Министерство образования Республики Беларусь

Учреждения образования «БЕЛОРУССКИЙ
ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий
Кафедра Информационных систем и технологий
Специальность 1-40 05 01 Информационные системы и технологии
Специализация 1-40 05 01-03 Информационные системы и технологии (издательско-полиграфический комплекс)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ДИПЛОМНОГО ПРОЕКТА НА ТЕМУ:

Симулятор швейной машинки

Обучающийся	4 курс, 2 группа			К.А. Корело
	курс, группа	дата	подпись	И. О. Ф.
Руководитель дипломного проекта	к.ф.-м.н., доцент			Н.И. Гурин
	должность, ученая степень, ученое звание	дата	подпись	И. О. Ф.
И.о. заведующего кафедрой	к.т.н.			Е.А. Блинова
	ученая степень, ученое звание	дата	подпись	И. О. Ф.
Консультант	ст. преп.			А.С. Соболевский
	должность, ученая степень, ученое звание	дата	подпись	И. О. Ф.
Нормоконтролер	ст. преп.			О.А. Нистюк
	должность, ученая степень, ученое звание	дата	подпись	И. О. Ф.

Дипломный проект защищен с оценкой _____

Председатель ГЭК	к.т.н., доцент			В.К. Дюбков
	должность, ученая степень, ученое звание	дата	подпись	И. О. Ф.

Минск 2025

Оглавление

					ДП 00.00.ПЗ		
		ФИО	Подпись	Дата			
Разраб.		*Корело К.А..*			*Содержание*	Лит.	Лист 1 / Листов 2
Пров.		*Гурин Н.И.*				У	
Н. контр.		*Нистюк О.А.*					*БГТУ 1-40 05 01, 2025*
Утв.		*Блинова Е.А.*					

Реферат

Пояснительная записка дипломного проекта содержит 61 страницу пояснительной записки, 6 таблиц, 10 формул, 20 иллюстраций, 10 источников литературы, 9 приложений.

3-D СИМУЛЯТОР, ЯЗЫК ПРОГРАММИРОВАНИЯ C#, UNITY, 3DS MAX

Целью дипломного проекта является создание интуитивно понятного симулятора для изучения работы швейной машинки.

Пояснительная записка состоит из введения, шести разделов и заключения.

В введении предоставляется информация о текущем состоянии в сфере рассматриваемой проблемы, а также поставлены цели дипломного проектирования.

Первая глава проводится аналитический обзор литературы и приложений-аналогов по тематике дипломного проекта.

Вторая глава посвящена обзору средств разработки и содержит описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвертой главе описывается контрольный пример, с проведением тестирования, и показывается поведения системы при разных внештатных ситуациях.

В пятой главе описано руководство пользователя, позволяющее подробно понять интерфейс программного средства.

В шестой главе производится расчет экономических параметров и себестоимости программного средства, разработанного в рамках дипломного проекта.

В заключении были подведены итоги по проделанной работе.

Abstract

The explanatory note of the graduation project contains 61 pages of explanatory notes, 6 tables, 10 formulas, 20 illustrations, 10 sources of literature, 9 appendices.

3-D SIMULATOR, PROGRAMMING LANGUAGE C#, UNITY, 3DS MAX

The purpose of the thesis project is to create an intuitive simulator for studying the operation of a sewing machine.

The explanatory note consists of an introduction, six sections and a conclusion.

The introduction provides information about the current state of the problem under consideration, as well as sets goals for graduate design.

The first chapter provides an analytical review of the literature on the subject of the thesis project.

The second chapter is devoted to an overview of the development tools and contains a description of the technologies used during the project.

The third chapter describes the development process, the principles of operation and the purpose of the created components of the project.

The fourth chapter describes a test case, with testing, and shows the behavior of the system in various emergency situations.

The fifth chapter describes the user's guide, which allows you to understand the interface of the software tool in detail.

In the sixth chapter, the economic parameters and the cost of the software developed as part of the graduation project are calculated.

In conclusion, the results of the work done were summarize.

Введение

Современные темпы развития информационных технологий требуют от системы образования на всех уровнях стремительной адаптации и трансформации. Одним из множества способов решения проблемы является разработка симуляторов различных научных и технических установок. Симуляторы для обучения – это интерактивные модели, имитаторы управления процессом, оборудованием, механизмом, а также имитаторы ситуации.

Главная задача – подготовить начинающих специалистов к работе с реальными машинами и устройствами, функционал которых разнообразен и сложен для освоения. В свою очередь, это повышает безопасность учащихся и снижает вероятность порчи оборудования.

В настоящее время использование симуляторов особо актуально для учащихся среднеспециальных и высших учебных заведений. Активным внедрением данного программного и аппаратного обеспечения занимается Белорусский государственный медицинский университет, имитируя различные виды хирургических вмешательств для обучения студентов без необходимости подвергать пациентов риску. Также применение этих технологий используется в области повышения квалификации действующих специалистов автотранспортной отрасли нашей страны, позволяя моделировать различные ситуации на дороге.

С точки зрения интересов общества и государства, симуляторы для обучения, могут быть полезным инструментом, стимулирующим развитие сразу нескольких важных цифровых отраслей. Кроме того, симуляторы позволят эффективно распоряжаться ресурсами учреждений образования для обеспечения наиболее качественного процесса обучения. В связи с таким развитием вышеописанной отрасли поставлена следующая цель.

Целью данного проекта является создание симулятора швейной машины, направленного на ознакомление с механиками работы этого оборудования.

Для реализации данного проекта были решены следующие задачи:

– найти и проанализировать учебные симуляторы различных установок;
– спроектировать симулятор швейной машины;
– разработать симулятор швейной машины;
– протестировать симулятор швейной машины;
– провести технико-экономическое обоснование проекта;
– написать руководство пользователя.

1 Анализ приложений-аналогов и постановка задач

В ходе данного раздела будет выполнен анализ приложений-аналогов, а также осуществлена постановка задач для дальнейшей реализации дипломного проекта. Постановка задач необходима для определения целей и задач, которые должны быть решены в процессе разработки симулятора. Она помогает сформулировать требования к приложению и определить его функциональность. Обзор аналогов важен для изучения уже существующих приложений, которые решают похожие задачи. Это позволяет понять, какие функции и возможности присутствуют на рынке, какие аспекты пользователи ценят больше всего и какие ошибки или недочеты есть у конкурентов. На основе обзора аналогов можно сделать выводы о том, что нужно улучшить или добавить в разрабатываемом симуляторе, чтобы он был конкурентоспособным и удовлетворял потребности пользователей.

В качестве программ аналогов мною были выбраны следующие приложения сходной тематики:

– «Машинист метро 3D» [1];
– «Flight Simulator» [2];
– «Метро симулятор» [3].

1.1 Обзор аналогов

1.1.1 Симулятор «Машинист метро 3D»

Машинист метро 3D – симулятор, управления поездом. Машинист метро 3D это приложение с удобным интерфейсом, что делает его более доступным для пользователей, которые не имели опыта управления поездами ранее, обучающее основам работы машиниста в метро.

Основные возможности приложения:

– прохождение обучения;
– тренировка опытных пользователей;
– настройка уроков по времени прохождения;
– сравнение результатов с другими пользователями;
– повышение уровня сложности.

При входе в приложение открывается главная страница, содержащая информацию об уровне машиниста, выбор режима симулятора и дополнительную информацию – язык и таблица результатов. Простой и интуитивно понятный дизайн интерфейса позволяет быстро освоиться даже новичкам.

После выбора режима пользователь может приступить к обучению или практике. Главная страница представлена на рисунке 1.1.

Рисунок 1.1 – Главная страница симулятора

Выбирая режим обучения, открывается окно с рабочим местом машиниста, множеством кнопок и приветственным текстом объясняющим, что выбран режим обучения и кнопкой «Продолжить», которая запустит процесс обучения. Далее последует пошаговая инструкция управления поездом. Начало обучения представлено на рисунке 1.2.

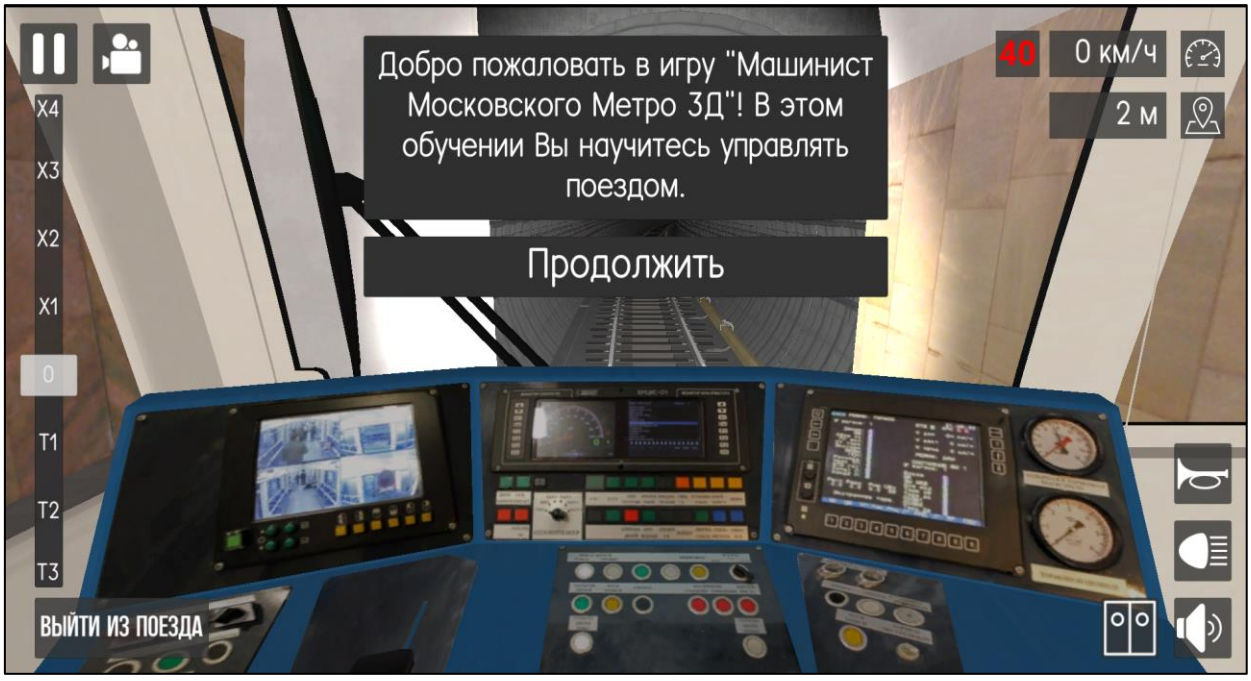

Рисунок 1.2 – Режим обучения

Выбирая режим для опытных пользователей, открывается окно настройки урока, в котором можно регулировать время занятия и маршрут следования транспорта, а также модель и поезда для виртуального обучения. Настройка урока представлена на рисунке 1.3.

