("PeriodicTable/" + Canvas + "/"
+ label + "/name").GetComponent<Text>();
    myText.text = name;
}

// Set label of element on button
private void SetLabel(string label)
{
    myText = GameObject.Find("PeriodicTable/" + Canvas + "/"
+ label + "/" + label).GetComponent<Text>();
    myText.text = label;
}

// Set number of element on button
private void SetNumber(string label, string number)
{
    myText = GameObject.Find("PeriodicTable/" + Canvas + "/"
+ label + "/number").GetComponent<Text>();
    myText.text = number;
}

//Set weight of elemet on button
private void SetWeight(string label, string weight)
{
    myText = GameObject.Find("PeriodicTable/" + Canvas + "/"
+ label + "/weight").GetComponent<Text>();
    myText.text = weight;
}
}

```

```

[AddComponentMenu("Camera/Simple Smooth Mouse Look ")]
public class CameraRotation : MonoBehaviour
{
    Vector2 _mouseAbsolute;
    Vector2 _smoothMouse;

    public Vector2 clampInDegrees = new Vector2(360, 180);
    public bool lockCursor;
    public Vector2 sensitivity = new Vector2(2, 2);
    public Vector2 smoothing = new Vector2(3, 3);
    public Vector2 targetDirection;
    public Vector2 targetCharacterDirection;

    // Assign this if there's a parent object controlling
    // motion, such as a Character Controller.
    // Yaw rotation will affect this object instead of the
    // camera if set.
    public GameObject characterBody;

    void Start()
    {
        // Set target direction to the camera's initial
        // orientation.
        targetDirection = transform.localRotation.eulerAngles;

        // Set target direction for the character body to its
        // initial state.
        if (characterBody)
            targetCharacterDirection =
                characterBody.transform.localRotation.eulerAngles;
    }

    void Update()
    {
        // Ensure the cursor is always locked when set
        if (lockCursor)
        {
            Cursor.lockState = CursorLockMode.Locked;
        }

        // Allow the script to clamp based on a desired target
        // value.
        var targetOrientation =
            Quaternion.Euler(targetDirection);
        var targetCharacterOrientation =
            Quaternion.Euler(targetCharacterDirection);

        // Get raw mouse input for a cleaner reading on more
        // sensitive mice.
        var mouseDelta = new Vector2(Input.GetAxisRaw("Mouse
X"), Input.GetAxisRaw("Mouse Y"));
    }
}

```

```

        // Scale input against the sensitivity setting and
        multiply that against the smoothing value.
        mouseDelta = Vector2.Scale(mouseDelta, new
        Vector2(sensitivity.x * smoothing.x, sensitivity.y *
        smoothing.y));

        // Interpolate mouse movement over time to apply
        smoothing delta.
        _smoothMouse.x = Mathf.Lerp(_smoothMouse.x,
        mouseDelta.x, 1f / smoothing.x);
        _smoothMouse.y = Mathf.Lerp(_smoothMouse.y,
        mouseDelta.y, 1f / smoothing.y);

        // Find the absolute mouse movement value from point
        zero.
        _mouseAbsolute += _smoothMouse;

        // Clamp and apply the local x value first, so as not to
        be affected by world transforms.
        if (clampInDegrees.x < 360)
            _mouseAbsolute.x = Mathf.Clamp(_mouseAbsolute.x, -
            clampInDegrees.x * 0.5f, clampInDegrees.x * 0.5f);

        // Then clamp and apply the global y value.
        if (clampInDegrees.y < 360)
            _mouseAbsolute.y = Mathf.Clamp(_mouseAbsolute.y, -
            clampInDegrees.y * 0.5f, clampInDegrees.y * 0.5f);

        transform.localRotation = Quaternion.AngleAxis(-
        _mouseAbsolute.y, targetOrientation * Vector3.right) *
        targetOrientation;

        // If there's a character body that acts as a parent to
        the camera
        if (characterBody)
        {
            var yRotation =
            Quaternion.AngleAxis(_mouseAbsolute.x, Vector3.up);
            characterBody.transform.localRotation = yRotation *
            targetCharacterOrientation;
        }
        else
        {
            var yRotation =
            Quaternion.AngleAxis(_mouseAbsolute.x,
            transform.InverseTransformDirection(Vector3.up));
            transform.localRotation *= yRotation;
        }
    }
}

```

```

[RequireComponent(typeof(CharacterController))]
public class CharacterMotor : MonoBehaviour
{
    #region Variables

    private float _groundCheckDistanceMin = 1.0f;
    private CharacterController _controller;
    private Vector3 _moveDirection;
    private Vector3 _desiredVelocity;
    private bool _invertX;
    private bool _invertY;

    #endregion

    #region Fields

    public float MovementSpeed = 10.0f;
    public float Acceleration = 10.0f;
    public bool SwapAxes;

    #endregion

    #region Properties

    public bool IsGrounded
    {
        get
        {
            var groundCheckDistance = _groundCheckDistanceMin;
            RaycastHit hitInfo;
            if (Physics.SphereCast(transform.position,
            _controller.radius, Vector3.down, out hitInfo,
            ((_controller.height / 2f) - _controller.radius) *
            transform.localScale.x + groundCheckDistance,
            Physics.AllLayers, QueryTriggerInteraction.Ignore))
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }

    public bool CanMove { get; set; }

    #endregion

    private StreamWriter writer;

```

```

// Use this for initialization
private void Start()
{
    string path =
Path.Combine(Application.persistentDataPath,
"authorizationLog.txt");
    writer = new StreamWriter(path, true,
System.Text.Encoding.Default);
    CanMove = true;
    _controller = GetComponent<CharacterController>();
}

private void FixedUpdate()
{
    float x = 0, y = 0;
    if (CanMove)
    {
        x = (!SwapAxes) ? Input.GetAxis("Vertical") :
Input.GetAxis("Horizontal");
        y = (!SwapAxes) ? Input.GetAxis("Horizontal") :
Input.GetAxis("Vertical");
    }

    var input = new Vector2(x, y);
    Movement(input);
    _controller.Move((_moveDirection + _desiredVelocity) *
Time.deltaTime);
}

private void Movement(Vector2 input)
{
    if (IsGrounded)
    {
        var targetVelocity = new Vector3(input.y *
MovementSpeed, 0, input.x * MovementSpeed);
        targetVelocity =
Camera.main.transform.TransformDirection(targetVelocity);
        targetVelocity.y = 0.0f;
        _moveDirection = Physics.gravity;
        _desiredVelocity = Vector3.Lerp(_desiredVelocity,
targetVelocity, Acceleration);
    }
    else
    {
        _moveDirection += Physics.gravity * Time.deltaTime *
2.0f;
    }
}
}

```

```

public class CreateElements : MonoBehaviour
{
    private Vector3 pos_elements;
    private GameObject clone_elements;
    private GameObject Display;
    public GameObject Light_Circle;
    private GameObject Light_Spectrometr;
    public GameObject elements;

    public Vector3 translate;

    private void Start()
    {
        pos_elements = GameObject.Find("laboratory
table/pos_elements").transform.position;

        translate = new Vector3(0.05f, 0, 0);
    }

    public void CreateElement()
    {
        Light_Spectrometr =
GameObject.Find("Light_spectrometr");
        if (Light_Spectrometr == true)
        {
            Light_Spectrometr.SetActive(false);
        }
        Display = GameObject.Find("Display");
        if (Display == true)
        {
            Display.SetActive(false);
        }
        if (clone_elements == true)
        {
            Destroy(clone_elements);
        }
        Display = GameObject.Find("Display");
        clone_elements = Instantiate(elements, pos_elements,
Quaternion.identity);
        Light_Circle.SetActive(true);
        //ClickButton();
        StartCoroutine(StartLight());
    }

    public IEnumerator StartLight()
    {
        yield return new WaitForSeconds(2f);
        Transform t = gameObject.GetComponent<Transform>();

        Light_Circle.SetActive(false);
    }
}

```

```

    }

    public void ClickButton()
    {
        Transform t = gameObject.GetComponent<Transform>();
        t.position += translate;
    }
}

```

```

[RequireComponent(typeof(Rigidbody))]
public class DragableObject : EventTrigger
{
    #region Fields
    /// <summary>
    /// Force you move objects with
    /// </summary>
    public float Force = 1200;

    /// <summary>
    /// Amount of 'jiggling'
    /// </summary>
    public float Damping = 0;

    /// <summary>
    /// Should object be moved only along the plane
    /// </summary>
    public bool StickToPlane = false;

    public bool drag = false;

    #endregion

    #region Variables

    private Transform _jointTrans;
    private float _dragDepth;
    private bool _dragging;
    private bool cursor;

    private Plane _dragPlane;

    /// <summary>
    /// Is object being held by user
    /// </summary>
    private bool _held;

    #endregion

    #region Methods

```

```

        public override void OnPointerDown(PointerEventData
eventData)
        {
            DragStart();
        }
        public override void OnPointerUp(PointerEventData eventData)
        {
            DragEnd();
        }

        public override void OnPointerEnter(PointerEventData
eventData)
        {
            cursor = true;
        }

        public override void OnPointerExit(PointerEventData
eventData)
        {
            cursor = false;
        }

        public void DragStart()
        {
            HandleInputBegin();
            _dragging = true;
            drag = true;
        }

        public void DragEnd()
        {
            HandleInputEnd();
            _dragging = false;
            drag = false;
        }

        public void Drag()
        {
            HandleInput();
            RadialForce(0.0001f, 1.0f);
        }

        private void Update()
        {
            if ((Input.GetKeyUp(KeyCode.Mouse2) ||
Input.GetKeyUp(KeyCode.JoystickButton3) ||
Input.GetKeyUp(KeyCode.JoystickButton5)))
            {

```

```

        PointerEventData eventData = new
PointerEventData(GameObject.Find("EventSystem").GetComponent<EventSystem>());
        OnPointerUp(eventData);
    }
    if
((Input.GetKeyDown(KeyCode.Mouse2) || Input.GetKeyDown(KeyCode.JoystickButton3) || Input.GetKeyDown(KeyCode.JoystickButton5)) &&
cursor)
    {
        PointerEventData eventData = new
PointerEventData(GameObject.Find("EventSystem").GetComponent<EventSystem>());
        OnPointerDown(eventData);
    }

    if (!_dragging) return;

    Drag();
}

/// <summary>
/// A little hack to make physics work when object picked up
/// </summary>
/// <param name="power">How strong to push</param>
/// <param name="radius">In what radius to push</param>
private void RadialForce(float power, float radius)
{
    Vector3 explosionPos = transform.position;
    Collider[] colliders =
Physics.OverlapSphere(explosionPos, radius);
    foreach (Collider hit in colliders)
    {
        Rigidbody rb = hit.GetComponent<Rigidbody>();

        if (rb != null)
            rb.AddExplosionForce(power, explosionPos,
radius);
    }
}

/// <summary>
/// Used to handle the point, where object picked
/// </summary>
/// <param name="rb">What object picked</param>
/// <param name="attachmentPosition">Where object
picked</param>
/// <returns></returns>
private Transform AttachJoint(Rigidbody rb, Vector3
attachmentPosition)
{

```

```

var go = new GameObject("Attachment Point")
{
    hideFlags = HideFlags.HideInHierarchy
};
go.transform.position = attachmentPosition;

var newRb = go.AddComponent<Rigidbody>();
newRb.isKinematic = true;

var joint = go.AddComponent<ConfigurableJoint>();
joint.connectedBody = rb;
joint.configuredInWorldSpace = true;
joint.xDrive = NewJointDrive(Force, Damping);
joint.yDrive = NewJointDrive(Force, Damping);
joint.zDrive = NewJointDrive(Force, Damping);
joint.slerpDrive = NewJointDrive(Force, Damping);
joint.rotationDriveMode = RotationDriveMode.Slerp;

return go.transform;
}

/// <summary>
/// Used to handle an object movement
/// </summary>
/// <param name="force">What force to apply</param>
/// <param name="damping">How much 'jiggling' to add</param>
/// <returns></returns>
private JointDrive NewJointDrive(float force, float damping)
{
    var drive = new JointDrive
    {
        positionSpring = force,
        positionDamper = damping,
        maximumForce = Mathf.Infinity
    };
    return drive;
}

/// <summary>
/// Called when object clicked
/// </summary>
private void HandleInputBegin()
{
    _held = true;
    Ray ray = Camera.main.ViewportPointToRay(new
Vector3(0.5f, 0.5f, 0f));
    RaycastHit hit;
    if (!Physics.Raycast(ray, out hit)) return;

    _jointTrans = AttachJoint(GetComponent<Rigidbody>(),
hit.point);

```

```

        if (StickToPlane)
        {
            var normal = Camera.main.transform.forward;
            normal.y = 0;
            _dragPlane.SetNormalAndPosition(normal, hit.point);
        }
        else
        {
            _dragDepth = (Camera.main.transform.position -
hit.point).magnitude;
        }
    }
    /// <summary>
    /// Used to handle object drag over the place
    /// </summary>
    /// <param name="screenPosition"></param>
    private void HandleInput()
    {
        if (_jointTrans == null) return;

        Ray ray = Camera.main.ViewportPointToRay(new
Vector3(0.5f, 0.5f,
0f));
        if (StickToPlane)
            _dragPlane.Raycast(ray, out _dragDepth);

        var pos = ray.GetPoint(_dragDepth);
        _jointTrans.position = pos;
    }
    /// <summary>
    /// Used when object released
    /// <para>
    /// Destroys any temporary objects
    /// </para>
    /// </summary>
    private void HandleInputEnd()
    {
        _held = false;
        if (_jointTrans != null)
        {
            Destroy(_jointTrans.gameObject);
        }
    }
}
#endregion
}

```