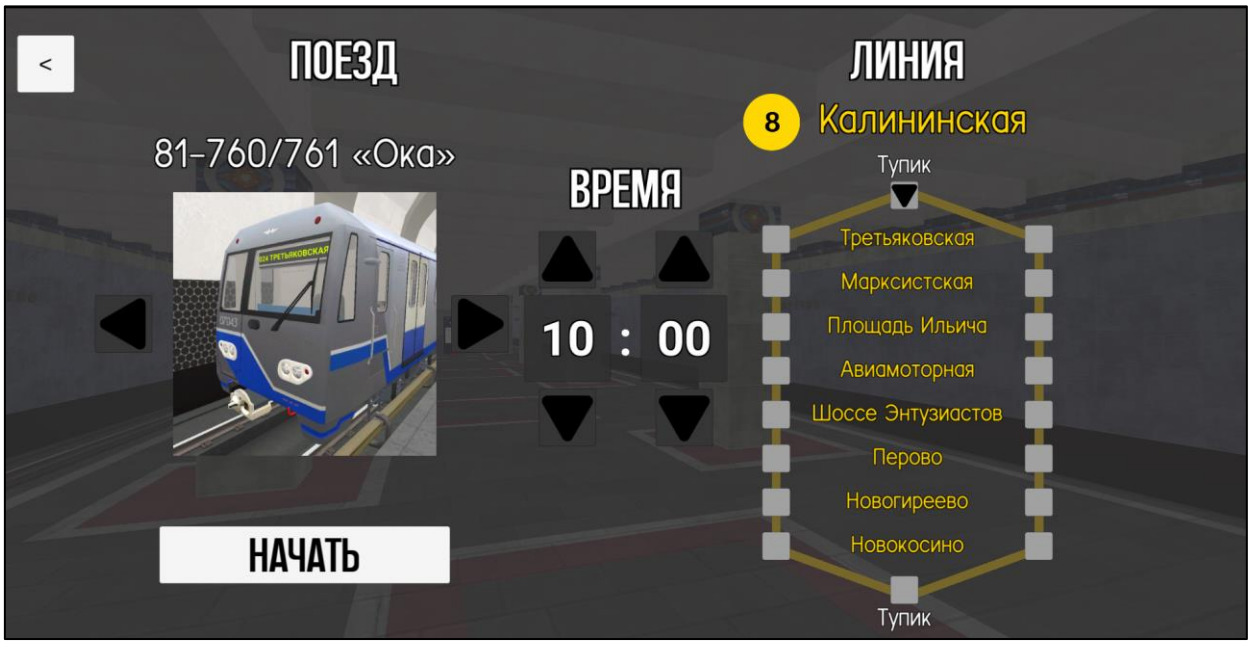

Рисунок 1.3 – Страница настройки занятия

Среди недостатков можно выделить следующие пункты: выбор типа транспорта не влияет на механику управления поездом, так как симулятор знакомит лишь с общим принципом работы всех поездов метро; если пользователь забыл последовательность действий нет возможности получить подсказку, даже на простом уровне сложности; в режиме обучения нет возможности вернуться к предыдущим этапам при необходимости. Также в процессе использования было выявлено, что система в независимости от режима использования симулятора, намеренно округляет результаты в сторону пользователя, что делает практически невозможным ситуацию, при которой пользователь некорректно выполнит задание. Задание будет считаться выполненным неправильно только в том случае если пользователь совершит грубую ошибку в управлении поездом.

1.1.2 Симулятор «Flight Simulator»

Симуляторы управления транспортом становятся все более популярными благодаря своей доступности и образовательной ценности. Они позволяют пользователям без специальной подготовки погрузиться в процесс управления сложными машинами.

Приложение «Flight Simulator» – это симулятор полетов на самолете, предназначенный для пользователей, желающих освоить азы пилотирования. Оно сочетает простоту управления с реалистичным воспроизведением полетных процессов. На главной странице представлены различные мисси с кратким

описанием того, что необходимо выполнить для успешного завершения урока. Главная страница представлена на рисунке 1.4. Удобный интерфейс и наглядные инструкции помогают пользователям быстро освоить управление самолетом, а реалистичная графика создает эффект полного погружения в процесс полета.

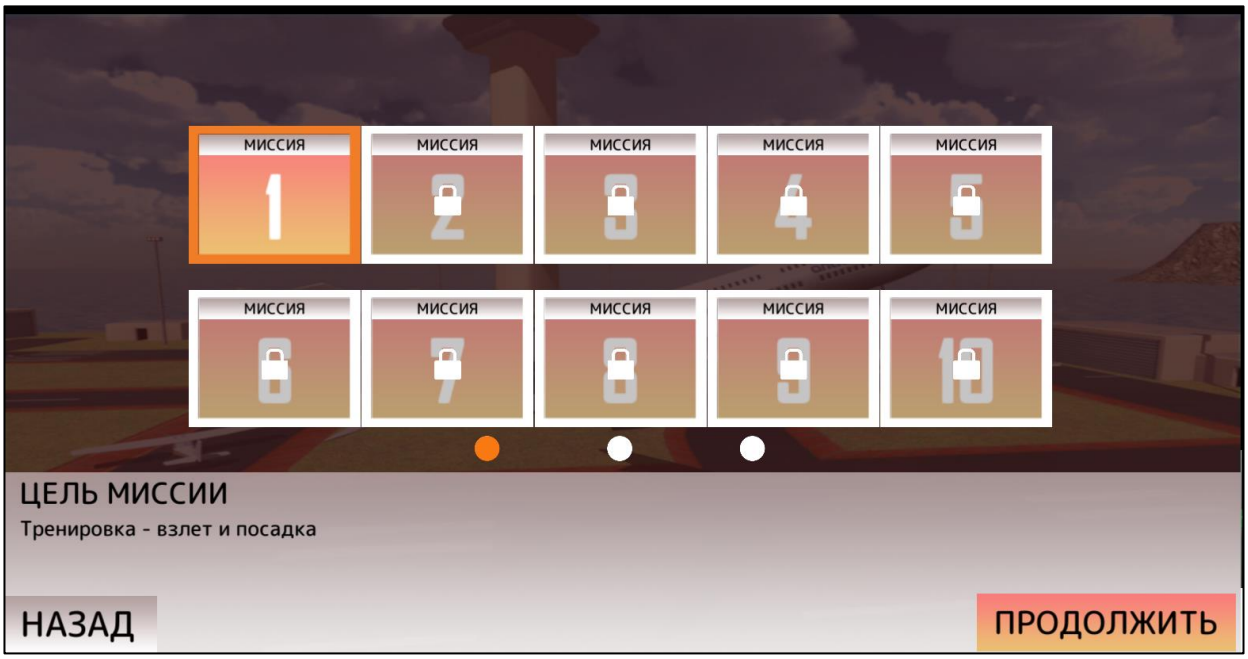

Рисунок 1.4 – Главная страница

В начале каждой миссии пользователь знакомится с новыми механиками управления. На экране отображаются элементы управления самолетом и время, отведенное на выполнение задачи. Пример процесса обучения показан на рисунке 1.5. Реалистичная графика усиливает эффект погружения в симуляцию.

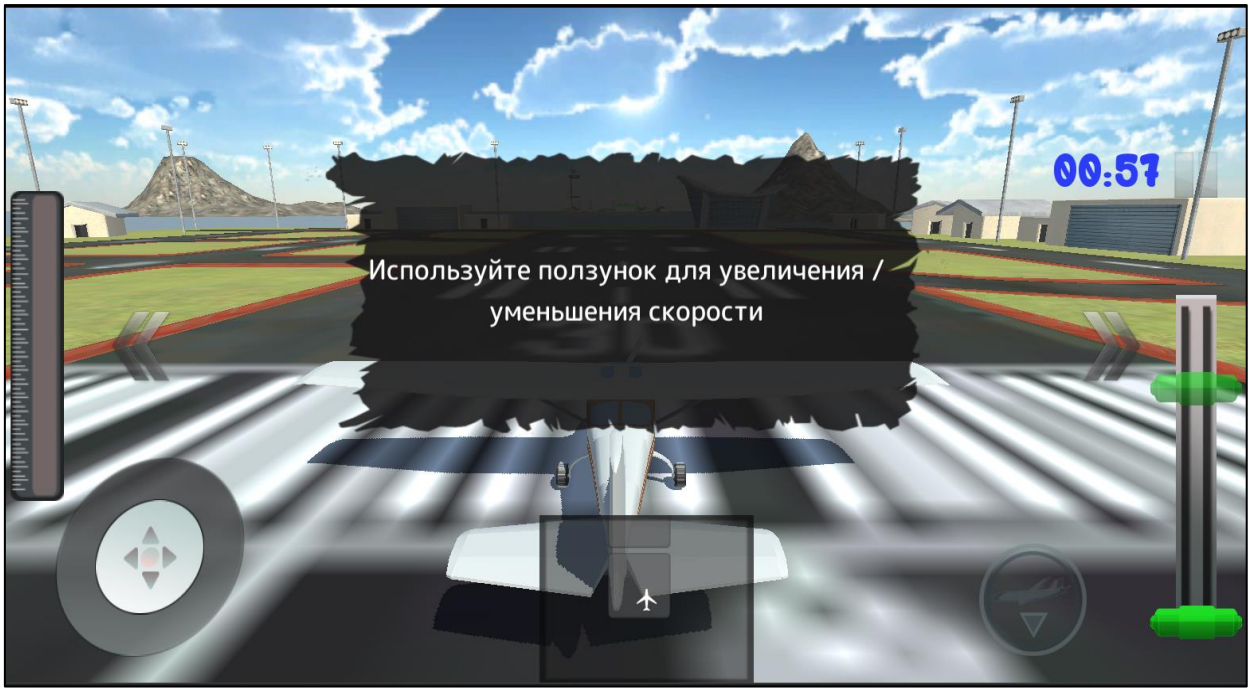

Рисунок 1.5 – Пример обучения в начале миссии

Обучение в симуляторе осуществляется через пошаговые инструкции, объясняющие, какие действия необходимо выполнить, как показано на рисунке 1.5. После перемещения ползунка пользователем симулятор предоставляет новые указания для продолжения. По завершении обучения система предлагает выполнить миссию без подсказок. При неудачном выполнении миссия считается проваленной, и ее нужно повторить.

К недостаткам можно отнести отсутствие вводного обучения основам управления, невозможность выбора уровня сложности, а также необходимость успешного завершения предыдущих миссий для доступа к новым. Кроме того, интерфейс может показаться недостаточно интуитивным для новичков. Отсутствие гибких настроек ограничивает индивидуальную адаптацию обучения.

1.1.3 Симулятор «Метро симулятор»

Приложение «Метро симулятор», подобно первому рассмотренному аналогу, представляет собой симулятор управления поездом метрополитена. Пользователь может выбрать тип транспорта и маршрут для следования, как и в предыдущем приложении. Меню выбора транспорта показано на рисунке 1.6. Интерфейс приложения интуитивно понятен, что делает его доступным для пользователей любого уровня подготовки. Симулятор предлагает реалистичное воспроизведение процесса управления поездом, включая различные сценарии движения. Доступны настройки для регулировки сложности и условий маршрута. Графическое оформление создает эффект присутствия в кабине машиниста.

Рисунок 1.6 – Меню выбора транспортного средства

После выбора транспорта и маршрута запускается симулятор управления поездом. В данном аналоге представлены две перспективы с которых можно контролировать работу транспорта: «вид из кабины» и «салон». В первом случае представлен вид на тоннель, в котором находится поезд и помогает

пользователю ориентироваться, для определения приближения к станции и своевременной остановки. «Вид из кабины» представлен на рисунке 1.7.

Рисунок 1.7 – «Вид из кабины»

«Салон» дает обзор на то, что происходит во время посадки и позволяет контролировать врем, которое поезд продолжает стоять на станции. Это дает представление пользователю о том, как выглядит работа машиниста и зону его ответственности. Вид из салона представлен на рисунке 1.8.