```

public class ElementInfo : MonoBehaviour, IPointerEnterHandler,
IPointerExitHandler
{
    private JsonSerializer info;

```

```

private string name;
public Pelement temp;
public JSInfo JSInfo;

private bool cursor;

public void Start()
{
    cursor = false;
    JSInfo = new JSInfo();
    name = gameObject.name.Replace("(Clone)", "");
    info =
GameObject.Find("Controller").GetComponent<JsonReader>();
    temp = Search(name);
    JSInfo.element = temp;
    JSInfo.idevent = 1;
    JSInfo.localDate = DateTime.UtcNow.ToString("yyyy-MM-
ddTHH\\:mm\\:ss"); ;

    Debug.Log(JSInfo.element.name);
    string js = CreateJson(JSInfo).ReadToEnd();
    try
    {
        string token =
CheckAuthorization.ExperimentAccount.Token;
        Debug.Log(SendQuery.POST(host.userDomain +
"applicationController/event/1/1",
            "{ action:" +
GetComponent<ElementInfo>().JSInfo.element.sernumber + "}",
            token));
    } catch(Exception e)
    {
        Debug.Log(e.Message.ToString());
    }
    Debug.Log(js);
}

public StreamReader CreateJson(JSInfo JSInfo)
{
    MemoryStream stream1 = new MemoryStream();
    DataContractJsonSerializer ser = new
DataContractJsonSerializer(typeof(JSInfo));
    ser.WriteObject(stream1, JSInfo);
    stream1.Position = 0;
    return new StreamReader(stream1);
}

public Pelement Search(string name)
{
    foreach (Pelement elements in info.json.PElements)
    {

```

```

        if (elements.name == name)
        {
            return elements;
        }
    }
    return null;
}

/// <summary>
/// Курсор не на объект
/// </summary>
public void OnPointerExit(PointerEventData eventData)
{
    cursor = false;
}

/// <summary>
/// Наведение курсора на объект
/// </summary>
public void OnPointerEnter(PointerEventData eventData)
{
    cursor = true;
}

/// <summary>
/// Взятие проб
/// </summary>
public void Update()
{
    if ((Input.GetMouseButtonDown(1) ||
Input.GetKeyUp(KeyCode.JoystickButton0)) && cursor)
    {
        GameObject.Find("Main
Camera").GetComponent<Sampling>().elem =
GetComponent<ElementInfo>().JSInfo;

        Debug.Log("Вы взяли " +
GetComponent<ElementInfo>().JSInfo.element.name);
        try
        {
            string token =
CheckAuthorization.ExperimentAccount.Token;

Debug.Log(SendQuery.POST(host.userDomain+"applicationController/
event/1/2", "{ action:" +
GetComponent<ElementInfo>().JSInfo.element.sernumber + "}",
token));
        }
        catch (Exception)
        {

```

```

        }
    }
}

[DataContract]
public class JSInfo
{
    [DataMember]
    public Pelement element;
    [DataMember]
    public int idevent;
    [DataMember]
    public string localDate;
}
public class GenerateReagents : MonoBehaviour
{
    private Reagents Reagents;

    // Use this for initialization
    void Start()
    {
        try
        {
            Reagents = new Reagents();
            string json = "{ \"reagents\": " +
SendQuery.GET(host.chemDomain +
"availableMolecule/getAvailableMoleculeList/09eb801e-da1f-11e9-
a006-cfc414a1059d", CheckAuthorization.ExperimentAccount.Token)
+ " }";

            Debug.Log(json);
            Reagents = JsonUtility.FromJson<Reagents>(json);
            for (int i = 0; i < transform.childCount && i <
Reagents.reagents.Count; i++)
            {
                GameObject game =
transform.GetChild(i).gameObject;
                game.SetActive(true);

                game.transform.Find("Canvas").GetComponentInChildren<Text>().tex
t = Reagents.reagents[i].name;
                game.GetComponent<Rigidbody>().freezeRotation =
true;

            }
        } catch (Exception e)
        {
            Debug.Log("Ошибка: " + e.Message);
        }
    }
}

```

```

    }

    // Update is called once per frame
    void Update()
    {

    }
}

[System.Serializable]
public class Reagents
{
    public List<Reagent> reagents = new List<Reagent>();
}

[System.Serializable]
public class Reagent
{
    public string id;
    public string name;
}

public class JSONReader : MonoBehaviour
{
    TextAsset asset;
    public ElementsArray json;
    public void Read()
    {
        asset = Resources.Load("PeriodicTable") as TextAsset;
        Debug.Log(asset.text);
        try
        {
            json =
JsonUtility.FromJson<ElementsArray>(asset.text);
        }
        catch (Exception e)
        {
            Debug.Log(e.Message);
            Debug.Log(e.StackTrace);
        }

    }

    private void Awake()
    {
        //try
        //{
        //    Read();
        //}
        //catch (Exception e)
        //{

```

```

        //      string path =
Path.Combine(Application.persistentDataPath, "errors.txt");
        //      StreamWriter debugFile = new StreamWriter(path,
true, System.Text.Encoding.Default);
        //      print(e.Message);
        //      Debug.Log(e.Message);
        //}
        try
        {

            string Elements = SendQuery.GET(host.chemDomain +
"chemicalelement/getChemicalElemList",
CheckAuthorization.ExperimentAccount.Token);
            Debug.Log(Elements);
            json =
JsonUtility.FromJson<ElementsArray>(Elements);

//Debug.Log(json.PElements[0].allotropList[0].molecule.id);
        }
        catch (Exception e)
        {
            //Debug.LogError(e.Message);
            Debug.Log(e.Message);
            Debug.Log(e.Data);
            Debug.Log(e.Source);
            Debug.Log(e.StackTrace);
            //Read();
        }

        //Debug.Log(asset.text);
    }
}

// Array of Elements from Json
[System.Serializable]
public class ElementsArray
{
    public Pelement[] PElements { get; set; }
}

//Data from JSon, Elements in an array
[System.Serializable]
public class Pelement
{
    public AllotropList[] allotropList { get; set; }
    public int elclass { get; set; }
    public int id { get; set; }
    public string name { get; set; }
    public int sernumber { get; set; }
    public string symbol { get; set; }
}

```

```

        public float weight { get; set; }
    }

[System.Serializable]
public class Molecules
{
    public string altName;
    public string id;
    public string name;
}

[System.Serializable]
public class Allotrop
{
    public Molecules molecule;
    public int type;
}

public class LoadScene : MonoBehaviour {

    public Image image;
    public Text proggresText;

    // Use this for initialization
    void Start () {
        StartCoroutine(AsyncLoad());
    }

    IEnumerator AsyncLoad()
    {
        AsyncOperation operation =
SceneManager.LoadSceneAsync(2);
        while (!operation.isDone)
        {
            float progress= operation.progress/0.9f;
            image.fillAmount = progress;
            proggresText.text = string.Format("{0:0}%",
progress*100);
            yield return null;
        }
    }
}

public class MenuButtons : MonoBehaviour
{
    private void Start()
    {
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;
    }
}

```

```

public void Exit()
{
    Application.Quit();
}

public void StartGame()
{
    string sceneName = "test";
    SceneManager.LoadScene(sceneName);
}

private void Update()
{
    if (Input.GetKey(KeyCode.JoystickButton0))
    {
        SceneManager.LoadScene("MainScene");
    }
}
}

public class OpenMenuReactions : MonoBehaviour,
IPointerClickHandler, IPointerEnterHandler, IPointerExitHandler
{

    public GameObject CountElements;
    public GameObject AcceptReaction;
    public GameObject CanvasFlask;
    public Transform transformFlask;

    private bool cursor;
    // Use this for initialization
    void Start()
    {
        cursor = false;
    }

    // Update is called once per frame
    void Update()
    {
        if ((Input.GetKeyUp(KeyCode.JoystickButton5) ||
Input.GetKeyUp(KeyCode.JoystickButton3)) && cursor)
        {
            try
            {
                JSInfo js = GameObject.Find("Main
Camera").GetComponent<Sampling>().elem;
                if (js != null)
                {
                    CountElements.SetActive(true);
                }
            }
            else

```

```

        {
            if
(CanvasFlask.GetComponent<StartReaction>().reactions != null)
            {
                AcceptReaction.SetActive(true);
            }
        }
    }
    catch (Exception e) { Debug.Log(e.Message); }
}

public void OnPointerClick(PointerEventData eventData)
{
    try
    {
        JSInfo js = GameObject.Find("Main
Camera").GetComponent<Sampling>().elem;
        if (js != null)
        {
            CountElements.SetActive(true);
            CanvasFlask.GetComponent<StartReaction>().game =
CountElements;
        }
        else
        {
            if
(CanvasFlask.GetComponent<StartReaction>().reactions != null)
            {
                AcceptReaction.SetActive(true);
            }
        }
    }
    catch (Exception e)
    {
        Debug.Log(e.Message);
    }
}

public void OnPointerExit(PointerEventData eventData)
{
    cursor = false;
}

public void OnPointerEnter(PointerEventData eventData)
{
    cursor = true;
}
}

public class OptonsSettings : MonoBehaviour {

    public GameObject PausePanel;
    public GameObject OptionsPanel;

```

```

public void Back()
{
    OptionsPanel.SetActive(false);
    PausePanel.SetActive(true);
    Cursor.visible = true;
}

}

public class PauseSettings : MonoBehaviour {

    public GameObject PausePanel;
    public GameObject OptionsPanel;

    private void Start()
    {
        PausePanel.SetActive(false);
        OptionsPanel.SetActive(false);
        Cursor.visible = true;
    }

    public void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (PausePanel.activeInHierarchy)
            {
                PausePanel.SetActive(false);
                Time.timeScale = 1;
            } else
            {
                if (!PausePanel.activeInHierarchy)
                {
                    PausePanel.SetActive(true);
                    Time.timeScale = 0;
                }
                if (OptionsPanel.activeInHierarchy)
                {
                    PausePanel.SetActive(true);
                    OptionsPanel.SetActive(false);
                    Cursor.lockState = CursorLockMode.Confined;
                }
            }
            Cursor.visible = true;
        }
    }

    public void Continue()
    {

```

```

        PausePanel.SetActive(!PausePanel.activeInHierarchy);
        Time.timeScale = 1;
    }

    public void Options()
    {
        OptionsPanel.SetActive(!OptionsPanel.activeInHierarchy);
        PausePanel.SetActive(!PausePanel.activeInHierarchy);
        Cursor.visible = true;
    }

    public void ExitApp()
    {
        PausePanel.SetActive(!PausePanel.activeInHierarchy);
        Time.timeScale = 1;
        SceneManager.LoadScene("MainMenu");
    }
}

public class PutElementInSpectrometr : MonoBehaviour
{
    public GameObject Display;
    public GameObject Light_spectrometr;
    public Material[] ArraySpectr;
    public Renderer rend_spectr;
    private Text myText;

    private JsonReader info;

    private void Start()
    {
        info =
GameObject.Find("Controller").GetComponent<JsonReader>();
    }

    private void Start_work_spectrometr(string nameOfObject,
string ShortNameOfObject, string WeightOfObject)
    {
        foreach (Pelement pelements in info.json.PElements)
        {
            SetNameOfElement(nameOfObject, pelements);
            SetShortNameOfElement(ShortNameOfObject, pelements);
            SetSpectrOfElement(nameOfObject, pelements);
            SetWeightOfElement(WeightOfObject, pelements);
        }
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Element")
        {
            GetComponent<AudioSource>().Play();
        }
    }
}

```

```

        string nameOfObject = other.gameObject.name;
        string ShortNameOfObject = other.gameObject.name;
        string WeightOfObject = other.gameObject.name;

        nameOfObject = nameOfObject.Replace("(Clone)", "");
        ShortNameOfObject = nameOfObject.Replace("(Clone)",
");";
        WeightOfObject = nameOfObject.Replace("(Clone)",
");";

        Display.SetActive(true);
        Light_spectrometr.SetActive(true);

        Start_work_spectrometr(nameOfObject,
ShortNameOfObject, WeightOfObject);
    }
}
//this method hides Display with information about chemical
elements in spectrometr
private void OnTriggerExit(Collider other)
{
    if (other.tag == "Element")
    {
        Display.SetActive(false);
        Light_spectrometr.SetActive(false);
    }
}
//void for output name chemical element on Display
private void SetNameOfElement(string nameOfObject, Pelement
pelements)
{
    myText =
GameObject.Find("Display/name").GetComponent<Text>();
    if (pelements.name == nameOfObject)
    {
        myText.text = pelements.name;
    }
}
//void for output spectr chemical element on Display
private void SetSpectrOfElement(string nameOfObject,
Pelement pelements)
{
    myText =
GameObject.Find("Display/name").GetComponent<Text>();
    if (pelements.name == nameOfObject)
    {
        rend_spectr.enabled = true;
        rend_spectr.sharedMaterial =
ArraySpectr[pelements.sernumber];
    }
}
}

```

```

        //void for output short name chemical element on Display
        private void SetShortNameOfElement(string ShortNameOfObject,
        Pelement pelements)
        {
            myText =
GameObject.Find("Display/symbol").GetComponent<Text>();
            if (pelements.name == ShortNameOfObject)
            {
                myText.text = pelements.symbol;
            }

        }
        //void for output weigt chemical element on Display
        private void SetWeightOfElement(string WeightOfObject,
        Pelement pelements)
        {
            myText =
GameObject.Find("Display/weight").GetComponent<Text>();
            if (pelements.name == WeightOfObject)
            {
                myText.text = pelements.weight.ToString();
            }
        }
    }
    public class Sampling : MonoBehaviour
    {
        public JSInfo elem;