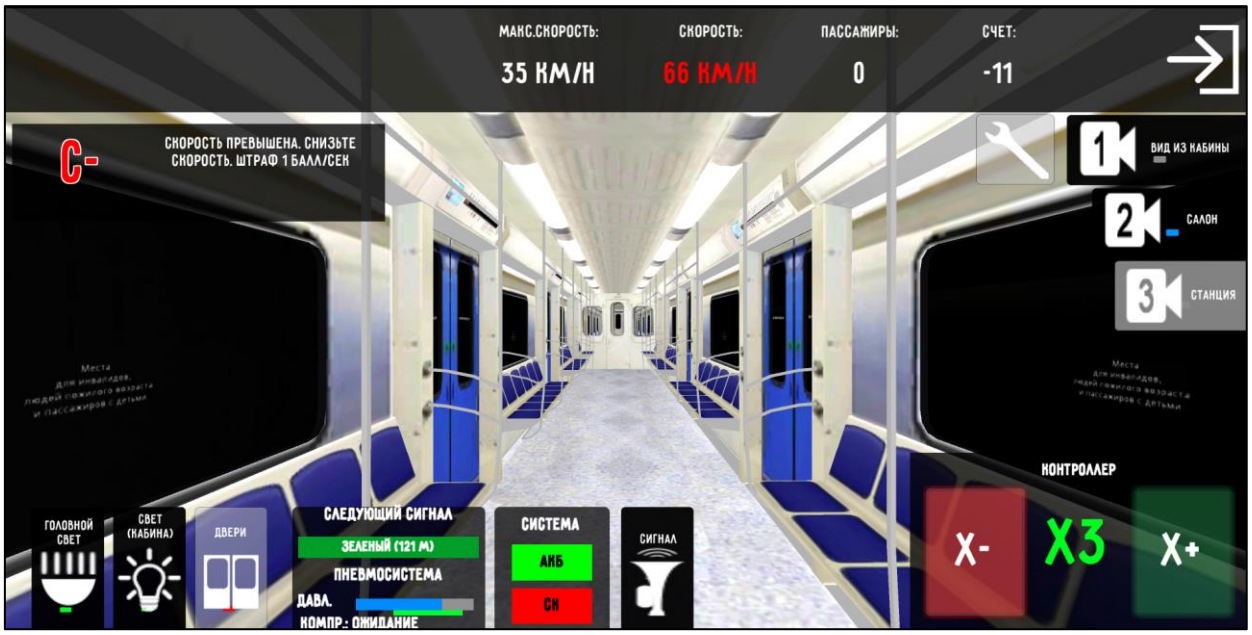

Рисунок 1.8 – Вид из салона

Для определения правильности выполнения задания используется система баллов, которую также можно увидеть на рисунке 1.8. В правом верхнем углу написано количество баллов, присвоенное пользователю за время выполнения задания, а слева причина ошибки, если таковая имеется.

К недостаткам можно отнести отсутствие обучения в самом начале, а также не самый приятный и понятный пользователю дизайн. Все это в совокупности приводит к тому, что даже опытный пользователь не сразу может разобраться в приложении и потратит одну две попытки прохождения симулятора, на освоение механик.

1.2 Постановка задачи

Целью дипломного проекта является создание симулятора швейной машинки. При обзоре аналогов, для разрабатываемого проекта определился следующий функционал:

– возможность передвижения пользователя по симулятору;
– выбор вида ткани и цвета ниток;
– обводка трафарета по ткани;
– обрезка ткани;
– сшивание кусочков ткани на швейной машине;
– переключение вида строчек: прямая и зигзагообразная;
– изучение элементов швейной машинки с помощью интерфейса.

Для разработки приложения планируется использовать современные технологии разработки симуляторов. В результате успешной реализации проекта ожидается создание удобного и надежного инструмента для ознакомления с механиками работы швейной машины, который будет повышать комфорт и безопасность пользователей, а также экономить энергоресурсы и снижать расходы на их обслуживание.

1.3 Выводы по разделу

При написании данного раздела был определен основной функционал будущего приложения. Были выявлены сильные и слабые стороны приложений-аналогов, что позволило сформировать требования к собственному продукту. Было установлено, что основной функционал приложения должен включать в себя возможность быстрого и удобного доступа к информации, а также интуитивно понятный пользовательский интерфейс.

Одной из слабых сторон аналогичных приложений было отсутствие обучающего материала, который позволит пользователям быстро адаптироваться и начать ориентироваться в приложении.

Также важным аспектом стало обеспечение интуитивно понятного и приятного дизайна, что в совокупности с вышеописанным недостатком может создавать проблему. Поэтому при разработке симулятора для данного дипломного проекта эти вопросы будут учтены

В целом, проведенный анализ позволил определить ключевые характеристики будущего приложения и сформулировать требования, которые необходимо учесть при его разработке.

2 Проектирование симулятора

В разделе осуществляется проектирование симулятора швейной машинки, а также описание используемых инструментов и технологий для разработки проекта. На этапе проектирования происходит определение структуры системы, описание взаимодействия компонентов, а также проектируются пользовательские интерфейсы.

В процессе проектирования симулятора были определены функциональные требования к программному средству, разработана диаграмма взаимодействия между его основными компонентами, спроектированы пользовательские интерфейсы с учетом удобства взаимодействия и наглядного отображения симулируемых процессов. Особое внимание уделялось реалистичности модели, логике симуляции и обеспечению корректного отображения данных, имитирующих поведение реальной системы.

2.1 Функциональные требования

Разработка программного обеспечения требует четкого определения его функциональности и взаимодействия с пользователем. Основной задачей является создание приложения, которое отвечает ожиданиям целевой аудитории и обеспечивает удобство использования.

Функциональные требования определяют, какие задачи должно выполнять разрабатываемое программное обеспечение, а также описывают поведение системы с точки зрения пользователя. Симулятор швейной машины, реализуемый в рамках данной дипломной работы, предназначен для имитации работы бытовой швейной машины в интерактивной форме, с целью обучения и демонстрации принципов ее функционирования. Приложение должно обладать рядом функций, обеспечивающих реалистичную, визуально и аудиально достоверную симуляцию, а также интуитивно понятный пользовательский интерфейс.

Программное обеспечение должно представлять собой однопользовательское приложение с возможностью запуска на персональном компьютере под управлением операционных систем семейства Windows. Пользовательский интерфейс должен быть реализован в виде интерактивной сцены с 3D-моделью швейной машинки, элементами управления и рабочей поверхностью, имитирующей ткань. Интерфейс должен быть простым и интуитивно понятным, обеспечивая доступ к основным функциям приложения без необходимости предварительного изучения документации.

Разработка ведется с использованием игрового движка Unity.

Он позволяет реализовать 3D-визуализацию, анимацию и обработку пользовательского ввода в интерактивной форме.

Основные функциональные возможности приложения включают:

– передвижение по пространству симулятора. Данная функция предоставляет пользователю возможность свободно перемещаться в пределах виртуального пространства, в котором располагается швейная машинка;

– выбор ткани и цвета ниток;

– обводка трафарета по ткани. Функция обводки трафарета предназначена для имитации ручной разметки ткани по заранее заданному шаблону;

– обрезка ткани. Функция позволяет пользователю выполнять виртуальную обрезку ткани в соответствии с заданной формой или линиями разметки;

– сшивание кусочков ткани на швейной машинке;

– сшивание подготовленных кусков ткани с помощью швейной машинки. Пользователь размещает фрагменты ткани на рабочей поверхности и осуществляет их соединение, управляя машинкой;

– изучение элементов швейной машинки с помощью интерфейса. В рамках обучающего режима реализована функция ознакомления пользователя с основными элементами швейной машинки;

– завершение работы с приложением и выход из симуляции.

2.2 Диаграмма вариантов использования

Диаграмма вариантов использования (Use Case Diagram) – это один из ключевых инструментов объектно-ориентированного анализа, применяемый для моделирования функциональных требований к системе. Она отображает взаимодействие пользователя (актора) с системой через сценарии использования (use cases), что позволяет определить границы системы, выявить основные функции и установить, кто инициирует взаимодействие с теми или иными компонентами.

Важно отметить, что для создания таких диаграмм используется стандартный язык моделирования – UML (Unified Modeling Language). Он является универсальным и широко используемым инструментом для визуального моделирования различных аспектов программных систем, включая их структуру, поведение и взаимодействие. Язык UML включает в себя различные виды диаграмм, предназначенные для описания разных сторон системы.

Для диаграмм вариантов использования используется конкретный подтип UML, который называется диаграмма вариантов использования (Use Case Diagram). Она является частью диаграмм поведения UML и служит для моделирования внешних взаимодействий системы с пользователями (акторами), а также для отображения основных бизнес-процессов или функциональных требований.

Основные элементы диаграммы вариантов использования в UML:

– акторы – в контексте симулятора швейной машинки актором будет являться, например, пользователь (обучающийся);

– варианты использования – функции или действия, которые система предоставляет актору;

– связи – линии, соединяющие акторов с вариантами использования, показывающие, какие действия могут выполняться каждым актором;

– границы системы – прямоугольник, обозначающий границу между системой и внешним миром, в пределах которой осуществляются все действия;

– расширения (Extend) и включения (Include) – связи между вариантами использования, показывающие, что один вариант может быть выполнен в дополнение или в рамках другого.

В контексте разработки симулятора швейной машинки диаграмма вариантов использования служит для представления всех возможных действий, которые может выполнять пользователь в процессе работы с приложением. Основной целью данной диаграммы является обеспечение целостного представления о поведении системы на начальном этапе проектирования, а также формирование базы для последующего построения архитектуры, интерфейса и логики приложения.

Диаграмма вариантов использования представлена на рисунке 2.1.

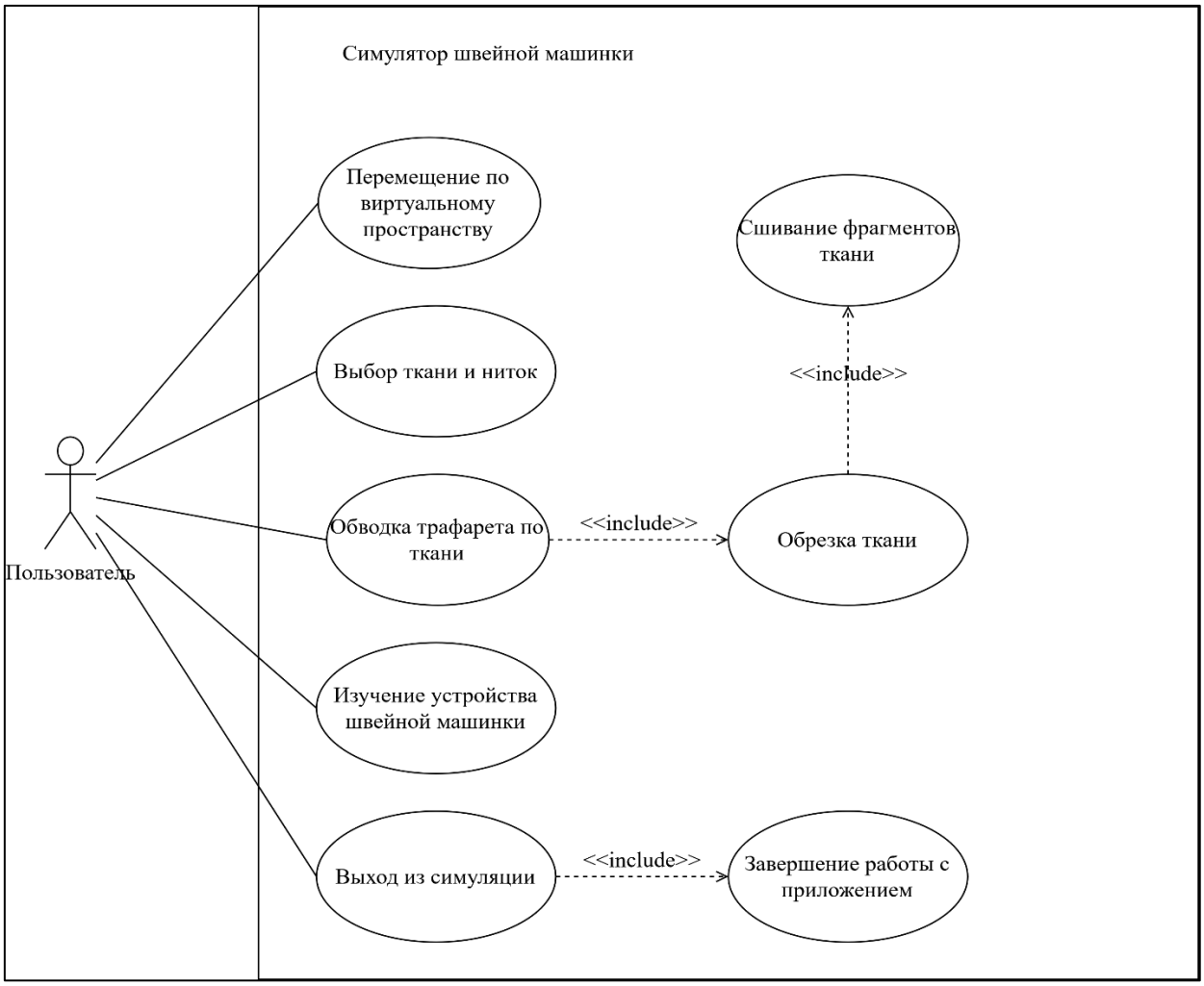

Рисунок 2.1 – Диаграмма вариантов использования

На разработанной диаграмме в качестве актора выступает пользователь (обучающийся), который взаимодействует с системой через графический интерфейс. К числу основных вариантов использования, реализуемых в 3D-симуляторе, относятся:

– перемещение по виртуальному пространству симулятора, включая обзор и навигацию с помощью клавиш WASD;

– выбор ткани и ниток, с возможностью задания цвета;

– обводка трафарета по ткани для управления швейной машинкой;

– обрезка ткани по линии разметки;

– сшивание фрагментов ткани с визуализацией результата;

– изучение устройства швейной машинки через обучающий интерфейс;

– завершение работы с приложением и выход из симуляции.

Диаграмма четко демонстрирует границу между системой и пользователем, обеспечивая наглядное отображение всей функциональности приложения.

2.3 Архитектура симулятора

Архитектура симулятора разработана с учетом модульности, что обеспечивает эффективное взаимодействие между его ключевыми компонентами и позволяет легко адаптировать систему к новым требованиям или расширениям. Структура, представленная на рисунке 2.2, включает два основных уровня: слой пользовательского интерфейса и слой логики, которые связаны между собой через специализированную систему сообщений, играющую роль центрального посредника. Этот подход обеспечивает четкое разделение функций, упрощает отладку и поддержание системы, а также способствует масштабируемости при добавлении новых возможностей.

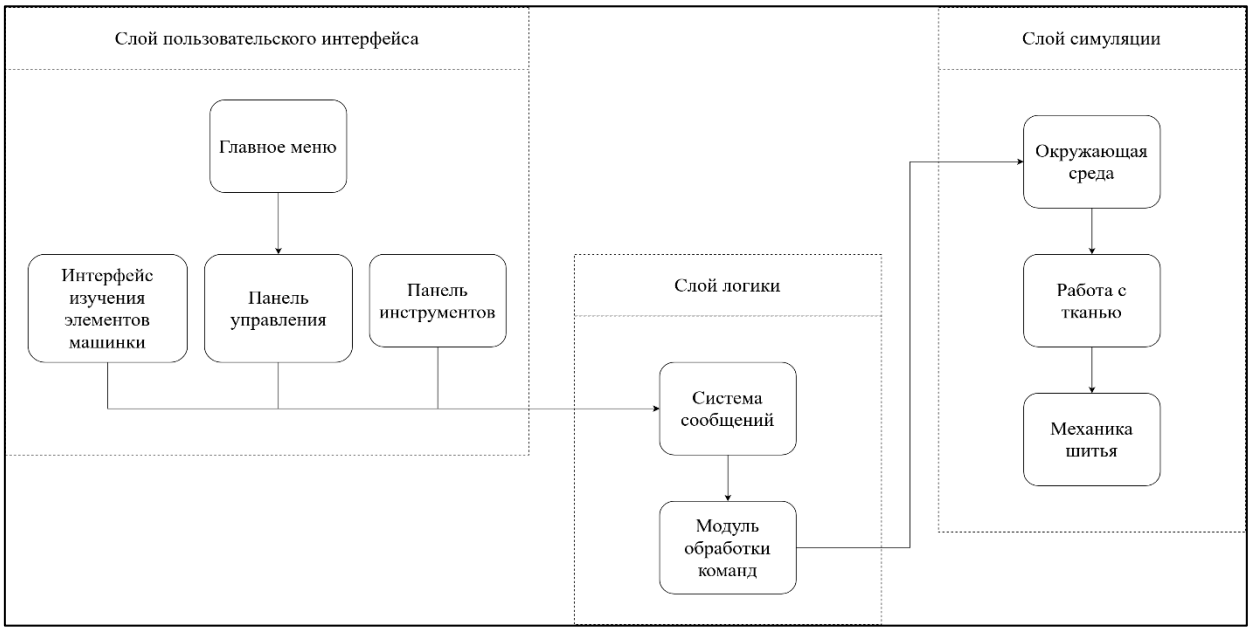

Рисунок 2.2 – Общая архитектура симулятора

Слой пользовательского интерфейса начинается с главного меню, которое служит центральным узлом для доступа ко всем функциям симулятора. Из него пользователь может перейти к панели управления, где доступны настройки громкости и полноэкранного режима, что позволяет адаптировать среду под индивидуальные предпочтения. Также предусмотрена панель

инструментов, представляющая инвентарь, где можно помещать и извлекать ткани, обеспечивая интерактивное управление материалами. Интерфейс изучения элементов машинки предоставляет возможность детального ознакомления с конструктивными частями, такими как игла, лапка и маховик, поддерживая образовательную цель приложения.

Слой логики включает систему симуляции, которая обрабатывает команды, поступающие через систему сообщений, и управляет основными операциями симулятора. Эта система взаимодействует с модулем обучения, обеспечивающим теоретические материалы для пользователей, и модулем шитья, отвечающим за моделирование движения иглы и создания стежков. Система сообщений выступает ключевым звеном, передавая пользовательские команды от интерфейса к логическому слою и возвращая результаты для отображения, что гарантирует плавную и синхронизированную работу.

Дополнительно архитектура учитывает возможность интеграции дополнительных модулей, таких как модуль тестирования или расширенные настройки, без значительных изменений в существующей структуре. Это достигается за счет модульного дизайна, где каждый компонент имеет четко определенные функции и точки взаимодействия. Такая гибкость особенно важна для образовательных симуляторов, где могут возникнуть потребности в адаптации под конкретные учебные программы или добавлении новых типов тканей и швейных техник.

2.4 Проектирование основных алгоритмов

Проектирование логики взаимодействия пользователя с симулятором осуществлялось с использованием классических блок-схем, отражающих этапы работы и принятия решений. Основной алгоритм взаимодействия построен по линейно-ветвящейся структуре и делится на три ключевых логических части, соединенных переходными точками, что обеспечивает читаемость даже при увеличении сложности схемы. Блок-схему алгоритма работы симулятора можно увидеть в приложении В.

В начале алгоритма пользователь запускает симулятор и переходит к выбору ткани, а затем – к этапу подготовки трафарета. Здесь реализован модуль последовательных действий: выбор трафарета, его обводка, обрезка и перенос заготовки на основную сцену. После этого осуществляется переход к швейной машинке и извлечение выкройки из инвентаря, после чего управление передается на следующий этап.

На втором этапе реализовано ветвление: пользователь может либо просмотреть элементы установки (для изучения устройства машинки), либо сразу перейти к выбору цвета нитки. Независимо от выбора, далее следуют действия по вставке шпульки и размещению ткани под иглу. Эти шаги выполняются последовательно, без дополнительных условий. Кроме того, система предоставляет пользователю возможность сохранить текущие настройки для последующего использования, что повышает удобство работы.

Завершающий этап – логика выбора строчки. Пользователю предлагается выбрать тип строчки: прямую или зигзагообразную. Выбор влияет только на визуальное оформление анимации процесса шитья, а не на структуру алгоритма. После выполнения операции шитья реализован выход из симулятора, завершающий выполнение алгоритма.

2.5 Выводы по разделу

На этапе проектирования симулятора швейной машины были определены его функциональные возможности и техническая архитектура. Сформулированы ключевые требования, включая выбор материалов, обводку трафаретов, обрезку ткани, выполнение швов и изучение компонентов машины через интерактивный интерфейс, что создает основу для эффективного образовательного инструмента. Разработанная в стандарте UML диаграмма вариантов использования четко очертила границы системы и основные сценарии взаимодействия пользователя с симулятором, обеспечив точность реализации проекта и минимизацию потенциальных ошибок.

Архитектурная модель системы состоит из двух основных уровней: пользовательского интерфейса и логики, связанных через систему сообщений, что гарантирует модульность, гибкость и масштабируемость. Пользовательский интерфейс включает панель управления с настройками громкости и полноэкранного режима, инвентарь для работы с тканями, интерфейс изучения элементов машины и модуль шитья, интегрированный с системой симуляции. Такое разделение функций упрощает процессы разработки, тестирования и будущей модернизации, позволяя создавать универсальный инструмент для обучения, адаптируемый к различным образовательным задачам.

Результаты проектирования заложили прочный фундамент для создания интуитивно понятного, надежного и визуально привлекательного приложения. Оно способствует повышению качества обучения и безопасности при освоении швейного дела, предоставляя пользователям удобный и реалистичный опыт. Использование UML-диаграмм и блок-схем позволило систематизировать процессы разработки, визуализировать взаимодействие компонентов и исключить избыточные функции.

Разработанная архитектура и методология проектирования, основанные на принципах модульности и визуализации процессов, создают надежную основу не только для текущего симулятора, но и для разработки аналогичных образовательных инструментов в смежных областях. Это позволяет адаптировать систему к новым задачам, обеспечивая высокую степень универсальности и готовность к внедрению инновационных решений, таких как интеграция с виртуальной реальностью или автоматизированные обучающие алгоритмы. Таким образом, проект становится отправной точкой для создания целой экосистемы образовательных симуляторов, способных отвечать современным требованиям к обучению.

3 Реализация симулятора

На этапе реализации симулятора швейной машинки был использован игровой движок Unity, обеспечивающий поддержку 3D-визуализации и интерактивных элементов. В рамках данного раздела были выполнены работы по созданию виртуального пространства, включающего модель швейной машинки и рабочую поверхность, созданные с помощью Autodesk 3ds Max, а также реализованы базовые функции управления, такие как выбор ткани и запуск процесса шитья. Особое внимание уделялось настройке физической модели для достоверного отображения поведения ткани и движения иглы, что позволило заложить основу для дальнейшей разработки функционала симулятора.

3.1 Создание 3d модели швейной машинки

Моделирование швейной машинки в 3ds Max проходило в два основных этапа, каждый из которых включал использование специализированных инструментов и техник для достижения высокого уровня детализации и реализма. На первом этапе акцент был сделан на создании базовой структуры корпуса машинки, что потребовало тщательной подготовки и точной работы с геометрией. Использовался инструмент Line для построения точного 2D-каркаса, который начинался с ручной отрисовки контура корпуса. Этот процесс включал создание множества опорных точек, соединенных сегментами, чтобы точно воспроизвести изгибы и углы корпуса, основываясь на референсных изображениях реальной швейной машинки. После завершения контура каркас подвергался выдавливанию с помощью инструмента Bevel, что позволило преобразовать плоскую форму в трехмерный объект с заданной глубиной. Для придания плавности и эстетичности форм использовался модификатор Smooth, который регулировал количество итераций сглаживания, чтобы избежать резких граней и добиться естественного перехода между поверхностями. Настройки этого модификатора включали контроль уровня сглаживания (обычно от 1 до 3 итераций) и сохранение исходной топологии для предотвращения искажений.

Далее внимание было уделено деталям, таким как ручки, регуляторы и другие мелкие элементы, которые играют ключевую роль в реалистичности модели. Эти компоненты формировались из стандартных примитивов, таких как Box для прямоугольных деталей и Cylinder для круглых элементов, таких как ручки или шпуля. После создания базовых форм применялся инструмент Edit Poly, который позволял вручную редактировать полигоны.

Например, для создания регуляторов шитья (скорости или натяжения ниток) добавлялись небольшие выемки и гравировки, что потребовало детальной работы с полигонами на уровне отдельных граней. Этот подход обеспечил высокую детализацию модели, сохраняя при этом оптимизацию для дальнейшего использования в игровом движке. Кроме того, для повышения визуальной привлекательности применялись текстуры, которые наносились с использованием UV-развертки, что позволило добавить реалистичные материалы, такие как металл для корпуса и пластик для ручек.