        // Use this for initialization
        void Start()
        {
            elem = null;
            //objecc = new GameObject();
        }
    }

    public class SendQuery
    {
        /// <summary>
        /// GET запрос
        /// </summary>
        public static string GET(string Url)
        {
            WebRequest request = WebRequest.Create(Url);
            request.Method = "GET";
            try
            {

                WebResponse response = request.GetResponse();
                Stream stream = response.GetResponseStream();
            }
        }
    }

```

```

        StreamReader streamReader = new
StreamReader(stream);
        string Output = streamReader.ReadToEnd();
        streamReader.Close();
        stream.Close();
        response.Close();
        return Output;
    }
    catch (Exception e)
    {
        return "Ошибка при запросе: " +
e.Message.ToString();
    }
}
/// <summary>
/// GET запрос с токеном в заголовке
/// </summary>
public static string GET(string Url,string token)
{
    HttpWebRequest request =
(HttpWebRequest)WebRequest.Create(Url);
    request.Method = "GET";
    request.Headers.Add("token", token);
    try
    {
        StreamReader streamReader = new
StreamReader(request.GetResponse().GetResponseStream(),Encoding.
UTF8);
        string Output = streamReader.ReadToEnd();
        streamReader.Close();
        return Output;
    }
    catch (WebException e)
    {
        return "Ошибка при запросе: " + e.Message.ToString()
+ "\nThe end Message"+ e.StackTrace
        + " \nThe end StackTrace\n "+e.Data
        + " \nThe end Data\n ";
    }
}

/// <summary>
/// GET запрос с токеном в заголовке и UserId
/// </summary>
public static string GET(string Url, string token,int uid)
{
    //Debug.Log("Отправка get запроса");
    HttpWebRequest request =
(HttpWebRequest)WebRequest.Create(Url);
    request.Method = "GET";
    request.Headers.Add("token", token);

```

```

        request.Headers.Add("user", uid.ToString());
        try
        {
            WebResponse response = request.GetResponse();
            Stream stream = response.GetResponseStream();
            StreamReader streamReader = new
StreamReader(stream);
            string Output = streamReader.ReadToEnd();
            streamReader.Close();
            stream.Close();
            response.Close();
            return Output;
        }
        catch (Exception e)
        {
            return "Ошибка при запросе: " +
e.Message.ToString();
        }
    }

    /// <summary>
    /// POST запрос
    /// </summary>
    public static string POST(string Url, string Data, string
token)
    {
        WebRequest request = WebRequest.Create(Url);
        request.Method = "POST";
        request.ContentType = "application/json;charset=utf-8";
        request.Headers.Add("token", token);

        byte[] sendData = Encoding.UTF8.GetBytes(Data);
        request.ContentLength = sendData.Length;
        Stream sendStream = request.GetRequestStream();
        sendStream.Write(sendData, 0, sendData.Length);
        sendStream.Close();

        try
        {
            StreamReader streamReader = new
StreamReader(request.GetResponse().GetResponseStream(),
Encoding.UTF8);
            string Output = streamReader.ReadToEnd();
            streamReader.Close();
            return Output;
        }
        catch (Exception e)
        {
            return "Ошибка при запросе: " +
e.Message.ToString();
        }
    }

```

```

    }
}

/// <summary>
/// Получение параметра при запуске и получение Task по
данному параметру
/// </summary>
public void TestQuery()
{
    if (Application.platform != RuntimePlatform.Android)
return;
    string path =
Path.Combine(Application.persistentDataPath, "testQuery.txt");
    StreamWriter writer = new StreamWriter(path, true,
System.Text.Encoding.Default);
    int taskId = 0;

    AndroidJavaClass UnityPlayer = new
AndroidJavaClass("com.unity3d.player.UnityPlayer");
    AndroidJavaObject currentActivity =
UnityPlayer.GetStatic<AndroidJavaObject>("currentActivity");

    AndroidJavaObject intent =
currentActivity.Call<AndroidJavaObject>("getIntent");
    bool hasExtra = intent.Call<bool>("hasExtra", "taskId");
    writer.WriteLine("hasExtra = " + hasExtra.ToString());
    if (hasExtra)
    {
        AndroidJavaObject extras =
intent.Call<AndroidJavaObject>("getExtras");
        taskId = extras.Call<int>("getInt", "taskId");
        writer.WriteLine("taskId = " + taskId.ToString());

        WebRequest request =
WebRequest.Create("http://192.168.0.84/services/app/task/" +
taskId.ToString());
        request.Method = "Get";

        try
        {
            var response = request.GetResponse();
            var answerStream = response.GetResponseStream();
            StreamReader sr = new
StreamReader(answerStream);
            writer.Write(sr.ReadToEnd());
            sr.Close();
            answerStream.Close();
            response.Close();
        }
        catch (Exception e)

```

```

        {
            writer.WriteLine("Ошибка при запросе: " +
e.Message.ToString());
        }
        writer.Close();
    }

}

}

```

```

public class SpiritLamp: MonoBehaviour {

    private bool SpiritLamp;
    public Transform transformFlask;
    private Vector3 position = new Vector3();
    // private Vector3 position2 = new Vector3();

    // Use this for initialization
    void Start () {
        GameObject Flask = GameObject.Find("ReactionFlask");
        position = Flask.GetComponent<Transform>().position;
        position.y = 1.256f;
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "ReactionFlask" && ! SpiritLamp)
        {
            if (!gameObject.GetComponent<DragableObject>().drag)
            {
                GameObject Flask =
GameObject.Find("ReactionFlask");

GameObject.Find("MenuReaction").transform.Find("Condition").Find
("Heat").gameObject.SetActive(true);
                SpiritLamp = true;
                gameObject.GetComponent<Transform>().position =
position;
                gameObject.transform.rotation =
Flask.transform.rotation;

                transformFlask.position = position;
                Flask.GetComponent<Rigidbody>().constraints =
RigidbodyConstraints.FreezeAll;

```

```

        gameObject.GetComponent<Rigidbody>().constraints
= RigidbodyConstraints.FreezeRotation |
RigidbodyConstraints.FreezePositionY;

gameObject.GetComponent<Rigidbody>().freezeRotation = true;
    }
}
}
private void OnTriggerStay(Collider other)
{
    if (other.tag == "ReactionFlask")
    {
        if (!gameObject.GetComponent<DragableObject>().drag)
        {
            SpiritLamp = true;
            position.y -= 0.23f;
            gameObject.transform.position = position;
            position.y += 0.23f;
        }
    }
}

private void OnTriggerExit(Collider other)
{
}

// Update is called once per frame
void Update () {
    if(gameObject.GetComponent<DragableObject>().drag)
    {
        if (SpiritLamp)
        {

GameObject.Find("ReactionFlask").GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeRotation |
RigidbodyConstraints.FreezePositionX |
RigidbodyConstraints.FreezePositionZ;
        gameObject.GetComponent<Rigidbody>().constraints
= RigidbodyConstraints.None;
        SpiritLamp = false;
gameObject.transform.Find("Fire").gameObject.SetActive(false);
        Reactions reactions =
GameObject.Find("MenuReaction").GetComponent<StartReaction>().reactions;
GameObject.Find("MenuReaction").transform.Find("Condition").Find("Heat").gameObject.SetActive(false);
        if (reactions != null)
        {
            reactions.temp = 0;

```

```

        }
    }
    if (SpiritLamp == false)
    {
        GameObject.Find("ReactionFlask").GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeRotation |
        RigidbodyConstraints.FreezePositionX |
        RigidbodyConstraints.FreezePositionZ;
        gameObject.GetComponent<Rigidbody>().constraints =
        RigidbodyConstraints.None;
    }
}
public class StartReaction : MonoBehaviour
{
    public GameObject CountElement;
    public List<JSInfo> jsinfos;
    public Reactions reactions { get; set; } = null;
    public GameObject response;
    public GameObject Condition;
    public GameObject AcceptReaction;
    public GameObject Pressure;
    public GameObject Light;
    public GameObject Heat;
    public GameObject ReactionFlask;
    public GameObject Explosion;
    public GameObject game;
    public ParticleSystem fire;
    public ParticleSystem smoke;
    public ResultReaction result;
    private int count;
    private float timeMax;
    private float timeTmp;
    private bool fireOnce;
    private bool smokeOnce;
    private bool explosionExists;

```

```

// use this for initialization
void Start()
{
    fireOnce = false;
    smokeOnce = false;
    explosionExists = false;
    List<int> testlist = new List<int>();
    smoke.Stop();
    fire.Stop();
    timeTmp = 0f;
    timeMax = 0f;
    count = 0;
    jsinfos = null;
}

public void inc(int incCount)
{
    count = count + incCount;

    game.GetComponentInChildren<InputField>().placeholder.GetComponent<Text>().text = count + "";
}

public void countZero()
{
    count = 0;
}

public void dec(int decCount)
{
    if (count > 0)
    {
        count = count - decCount;
    }
}

```

```

game.GetComponentInChildren<InputField>().placeholder.GetComponent<Text>().text = count + "";
    }
}

public void accept()
{
    if (count < 1)
    {
        CountElement.SetActive(false);
        GameObject.Find("Main
Camera").GetComponent<Sampling>().elem = null;
    }
    else
    {
        if (reactions == null)
        {
            reactions = new Reactions();
        }
        Sampling elements = GameObject.Find("Main
Camera").GetComponent<Sampling>();
        Components comp = new Components();
        comp.molar = count;
        comp.molarConcentration = 101f;
        comp.moleculeId =
elements.elem.element.allotropList[0].molecule.id.ToString();
        reactions.componentList.Add(comp);
        GameObject.Find("Main
Camera").GetComponent<Sampling>().elem = null;
        count = 0;
        CountElement.SetActive(false);
    }
}
}

```

```

public void acceptCondition(int id)
{
    if (count > 0)
    {
        if (reactions == null)
        {
            reactions = new Reactions();
        }
        if (id == 1)
        {
            reactions.temp = count;

GameObject.Find("spirtovka").transform.Find("Fire").gameObject.SetActive(true);
        }
        if (id == 2)
        {
            reactions.light = count;
        }
        if (id == 3)
        {
            reactions.pressure = count;
        }
        countZero();
        Debug.Log(JsonUtility.ToJson(reactions));
        Condition.SetActive(false);
    }
    else
    {
        if (id == 1)
        {

```

```

GameObject.Find("spirtovka").transform.Find("Fire").gameObject.SetActive(false);
    }

```

```

    }

```

```

}

```

```

public void acceptReactions()
{
    if (reactions != null)
    {

```

```

GameObject.Find("spirtovka").transform.Find("Fire").gameObject.SetActive(false);

```

```

        AcceptReaction.SetActive(false);
        try
        {
            response.SetActive(true);
            string jsonRequest =
JsonUtility.ToJson(reactions);
            string responseReaction =
SendQuery.POST(host.chemDomain+"reaction/makeReaction",
jsonRequest, CheckAuthorization.ExperimentAccount.Token);
            if (responseReaction != "")
            {
                Debug.Log(responseReaction);
                result =
JsonUtility.FromJson<ResultReaction>(responseReaction);
                response.GetComponentInChildren<Text>().text
= "Реакция прошла успешно";
                reactions = null;

```

```

        foreach (EffectList effect in
result.effectList)
        {
            if (effect.effectType.name == "fire")
            {
                if (effect.effectParamList.Count >
0)
                {
                    if
(effect.effectParamList.FindAll(x => x.effectParamType.name ==
"color").Count != 0)
                    {
                        string color =
effect.effectParamList.FindLast(x => x.effectParamType.name ==
"color").value;

                        Color32 colorBegin = new
Color32(Convert.ToByte(color[0] + color[1]),
Convert.ToByte(color[2] + color[3]), Convert.ToByte(color[4] +
color[5]), 50);

                        fire.startColor =
colorBegin;
                    }
                }
            }
            if (effect.timeTo / 1000f > timeMax)
            {
                timeMax = effect.timeTo / 1000f;
            }
        }
    }
    else
    {
        response.GetComponentInChildren<Text>().text = "Реакция не
прошла";
    }
}

```

```

        }
    }
    catch (Exception e)
    {
        Debug.Log(e.Message.ToString());
    }
    Invoke("Activation", 5f);
}

}

public void changeCondition()
{
    Condition.SetActive(true);
    AcceptReaction.SetActive(false);
}

public void changeOnPressure()
{
    Pressure.SetActive(true);
    Condition.SetActive(false);
    game = Pressure.transform.Find("InputField").gameObject;

    game.GetComponent<InputField>().placeholder.GetComponent<Text>()
    .text = "0";
}

public void changeOnLight()
{
    Light.SetActive(true);
    Condition.SetActive(false);
    game = Light.transform.Find("InputField").gameObject;

    game.GetComponent<InputField>().placeholder.GetComponent<Text>()
    .text = "0";
}

```

```

    }

    public void changeOnHeat()
    {
        Heat.SetActive(true);
        Condition.SetActive(false);
        game = Heat.transform.Find("InputField").gameObject;

        game.GetComponent<InputField>().placeholder.GetComponent<Text>().text = "0";

    }

    private void Activation()
    {
        response.SetActive(false);
    }

    // update is called once per frame
    void Update()
    {

        CountElement.GetComponentInChildren<InputField>().placeholder.GetComponent<Text>().text = count + "";

        if (timeTmp < timeMax)
        {

            timeTmp += Time.deltaTime;
            foreach (EffectList effectList in result.effectList)
            {
                if ((effectList.timeFrom / 1000f < timeTmp) &&
                    (effectList.timeTo / 1000f > timeTmp))

```