Вторая часть работы была посвящена созданию стола, на котором будет размещена швейная машинка, что потребовало отдельного подхода к моделированию. Стол проектировался как функциональный элемент с учетом эргономики и реалистичности. Основная структура стола создавалась с использованием примитивов Box, которые затем редактировались через Edit Poly для формирования ножек и столешницы с учетом углов и изгибов. Внутри стола было предусмотрено специальное пространство для размещения педали с колесом, которые обеспечивают привод в движение иглы машинки. Моделирование педали началось с создания базовой формы из примитива Cylinder, к которой добавлялась детализация через добавление углублений и выступов, имитирующих текстуру реальной педали. Колесо моделировалось отдельно как цилиндрический объект с вырезанными спицами при помощи инструмента Boolean с операндом Subtract, который позволял вычитать одну форму из другой для создания сложных вырезов. Этот инструмент также использовался для формирования внутреннего пространства стола, где педаль и колесо интегрировались в общую конструкцию. Дополнительно применялись операнды Union и Intersection для объединения отдельных элементов стола и проверки их совместимости, что упростило процесс сборки. Например, ножки стола соединялись с основанием через Union, чтобы обеспечить целостность модели.

После завершения моделирования всей конструкции швейная машинка вместе со столом подвергалась финальной проверке на наличие геометрических ошибок, таких как несовпадения полигонов или лишние вершины, с использованием инструмента Check Geometry. Для обеспечения совместимости с игровым движком Unity модель была экспортирована в формате FBX, который поддерживает передачу геометрии, текстур и анимаций. Перед экспортом проводилась оптимизация, включая снижение количества полигонов там, где это не влияло на визуальное качество, и настройка UV-карт для корректного наложения текстур в сцене. Этот процесс завершил этап моделирования, подготовив модель к последующему импорту и анимации в Unity, где она станет частью интерактивной сцены симулятора швейной машинки.

3.2 Разработка окружения симулятора в Unity

Разработка окружения симулятора швейной машинки в Unity началась с создания виртуального пространства, которое должно было отражать реалистичную мастерскую для шитья. Для этого использовались основные панели и инструменты Unity, которые обеспечили гибкость и контроль над процессом.

Основной рабочей областью стала Scene View, где визуализируется и редактируется сцена в реальном времени. Эта панель позволяет перемещаться по сцене с помощью мыши и клавиш (W для перемещения, E для вращения, Q для выбора инструментов), а также масштабировать вид с помощью колесика мыши. Для точного позиционирования объектов использовался инструмент Move Tool (горячая клавиша W), который позволяет перемещать объекты по осям X, Y, Z с помощью цветных стрелок. Rotate Tool (E) и Scale Tool (R) дополняли функционал, позволяя вращать и масштабировать модели, что было особенно важно при выравнивании швейной машинки и мебели.

Панель Hierarchy служила для управления объектами сцены, отображая их в виде иерархической структуры. Здесь добавлялись импортированные модели, такие как швейная машинка, столы и стулья, а также создавались пустые объекты (GameObjects) для организации сцены в группы. Панель Inspector предоставляла доступ к свойствам каждого объекта, включая позицию, ротацию, масштаб и компоненты, такие как Collider или Rigidbody, которые использовались для физического взаимодействия. Например, для швейной машинки добавлялся Box Collider, чтобы обеспечить правильное поведение при столкновении с тканью в симуляции. Панель Project использовалась для управления активами, такими как импортированные модели в формате FBX, текстуры и материалы, которые загружались из 3ds Max.

Импорт моделей окружения начался со швейной машинки, экспортированной из 3ds Max в формате FBX. После импорта в Unity модель появилась в Project, где были проверены настройки импорта (Import Settings), такие как Scale Factor (установлен на 1 для сохранения размеров) и Normals (установлены на Calculate для корректного освещения). Текстуры, связанные с моделью, автоматически применялись через материал, который создавался в разделе Materials. Для устранения возможных ошибок, таких как разрыв UV-карт, использовался инструмент Model Preview в Import Settings, где вручную корректировались параметры. После этого модель перетаскивалась в Scene View, где с помощью Move Tool она размещалась в центре сцены на координатах (0, 0, 0), что стало отправной точкой для дальнейшего размещения.

Следующим шагом стал импорт стола, на котором расположена швейная машинка. Модель стола, также созданная в 3ds Max, включала внутреннее пространство для педали и колеса. После импорта в Unity стол размещался под швейной машинкой с использованием точечного позиционирования в Inspector (например, Y-координата устанавливалась на 0.5 для выравнивания с поверхностью). Для добавления реализма столу применялись материалы с текстурами дерева.

Далее импортировались стулья, созданные как отдельные модели в 3ds Max с использованием примитивов и Edit Poly. После импорта в Unity стулья дублировались (Ctrl+D) и размещались вокруг стола с интервалами примерно в 1.5 метра, чтобы создать рабочую зону. Для каждого стула корректировалась ротация (например, 90 градусов по оси Y), чтобы они были обращены к столу. Текстуры стульев (дерево и ткань сиденья) настраивались через Material, с добавлением вариаций цветов для разнообразия (коричневый, серый).

Полки для ткани представляли собой длинные горизонтальные структуры, импортированные как одна модель с несколькими уровнями. После размещения в Scene View полки располагались вдоль стен сцены и дублируясь с небольшим шагом по оси X. На полках размещались плоские объекты (Plane), представляющие куски ткани, которые окрашивались через материал с текстурами разных цветов и узоров. Для физического взаимодействия ткань получала компонент Rigidbody с ограниченной гравитацией, чтобы она оставалась на полке.

Шкаф с мелкими элементами (книги, горшки с цветами, чашки) моделировался как отдельная сборка в 3ds Max с использованием Boolean и Edit Poly. После импорта в Unity шкаф размещался в углу сцены, а мелкие объекты добавлялись через дублирование простых примитивов (Cube для книг, Sphere для горшков, Cylinder для чашек). Для каждого объекта применялись материалы: книги получали текстуру бумаги, горшки – керамики, а чашки – глянцевой поверхности через настройки Metallic.

Размещение объектов на сцене проводилось с учетом эргономики мастерской. Швейная машинка и стол размещались чуть левее центра, стулья – вокруг, полки – вдоль стен, а шкаф – в углу. Освещение сцены настраивалось через Directional Light, расположенный над сценой с углом 45 градусов. Тени включались через настройку Shadow Casting для всех объектов, что добавило глубину окружению. Камера (Main Camera) настраивалась на обзор всей мастерской с позиции пользователя, что завершило этап разработки окружения.

3.3 Разработка анимаций в Unity через панель Animation

Разработка анимаций для симулятора швейной машинки в Unity через панель Animation стала важным этапом, который позволил придать трехмерным моделям динамику и сделать взаимодействие с программой более реалистичным. Процесс начался с использования встроенных инструментов Unity, где панель Animation предоставила возможность создания и редактирования анимационных клипов путем задания ключевых кадров для каждого движения.

Создание анимаций началось с добавления компонента Animator к объектам сцены, таким как игла, лапка и маховик, что позволило управлять анимационными клипами. Далее в Unity была использована панель Animator, которая предоставляет графический интерфейс для организации анимационных состояний и переходов между ними. В этой панели создавались состояния (States), соответствующие различным движениям объектов, например, «Idle» или «CuttingLoop», как показано на рисунке 3.1.

Связь между состояниями настраивалась с помощью переходов (Transitions), где указывались условия их активации на основе различных параметров. Параметры имели различные типы: bool – для логических переключателей, trigger – для разовых запусков, а также float и int, которые использовались, например, для управления скоростью вращения маховика или выбора режима шитья., что позволяло запускать анимации по пользовательскому вводу через скрипт.

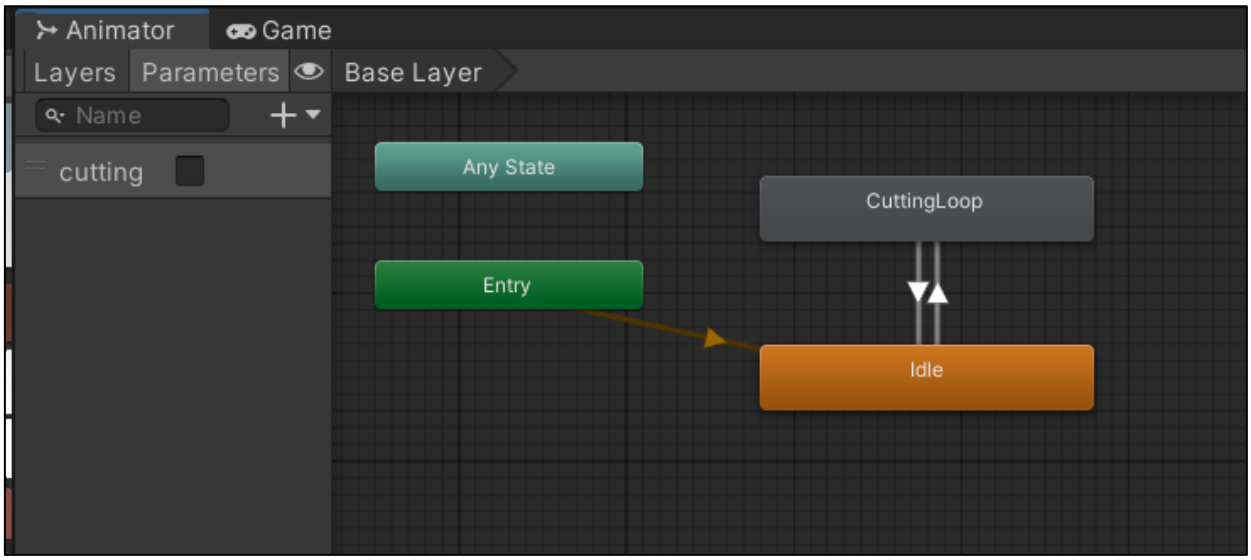

Рисунок 3.1 – Панель Animator

Такой подход обеспечил гибкое управление логикой воспроизведения анимаций и удобную интеграцию с другими компонентами симулятора. Панель Animator позволила визуализировать и отлаживать логику переходов, делая поведение объектов более наглядным и контролируемым.

В панели Animation формировались отдельные клипы для каждого действия, например, для опускания иглы или вращения маховика. Процесс записи активировался кнопкой записи (красный кружок в интерфейсе), после чего задавались позиции и вращения объектов в ключевых моментах. Для имитации движения иглы вниз и вверх определялись начальная позиция на высоте, точка контакта с тканью и возвращение в исходное положение, фиксируемые ключевыми кадрами. Требовалось несколько итераций для достижения естественного ритма, схожего с реальной работой швейной машинки. Аналогичный подход применялся для анимации лапки, где задавалось ее опускание и поднятие, а также для маховика, где фиксировалось его вращение для передачи движения игле. Данную функцию управления анимацией через пользовательский ввод можно увидеть в листинге 3.1.

```
void Update() {

if (Input.GetKeyDown(KeyCode.Space))

 animator.SetTrigger("on"); } }
```

Листинг 3.1 – Функция управления анимацией через пробел

Работа с панелью Animation позволила детально настроить каждое движение, чтобы оно выглядело правдоподобно. Например, для анимации вращения маховика задавались ключевые кадры, определяющие полный оборот, с последующим зацикливанием клипа через настройки Loop Time в инспекторе Unity. Это обеспечило непрерывное вращение, пока пользователь взаимодействует с симулятором. Для опускания лапки задавались два ключевых

кадра – начальная позиция в поднятом состоянии и конечная в опущенном, с плавным переходом между ними.

Предварительный просмотр анимаций в редакторе Unity оказался полезным для выявления и устранения недочетов, таких как резкие изменения позиции или неестественные движения. При анимации маховика изначально наблюдались скачки, которые были устранены путем добавления дополнительных ключевых кадров для сглаживания вращения. Оптимизация анимаций также играла важную роль: количество ключевых кадров сокращалось в тех случаях, где детализация не была критична, например, для простого циклического вращения маховика использовалось всего четыре ключевых кадра на один оборот. Такой подход позволил снизить нагрузку на систему, сохранив визуальное качество, что важно для обеспечения стабильной работы симулятора на различных устройствах.