```

        {
            if (effectList.effectType.name == "fire")
            {
                if (!fireOnce)
                {
                    fire.Play();
                    fireOnce = true;
                }
                if (effectList.effectParamList.Count >
0)
                {
                    if
(effectList.effectParamList.FindAll(x => x.effectParamType.name
== "color").Count > 1)
                    {
                        string color =
effectList.effectParamList.FindLast(x => x.effectParamType.name
== "color").value;

                        Color32 colorEnd = new
Color32(Convert.ToByte(color[0].ToString() +
color[1].ToString(), 16), Convert.ToByte(color[2].ToString() +
color[3].ToString(), 16), Convert.ToByte(color[4].ToString() +
color[5].ToString(), 16), 125);

                        color =
effectList.effectParamList.Find(x => x.effectParamType.name ==
"color").value;

                        Color32 colorBegin = new
Color32(Convert.ToByte(color[0].ToString() +
color[1].ToString(), 16), Convert.ToByte(color[2].ToString() +
color[3].ToString(), 16), Convert.ToByte(color[4].ToString() +
color[5].ToString(), 16), 125);

```

```

        fire.startColor =
UnityEngine.Color.Lerp(colorBegin, colorEnd, timeTmp /
(effectList.timeTo / 1000f));
    }
    else
    {
        if
(effectList.effectParamList.FindAll(x => x.effectParamType.name
== "color").Count == 1)
        {
            string color =
effectList.effectParamList.FindLast(x => x.effectParamType.name
== "color").value;

            Color32 colorEnd = new
Color32(Convert.ToByte(color[0].ToString() +
color[1].ToString(), 16), Convert.ToByte(color[2].ToString() +
color[3].ToString(), 16), Convert.ToByte(color[4].ToString() +
color[5].ToString(), 16), 125);

            fire.startColor = colorEnd;
        }
    }
}
if (effectList.effectType.name == "gas")
{
    if (!smokeOnce)
    {
        smoke.Play();
        smokeOnce = true;
    }
    if (effectList.effectParamList.Count >
0)
    {

```

```

        if
(effectList.effectParamList.FindAll(x => x.effectParamType.name
== "color").Count > 1)
        {
            string color =
effectList.effectParamList.FindLast(x => x.effectParamType.name
== "color").value;

            Color32 colorEnd = new
Color32(Convert.ToByte(color[0].ToString() +
color[1].ToString(), 16), Convert.ToByte(color[2].ToString() +
color[3].ToString(), 16), Convert.ToByte(color[4].ToString() +
color[5].ToString(), 16), 125);

            color =
effectList.effectParamList.Find(x => x.effectParamType.name ==
"color").value;

            Color32 colorBegin = new
Color32(Convert.ToByte(color[0].ToString() +
color[1].ToString(), 16), Convert.ToByte(color[2].ToString() +
color[3].ToString(), 16), Convert.ToByte(color[4].ToString() +
color[5].ToString(), 16), 125);

            smoke.startColor =
UnityEngine.Color.Lerp(colorBegin, colorEnd, timeTmp /
(effectList.timeTo / 1000f));
        }
        else
        {
            if
(effectList.effectParamList.FindAll(x => x.effectParamType.name
== "color").Count == 1)
            {
                string color =
effectList.effectParamList.FindLast(x => x.effectParamType.name
== "color").value;

```

```

        Color32 colorEnd = new
Color32(Convert.ToByte(color[0].ToString() +
color[1].ToString(), 16), Convert.ToByte(color[2].ToString() +
color[3].ToString(), 16), Convert.ToByte(color[4].ToString() +
color[5].ToString(), 16), 125);

        smoke.startColor = colorEnd;
    }
}
}
}
if (effectList.effectType.name ==
"explosion")
{
    explosionExists = true;
    Instantiate(Explosion,
ReactionFlask.transform.position, Quaternion.identity,
ReactionFlask.transform);
}
}
else
{
    if (effectList.effectType.name == "fire")
    {
        fire.Stop();
    }
    if (effectList.effectType.name == "gas")
    {
        smoke.Stop();
    }
    if (explosionExists)
    {
        Destroy(ReactionFlask.transform.FindChild("ExplosionMobile(Clone
)").gameObject);
    }
}
}
}
}

```

```

        }
    }
}
else
{
    fireOnce = false;
    smokeOnce = false;
    smoke.Stop();
    fire.Stop();
    timeTmp = 0f;
    timeMax = 0f;
}
}

public void Cancel()
{
    reactions = null;
    AcceptReaction.SetActive(false);
}
}

[System.Serializable]
public class Reactions
{
    public float temp;
    public float pressure;
    public float light;
    public List<Components> componentList;

    public Reactions()
    {
        temp = 293;
        pressure = 101000;
        componentList = new List<Components>();
    }

    public Reactions(float temp, float pressure, float light,
        List<Components> componentList)

```

```

        {
            this.temp = temp;
            this.pressure = pressure;
            this.light = light;
            this.componentList = componentList;
        }
    }

[System.Serializable]
public class Components
{
    public string moleculeId;
    public float molar;
    public float molarConcentration;
    public float particleSize;
}

/// <summary>
/// The structure of the response to the service reactions
/// </summary>

[System.Serializable]
public class ResultReaction
{
    public List<ResultList> resultList;
    public List<EffectList> effectList;

    public ResultReaction()
    {
        resultList = new List<ResultList>();
        effectList = new List<EffectList>();
    }

    public ResultReaction(List<ResultList> resultList,
List<EffectList> effectList)
    {
        this.resultList = resultList;
        this.effectList = effectList;
    }
}

[System.Serializable]
public class EffectList
{
    public List<EffectParamList> effectParamList;
    public EffectType effectType;
    public string id;
    public long timeFrom;
    public long timeTo;
}

```

```
[System.Serializable]
public class EffectParamList
{
    public EffectParamType effectParamType;
    public string id;
    public string value;
}
```

```
[System.Serializable]
public class EffectType
{
    public string id;
    public string name;
}
```

```
[System.Serializable]
public class EffectParamType
{
    public string id;
    public string name;
}
```

```
[System.Serializable]
public class ResultList
{
    public float Concentration;
    public string id;
    public float molar;
    public string name;
}
```

```
public class TaskOutput : MonoBehaviour
{
    private JSResp jsResp;
    public GameObject game;
    //Вывод задания
    // Use this for initialization
    void Start()
    {
        if (Tasks.task != "")
        {
            try
            {
                jsResp =
                JsonUtility.FromJson<JSResp>(Tasks.task);
                game.SetActive(true);
                game.GetComponentInChildren<Text>().text =
                jsResp.name;
                GameObject.Find("acceptTask").SetActive(true);
            }
            catch { }
        }
    }
}
```