Интеграция анимаций с логикой сцены осуществлялась через скрипты, связывающие пользовательский ввод с воспроизведением анимационных клипов. Активация анимации происходила при нажатии определенной клавиши, что соответствовало концепции управления процессом шитья. Для более сложных действий, таких как последовательное воспроизведение анимаций, применялись корутины, которые позволяли контролировать временные интервалы между различными этапами. Пример такой реализации показан в листинге 3.2.

```
IEnumerator AnimateThread() {
animator.SetTrigger("start");
yield return new WaitForSeconds(1.5f);
animator.SetTrigger("move"); }
```

Листинг 3.2 – Корутина контролирующая временные интервалы

Полученный результат демонстрирует, как использование панели Animation в Unity позволяет создать динамичную и реалистичную среду для симулятора швейной машинки. Настройка анимаций через ключевые кадры и их интеграция с пользовательским вводом обеспечили интерактивность и удобство использования, что делает программу эффективным инструментом для изучения работы швейной машинки.

3.4 Разработка интерфейса

Основной целью было создание интуитивно понятного и функционального интерфейса, который позволил бы управлять процессами симуляции, выбирать необходимые элементы и получать визуальную обратную связь. Работа велась с использованием встроенных инструментов Unity UI, таких как Canvas, Button, TextMeshPro и Image, что позволило реализовать необходимые элементы управления и отображения информации.

Процесс начался с проектирования структуры интерфейса, где учитывались основные потребности пользователя: доступ к настройкам симулятора, выбор инструментов и материалов, а также управление процессом шитья. Для

реализации использовалась система Canvas, настроенная в режиме Scale With Screen Size, чтобы интерфейс корректно отображался на устройствах с разным разрешением экрана. Элементы интерфейса, такие как кнопки, текстовые поля и изображения, создавались и настраивались в иерархии сцены, а их функциональность обеспечивалась через скрипты на языке C#. Например, для обработки нажатий на кнопки применялись методы, подобные тем, что были использованы в файле кода для управления анимациями, показанной в листинге 3.1, где метод Update() реагировал на ввод пользователя.

Особое внимание уделялось визуальному оформлению интерфейса. Для этого использовались изображения, предоставленные в качестве ресурсов, которые были импортированы в проект и применены к компонентам Image в Unity. Эти изображения стали основой для оформления кнопок, фоновых панелей и других элементов интерфейса, обеспечивая единый стиль, соответствующий тематике швейной машинки. Настройка цветов, шрифтов и размеров текста проводилась с использованием TextMeshPro, что позволило добиться читаемости текста на различных фонах. Прозрачность фоновых панелей регулировалась через параметры альфа-канала, чтобы элементы интерфейса не отвлекали от основной сцены симулятора.

Функциональность интерфейса обеспечивалась через скрипты, которые связывали элементы UI с логикой симулятора. Например, для активации определенных действий, таких как запуск процесса шитья, использовались кнопки, которые при нажатии вызывали методы, аналогичные тем, что применялись для управления анимациями. В данном случае корутины из листинга 3.2 могли быть адаптированы для последовательного отображения элементов интерфейса, например, для постепенного открытия панелей или активации подсказок. Это позволило сделать взаимодействие с интерфейсом более плавным и предсказуемым для пользователя.

Тестирование интерфейса проводилось на разных этапах разработки, чтобы убедиться в его удобстве и работоспособности. Проверялась отзывчивость кнопок, читаемость текста и корректность отображения изображений, особенно тех, что были предоставлены в качестве ресурсов. При необходимости вносились корректировки, такие как изменение расположения элементов или увеличение размеров кнопок для улучшения их доступности.

3.4.1 Разработка главного меню

Разработка главного меню для симулятора швейной машинки в Unity стала начальным этапом создания пользовательского интерфейса, направленным на обеспечение удобного доступа к основным функциям программы. Главное меню, расположенное в отдельной сцене «Menu 1», предназначено для навигации по симулятору, включая запуск основной сцены, настройку параметров и выход из приложения. Для реализации использовался Canvas, настроенный в режиме Screen Space - Overlay, что обеспечило стабильное отображение меню поверх сцены независимо от камеры.

Элементы интерфейса, такие как кнопки и заголовки, создавались с использованием компонентов Button и Image, которые интегрировались в иерархию сцены через редактор Unity. Проектирование меню началось с определения ключевых функций: запуск симуляции, переход к меню настроек и завершение работы приложения. Центральным элементом интерфейса стал заголовок «Симулятор швейной машинки», реализованный с помощью компонента Image, в который был вставлен спрайт с заранее подготовленным текстом. Графика заголовка была создана с использованием онлайн-сервиса textdrom.com [4], что позволило придать ему выразительный внешний вид. Благодаря этому заголовок четко выделялся на фоне, оформленном в стиле цветочных узоров, как показано на рисунке 3.2.

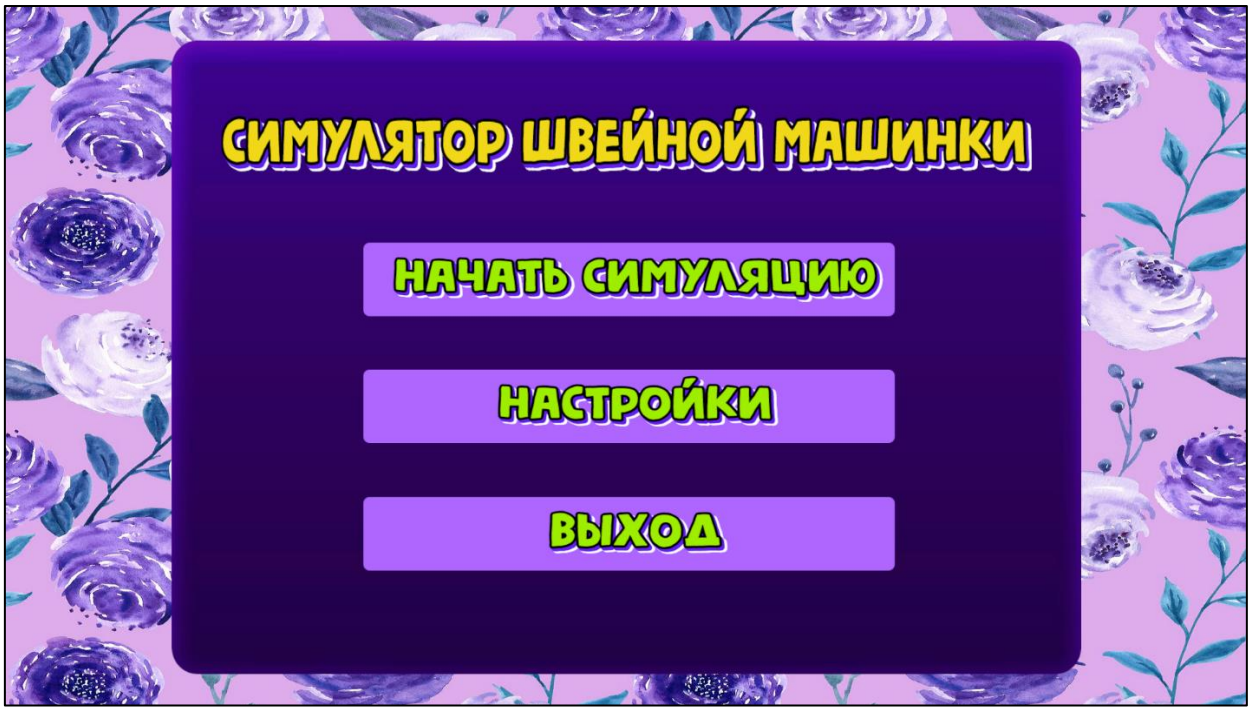

Рисунок 3.2 – Главное меню

Кнопки «Начать симуляцию», «Настройки» и «Выход» размещались вертикально с равными интервалами, а их дизайн основывался на импортированных изображениях, обеспечивая единый стиль, соответствующий тематике швейной машинки. Размеры кнопок подбирались с учетом эргономики, а прозрачность фона регулировалась через альфа-канал для предотвращения наложения на сцену.

Функциональность меню обеспечивалась через скрипты, привязанные к компонентам Button. Каждая кнопка в меню имела компонент OnClick, в который через инспектор Unity добавлялась ссылка на соответствующий метод из скрипта. Для обработки нажатий использовались методы, реализованные в классе StartGame, показанном в листинге 3.3. Этот скрипт подключался к сцене меню и выполнял ключевые действия по навигации между сценами приложения.

Метод StartGames() используется для запуска основной симуляции – при вызове он очищает все сохраненные пользовательские данные через PlayerPrefs.DeleteAll() и загружает сцену с именем, заданным в переменной

sceneToLoad. Метод Menu() выполняет возврат на стартовую сцену «Menu 1», позволяя пользователю вернуться к главному меню. Метод ContinueGames() позволяет напрямую перейти к основной сцене без удаления сохраненных данных, что может быть полезно при реализации функции продолжения сессии.

```
public class StartGame : MonoBehaviour
{
    public string sceneToLoad = "SampleScene";
    public void StartGames()

    {
        PlayerPrefs.DeleteAll();
        SceneManager.LoadScene(sceneToLoad);
    }

    public void Menu()
    {
        SceneManager.LoadScene("Menu 1");
    }

    public void ContinueGames()
    {
        SceneManager.LoadScene("SampleScene");
    }
}
```

Листинг 3.3 – Класс StartGame

Кнопка «Выход» вызывала метод ExitGames(), завершающий выполнение приложения через команду Application.Quit(). Этот метод позволяет корректно завершить работу программы при ее запуске в сборке (Build), например, в формате .exe. В редакторе Unity при тестировании метод не завершает выполнение, что является нормальным поведением и предусмотрено системой. Такой подход позволяет реализовать стандартную функциональность выхода, привычную пользователям. Данный метод продемонстрирован в листинге 3.4.

```
public class ExitGame : MonoBehaviour
{
    public void ExitGames()
    {
        Application.Quit();
    }
}
```

Листинг 3.4 – Класс ExitGames

Настройки реализовывались через вызов отдельной сцены или панели с компонентом Settings, где пользователь мог управлять громкостью и полноэкранным режимом. Более подробно процесс создания меню настроек описан в подразделе 3.4.2 «Разработка меню настроек».

В результате было создано главное меню, которое обеспечивает удобный старт работы с симулятором, интегрируя визуальные элементы и функциональность в единый интерфейс, соответствующий образовательной цели программы.

3.4.2 Разработка меню настроек

Меню настроек в симуляторе швейной машинки предназначено для управления основными параметрами приложения – звуковой громкостью и режимом отображения (полноэкранный или оконный). Оно реализовано в сцене Menu 1 с помощью интерфейса Unity UI и встроено в общий пользовательский интерфейс через компонент Canvas с режимом Screen Space - Overlay.

Структура меню включает слайдер для регулировки громкости (Slider) и кнопку-переключатель полноэкранного режима (Button). Для управления аудио использовался компонент AudioMixer, что позволило гибко регулировать звук без необходимости напрямую изменять параметры аудиофайлов. Пример соответствующего кода представлен в листинге 3.5.

```
public void AudioVolume(float sliderValue)
{
    am.SetFloat("masterVolume", sliderValue);
    PlayerPrefs.SetFloat("volume", sliderValue);
    Debug.Log("Сохранили громкость: " + sliderValue);
}
```

Листинг 3.5 – Класс AudioVolume

Значение громкости сохраняется через PlayerPrefs, чтобы настройки пользователя автоматически применялись при следующем запуске приложения. Это обеспечивает сохранение пользовательского опыта и избавляет от необходимости повторной настройки при каждом запуске. При старте сцены скрипт Settings проверяет наличие сохраненных данных и загружает их, устанавливая значение слайдера и параметр в AudioMixer. В случае отсутствия сохраненного значения устанавливается громкость по умолчанию, что исключает ошибки и гарантирует корректное функционирование звука.

Для управления полноэкранным режимом реализована функция FullScreenToggle(), которая переключает текущий режим отображения. Как показано в листинге 3.6.

```
public void FullScreenToggle()
{
    isFullScreen = !isFullScreen;
    Screen.fullScreen = isFullScreen;
}
```

Листинг 3.6 – Управление полноэкранным режимом

Пользовательский интерфейс меню разработан с учетом простоты взаимодействия: слайдер и кнопка расположены вертикально, с четкими подписями и

иконками, обеспечивающими однозначное понимание их назначения. Все элементы имеют достаточные размеры и интервалы, что облегчает работу с интерфейсом и снижает вероятность ошибочного нажатия. Расположение и визуальное оформление элементов подобраны таким образом, чтобы обеспечить удобное восприятие и быструю навигацию в рамках фиксированного разрешения сцены. На рисунке 3.3 представлена визуализация меню настроек.

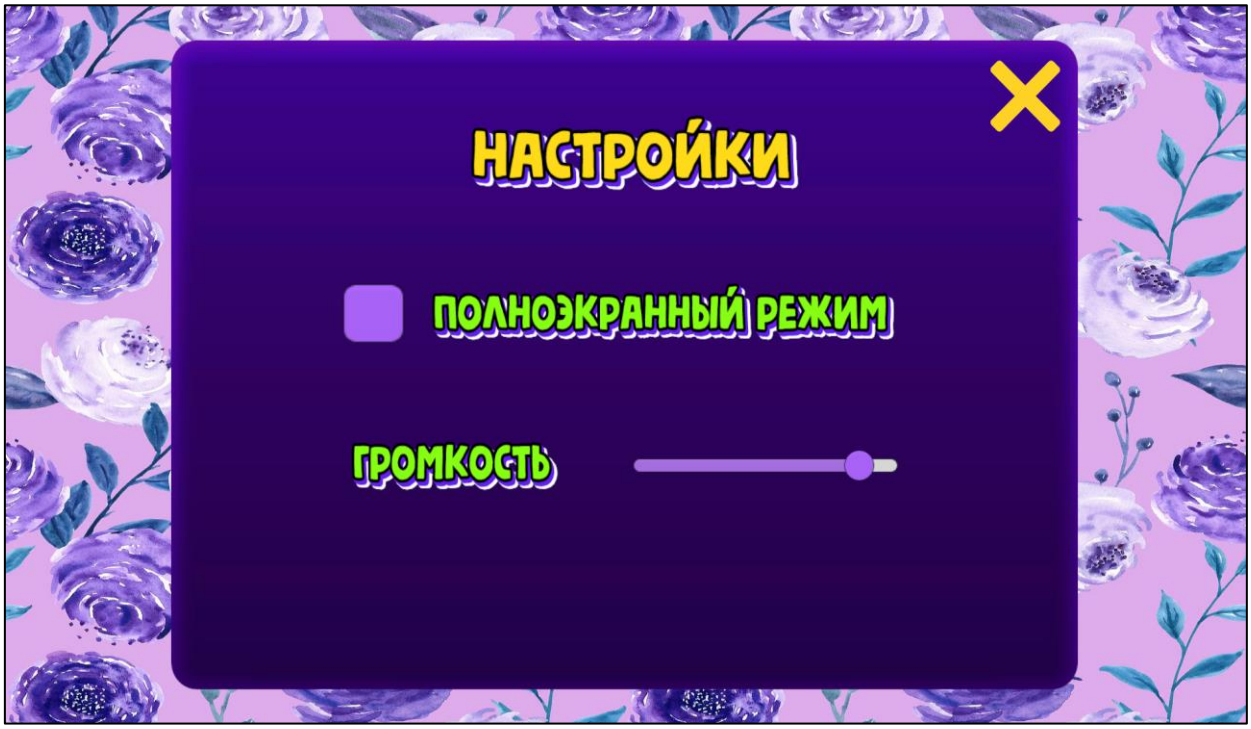

Рисунок 3.3 – Меню настроек

Тестирование функциональности меню показало стабильную работу всех элементов: громкость изменяется в реальном времени, параметры сохраняются между сессиями, а переключение между оконным и полноэкранным режимами не вызывает ошибок. Таким образом, меню настроек стало важным элементом, обеспечивающим индивидуальную настройку симулятора под предпочтения пользователя.

3.4.3 Разработка инвентаря

Разработка инвентаря в симуляторе швейной машинки стала важным этапом создания интерфейса, обеспечивающего управление тканями и трафаретами. Инвентарь реализован в сцене «SampleScene» как панель на Canvas в режиме Screen Space - Overlay, что гарантирует стабильное отображение поверх сцены. Элементы панели – слоты и кнопки – создавались с помощью компонентов Image и Button, размещенных в иерархии сцены.

Инвентарь включает два слота: для ткани и трафарета. Каждый слот представляет собой объект с изображением предмета и кнопкой для взаимодействия. Элементы размещались в нижней части экрана с учетом удобства

доступа. Управление открытием и закрытием панели реализовано через скрипт OpenClosePanel, пример которого приведен в листинге 3.7.

```
public void TogglePanel()
{
    if (!flag) {
        targetPanelPosition = panelStartPosition;
        targetButtonPosition = buttonStartPosition;
        buttonImage.sprite = closedSprite;
        flag = true;}
    else{
        targetPanelPosition = new Vector2(panelStartPosition.x,
-Screen.height / 1.55f);
        targetButtonPosition = new Vector2(buttonStartPosi-
tion.x, -Screen.height / 2.2f);
        buttonImage.sprite = openedSprite;
        flag = false;
    }}
```

Листинг 3.7 – Скрипт OpenClosePanel

Функциональность инвентаря реализовывалась через несколько скриптов. Класс Inventory управлял состоянием слотов, используя массивы isFull и itemTags для отслеживания заполненности и типа предметов, данную функцию управления состоянием слотов можно увидеть в листинге 3.8. Добавление предметов в инвентарь осуществлялось через скрипт Pickup, где метод OnMouseDown() проверял доступность слота и добавлял ткань с тегом «Cloth», одновременно изменяя материалы объектов сцены. Для трафаретов использовался скрипт Slot, где метод OnSlotButtonClicked() активировал объекты, связанные с выбранным трафаретом, и запускал анимации через вызов метода PlayAnim(). Например, выбор ткани из первого слота запускал анимацию иглы, как это реализовано в SimpleNeedleAnimation. Второй слот, предназначенный для трафаретов, активировал соответствующие объекты сцены, такие как object1 и object2, после чего слот очищался через вызов DropItem().

```
public void AddSpriteToSlot(Sprite sprite)
{
    int targetSlot = 1;
    if (targetSlot < slots.Length){
        if (!isFull[targetSlot]){
            isFull[targetSlot] = true;
            itemTags[targetSlot] = sprite.name;
            Image slotImage = slots[targetSlot].GetComponent<Im-
age>();
            if (slotImage != null)
            {
                slotImage.sprite = sprite;
                slotImage.enabled = true;
}}}}
```

Листинг 3.8 – Скрипт Slot

Инвентарь был интегрирован с другими элементами симулятора. При выборе трафарета из второго слота его спрайт передавался в ScissorsController через метод SetPanelSprite() скрипта MoveStencil.cs, что позволяло использовать его при вырезании ткани. Для наглядности применялись изображения, отображаемые в слотах. Инвентарь тестировался на корректность добавления, удаления и взаимодействия с другими объектами сцены. При необходимости настраивались размеры и положение слотов. В итоге панель стала удобным инструментом для управления предметами во время симуляции.

3.4.4 Разработка панели элементов установки

Панель реализована в сцене «SampleScene» как интерактивный элемент, отображающий описания компонентов, таких как игла, лапка и маховик, и позволяющий активировать соответствующие действия. Для создания панели использовался Canvas, настроенный в режиме Screen Space - Overlay, что обеспечило стабильное отображение поверх сцены. Элементы интерфейса, такие как кнопки и текстовое поле, создавались с использованием компонентов Button и TMP_Text, интегрированных в иерархию сцены.

Проектирование панели началось с определения ее структуры: набор кнопок, каждая из которых связана с конкретной деталью швейной машины, и текстовое поле для отображения описания. Кнопки размещались вертикально в левой части экрана, как показано на рисунке 3.4, где использовались иконки для обозначения элементов, что соответствовало тематике симулятора.

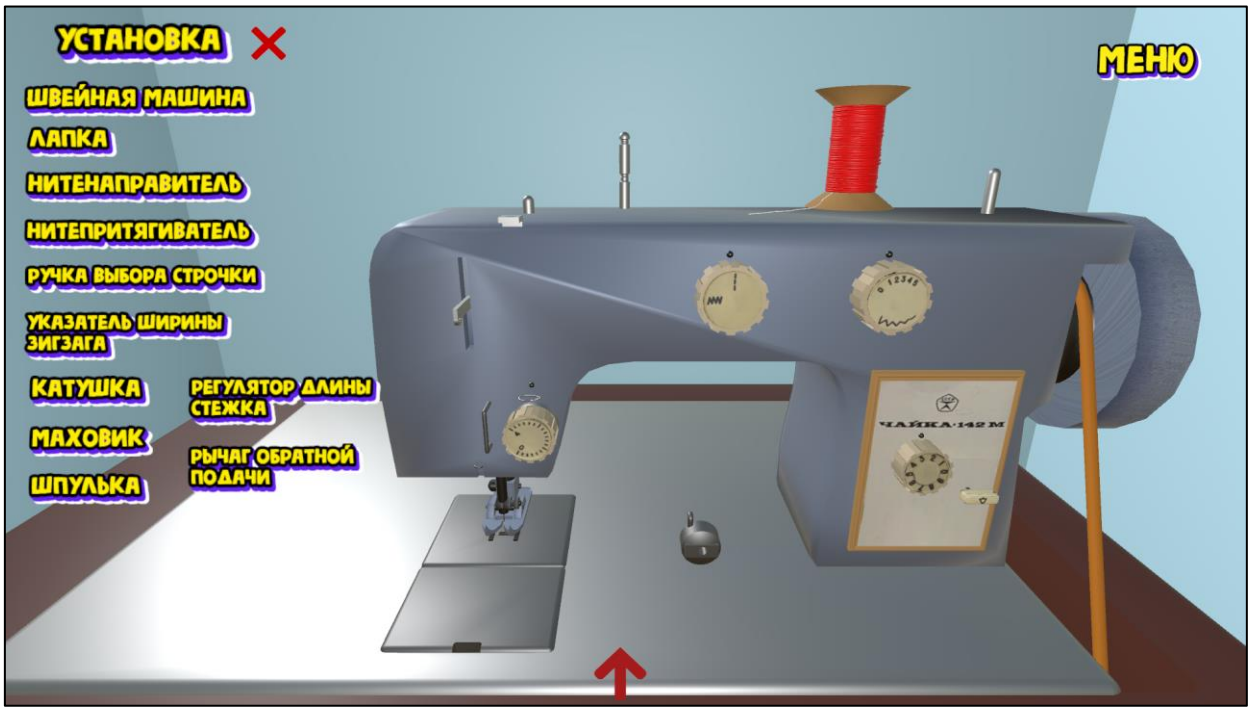

Рисунок 3.4 – Панель «Установка»

Текстовое поле, управляемое компонентом TMP_Text, размещалось рядом с кнопками и обновлялось при нажатии, обеспечивая динамическую

обратную связь. Панель управлялась скриптом ShowYstanPanel, где данную функцию открытия и закрытия панели можно увидеть в листинге 3.9.

```
public void Open()
{   btn1.SetActive(true);
    btn2.SetActive(true);
    btn3.SetActive(true);
    btn4.SetActive(true);
    btn5.SetActive(true);
    btn6.SetActive(true);
    btn7.SetActive(true);
    btn8.SetActive(true);
    btn9.SetActive(true);
    btn10.SetActive(true);
    btn11.SetActive(true);
    btnclose.SetActive(true);
}
```

Листинг 3.9 – Скрипт ShowYstanPanel

Функциональность панели обеспечивалась через скрипт ShowYstanPanel, где методы, такие как TextMachine() и TextLapka(), обновляли текстовое поле с описаниями , как показано в листинге 3.10. Например, нажатие на кнопку «Игла» вызывало метод TextMachine(), отображая информацию о швейной машине, а кнопка «Лапка» – метод TextLapka(), предоставляя данные о лапке. Интеграция с анимациями осуществлялась через скрипты, такие как SimpleNeedleAnimation и lapkaAnim, где активация кнопок могла инициировать соответствующие действия, например, движение иглы или опускание лапки. Это обеспечивало связку интерфейса с механикой симулятора, делая панель интерактивной.

```
public void TextMachine()
{
    message.text = "Швейная машина – техническое устройство для
соединения и отделки материалов методом шитья";
}

public void TextLapka()
{
    message.text = "Лапка нажимная – устройство, которое удержи-
вает ткань";
}
```

Листинг 3.10 – Обновление текстовых полей через методы

Тестирование панели проводилось для проверки корректности отображения описаний и отзывчивости кнопок. Проверялась читаемость текста в текстовом поле, и соответствие действий кнопок их назначению. При необходимости корректировались размеры кнопок и их расположение, чтобы избежать перекрытия с другими элементами интерфейса. В результате панель

элементов установки стала удобным инструментом, предоставляющим пользователю информацию о деталях швейной машины и интегрирующим управление с процессом симуляции.