```

        Invoke("Activation", 10f);
    }
    catch (Exception e)
    {
        Debug.Log("Ошибка при запросе: " +
e.Message.ToString());
    }
}

private void Activation()
{
    game.SetActive(false);
}

// Update is called once per frame
void Update()
{
}
}

[System.Serializable]
public class App
{
    public string androidPath { get; set; }
    public string id { get; set; }
    public string name { get; set; }
}

// struct response
[DataContract]
public class JSResp
{
    [DataMember]
    public App app;
    [DataMember]
    public string id;
    [DataMember]
    public string json;
    [DataMember]
    public string name;
}

public class Tasks : MonoBehaviour
{
    public static string task { get; private set; } = "";
    // Use this for initialization
    void Start()

```

```

{
    if (Application.platform == RuntimePlatform.Android)
    {
        int taskId = 0;

        AndroidJavaClass UnityPlayer = new
        AndroidJavaClass("com.unity3d.player.UnityPlayer");
        AndroidJavaObject currentActivity =
        UnityPlayer.GetStatic<AndroidJavaObject>("currentActivity");

        AndroidJavaObject intent =
        currentActivity.Call<AndroidJavaObject>("getIntent");
        bool hasExtra = intent.Call<bool>("hasExtra",
        "taskId");

        if (hasExtra)
        {
            AndroidJavaObject extras =
            intent.Call<AndroidJavaObject>("getExtras");
            taskId = extras.Call<int>("getInt", "taskId");
            task =
            SendQuery.GET("https://appsapi.experiment.school/v01/services/ap
            p/task/" + taskId + "/2");
            Application.LoadLevel("MainScene");
        }
    }

    // Update is called once per frame
    private void Update()
    {

    }
}

public class UpdateMenu : MonoBehaviour {

    public GameObject menu;

    // Use this for initialization
    void Start () {
        gameObject.transform.position = menu.transform.position;
    }

    // Update is called once per frame
    void Update () {
        menu.transform.position = gameObject.transform.position;
    }
}

```

Приложение Г
Результаты структурного тестирования
(рекомендуемое)

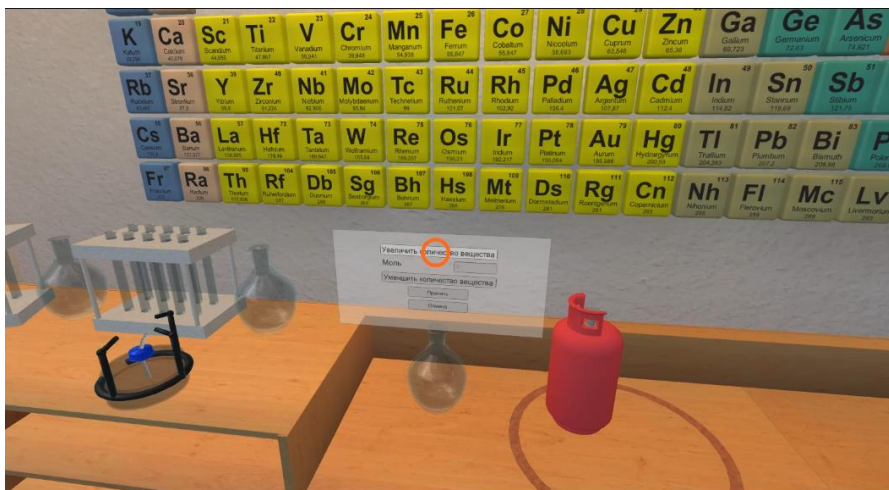


Рисунок Г.1 – Результат теста когда пользователь пытается добавить элемент

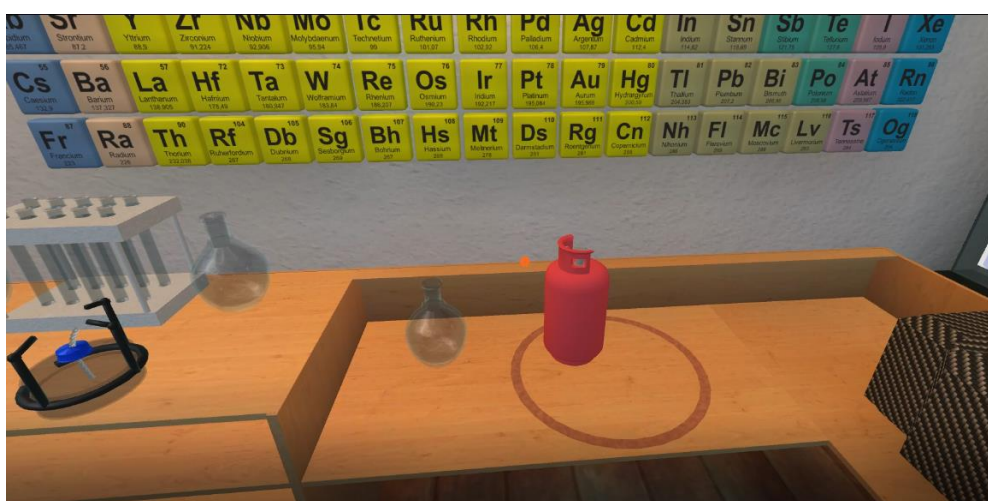


Рисунок Г.2 – Результат теста когда пользователь не взаимодействует с колбой

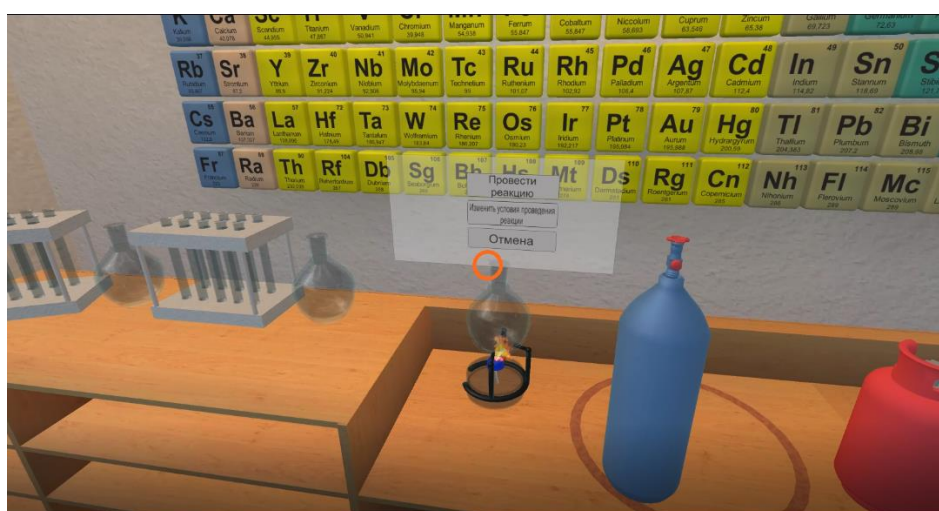


Рисунок Г.3 – Результат теста когда пользователь взаимодействует с не пустой колбой