3.4.5 Разработка панели выбора цвета ниток

Панель выбора цвета ниток была реализована в сцене SampleScene с использованием компонента Canvas. Основной задачей панели является предоставление пользователю возможности выбирать цвет нитки, который затем применяется ко всем 3D-моделям элементов швейной машинки, таким как нитка, шпулька и строчка. Это позволяет наглядно видеть результат изменения цвета в реальном времени и усиливает эффект персонализации в ходе симуляции.

Каждый элемент выбора цвета представляет собой отдельную кнопку (Button), окрашенную в соответствующий цвет с помощью компонента Image. Кнопки выстраивались в сетку с равными отступами, что обеспечило удобную навигацию и четкую визуальную идентификацию цвета. Для повышения читаемости каждая кнопка имела четкие границы и легкий визуальный эффект при наведении курсора. В качестве цвета ниток использовались как стандартные однотонные цвета (черный, белый, красный и т.д.), так и кастомные оттенки, соответствующие цветовой палитре интерфейса.

Основная логика обработки клика по кнопке реализована в скрипте treadColorChange, прикрепленном к панели. При нажатии на кнопку вызывается метод ChangeColorFromButton(), который получает цвет кнопки через компонент Image и применяет его к всем объектам, хранящимся в массиве objectsToChange. Фрагмент соответствующего кода приведен в листинге 3.11:

```
public void ChangeColorFromButton(GameObject button)
{
    Debug.Log("Button clicked: " + button.name);
    Button clickedButton = button.GetComponent<Button>();
    if (clickedButton != null)
    {
        Image buttonImage = clickedButton.GetComponent<Image>();
        if (buttonImage != null)
        {
            Color newColor = buttonImage.color;
            ApplyColorToObjects(newColor);
}}}
```

Листинг 3.11 – Метод ChangeColorFromButton()

Для применения цвета к нужным объектам используется вспомогательный метод ApplyColorToObjects(), показанный в листинге 3.12, который перебирает массив Renderer и устанавливает заданный цвет. Метод проверяет наличие каждого объекта в массиве и, если он не равен null, изменяет цвет материала через свойство sharedMaterial.color. Такой подход позволяет одновременно обновить внешний вид нескольких элементов сцены, включая нити, шпульку и строчку. Использование sharedMaterial обеспечивает изменение материала на уровне

префаба, что может быть полезно при необходимости визуального обновления сразу всех связанных объектов. При этом структура кода остается простой и легко масштабируемой при расширении функциональности.

```
private void ApplyColorToObjects(Color color)
{
    foreach (Renderer obj in objectsToChange)
    {
        if (obj != null)
        {
            obj.sharedMaterial.color = color;
        }
    }
}
```

Листинг 3.12 – Метод ApplyColorToObjects()

Панель выбора цвета ниток была спроектирована с прицелом на простоту расширения: добавление новых цветов или текстур не требует изменений в кодовой части, достаточно создать новую кнопку с нужным оттенком в редакторе Unity и назначить ей метод ChangeColorFromButton() через инспектор.

Для повышения информативности и удобства взаимодействия с панелью планируется внедрение визуальной индикации текущего выбранного цвета, например, отображение рамки вокруг активной кнопки или вывод цвета в отдельном индикаторе. Это позволит пользователю лучше ориентироваться при выборе и переключении между различными оттенками.

В дальнейшем возможно расширение функционала панели – добавление поддержки пользовательских палитр, сохранение последнего выбранного цвета между сессиями с использованием PlayerPrefs, а также фильтрация доступных цветов в зависимости от типа выбранной ткани или режима симуляции. Эти доработки обеспечат более персонализированный и адаптивный пользовательский опыт.

С технической точки зрения панель выбора цвета является легким и стабильным элементом интерфейса: взаимодействие осуществляется исключительно по событию нажатия на кнопку, что минимизирует нагрузку на систему и исключает лишние вычисления в процессе работы. Скрипт treadColorChange легко масштабируется, так как использует универсальный массив Renderer[], благодаря чему может быть применен и для других объектов, нуждающихся в смене цвета – например, для декорирования элементов окружения или интерфейса.

Визуальное оформление панели соответствует общей стилистике симулятора: каждая кнопка имеет скругленные края и контур. Это делает взаимодействие пользователя с интерфейсом более приятным и интуитивно понятным. В целом, реализованная панель значительно повышает уровень интерактивности симулятора и позволяет пользователю настраивать внешний вид изделий, что особенно важно в образовательном контексте, ориентированном на освоение основ швейного дела.

3.5 Разработка обводки и обрезки ткани

Функциональность обводки и обрезки ткани играет важную роль в симуляции процесса кройки, приближая взаимодействие пользователя к реальным действиям в швейном деле. Данный механизм был реализован в сцене SampleScene с использованием объектов, имитирующих ножницы и ткань, а также соответствующих скриптов, обрабатывающих пересечения и действия по обрезке.

Процесс начинается с выбора пользователем трафарета, который накладывается на ткань. Далее активируется инструмент обрезки — ножницы, реализованные как 3D-модель с привязкой к курсору или движению мыши. Обводка осуществляется путём отслеживания координат касания вдоль границ трафарета. При подтверждении действия запускается процесс «вырезания» — на практике это реализуется через скрытие или отключение определённых частей объекта ткани, которые находятся вне границ трафарета.

Пример кода, реализующего проверку касания ножниц с тканью и активацию действия, приведён в листинге 3.13.

```
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Tkan"))
    {
        Debug.Log("Касание ткани зафиксировано");
        // Вызов метода обрезки
        CutOutline();
    }
}
```

Листинг 3.13 – Фрагмент кода обработки касания ткани

Фрагмент кода с использованием метода OnTriggerEnter позволяет отслеживать момент соприкосновения ножниц с тканью за счёт коллайдеров. Объект ткани имеет тег «Tkan», что даёт возможность быстро идентифицировать его в логике взаимодействия. После срабатывания триггера вызывается метод CutOutline(), внутри которого реализуется логика визуального отделения нужного участка. Такой способ упрощает реализацию взаимодействия, делая код компактным, понятным и легко масштабируемым под другие инструменты, например, виртуальный резак или лекало.

Метод CutOutline() обрабатывает геометрию ткани или визуально изменяет область по контуру трафарета, например, путём замены материала, сокрытия части меша или включения заранее подготовленного вырезанного объекта. Такой подход позволил реализовать обрезку без сложной физической деформации объекта, сохранив при этом реализм восприятия.

Также предусмотрены визуальные и звуковые эффекты — щелчок ножниц и изменение текстуры ткани — что усиливает эффект взаимодействия. Реализация обрезки ткани не только добавляет интерактивности, но и способствует обучению базовым операциям, применяемым в швейном производстве, что особенно ценно в образовательной среде.

3.6 Выводы по разделу

Разработка анимаций и интерфейса для симулятора швейной машинки в Unity стала важным этапом, который позволил реализовать функциональный и интерактивный продукт, направленный на обучение основам шитья. В ходе работы над разделом 3.3 были созданы анимации ключевых элементов, таких как игла, лапка и маховик, с использованием панели Animation. Это обеспечило реалистичное воспроизведение механики швейной машины, где движения деталей были настроены через ключевые кадры, а их запуск осуществлялся через пользовательский ввод, что сделало симулятор более динамичным и понятным для пользователя.

Раздел 3.4 был посвящен созданию интерфейса, включающего главное меню, инвентарь и панель элементов установки. Главное меню, реализованное в отдельной сцене, предоставило удобную навигацию для запуска симуляции, настройки параметров и выхода из приложения. Инвентарь стал инструментом для управления предметами, такими как ткани и трафареты, обеспечивая их выбор и интеграцию с другими механиками симулятора. Панель элементов установки позволила пользователю получать информацию о деталях швейной машины и взаимодействовать с ними, связывая интерфейс с анимациями. Все элементы интерфейса были оформлены с использованием предоставленных изображений, что придало симулятору единый визуальный стиль, соответствующий тематике шитья.

В результате проведенной работы был создан симулятор, который сочетает в себе функциональность и удобство использования. Анимации и интерфейс, разработанные в рамках данного раздела, обеспечили интуитивное взаимодействие с программой, что делает ее эффективным инструментом для обучения. Оптимизация анимаций и тщательное тестирование интерфейса позволили добиться стабильной работы симулятора, сохраняя при этом визуальную привлекательность и реалистичность процессов. Этот опыт показал важность комплексного подхода к разработке, где каждая деталь, от анимации до интерфейса, играет ключевую роль в создании целостного продукта.

4 Тестирование симулятора

4.1 Цели и задачи тестирования

Целью тестирования симулятора швейной машинки является проверка его работоспособности, соответствия заданным функциональным требованиям и устойчивости к различным сценариям взаимодействия с пользователем. Тестирование позволяет убедиться в корректности выполнения всех предусмотренных функций, а также выявить возможные ошибки или недоработки, которые могут повлиять на пользовательский опыт.

Основные задачи, решаемые в процессе тестирования, включают следующие ключевые пункты:

– проверка реализации всех ключевых функций симулятора: выбор ткани, настройка параметров, обводка трафарета, обрезка, сшивание, взаимодействие с элементами интерфейса;

– оценка стабильности работы симулятора при длительном использовании и в условиях высокой нагрузки;

– проверка реакции системы на внештатные или ошибочные действия пользователя (например, попытка запустить шитье без выбора ткани);

– оценка корректности визуализации и анимации элементов, в том числе иглы, лапки, трафарета и прочих компонентов симулятора;

– проверка работоспособности обучающих панелей, всплывающих подсказок и навигации по интерфейсу;

– установление соответствия между ожидаемым поведением системы и фактическими результатами при выполнении контрольных сценариев;

– подготовка рекомендаций для устранения выявленных недостатков и повышения качества программного средства.

Реализация этих задач позволяет обеспечить надежность, удобство и функциональную полноту симулятора, что особенно важно в контексте его использования в образовательных целях.

4.2 Методика тестирования

Тестирование симулятора швейной машинки проводилось с использованием нескольких подходов, направленных на комплексную проверку всех аспектов работы программного обеспечения. Для достижения достоверных и объективных результатов была выбрана комбинированная методика, включающая как ручное, так и автоматизированное тестирование.

Основные виды тестирования, примененные в рамках данного проекта:

– функциональное тестирование. Проверялись все основные функции симулятора: запуск и завершение приложения, перемещение пользователя в 3D-пространстве, выбор ткани и трафарета, выполнение обводки и обрезки, шитье, взаимодействие с интерфейсом и обучающими панелями. Для каждой функции составлялись отдельные сценарии (use cases) с ожидаемыми результатами;

– тестирование пользовательского интерфейса. Оценивалась корректность отображения и отклика элементов интерфейса: меню, кнопок, панелей инвентаря и обучения. Проверялась адаптивность интерфейса к различным разрешениям экрана и удобство взаимодействия;

– тестирование на устойчивость к ошибкам. Имитация неправильных действий пользователя (например, попытка выполнить операцию без выбора нужного элемента) позволила проверить, как система реагирует на некорректные сценарии и исключительные ситуации. Оценивается наличие защитных механизмов и соответствующих сообщений об ошибках;

– тестирование производительности. Проводилась оценка скорости отклика симулятора при высокой активности пользователя и длительной работе. Анализировалось потребление системных ресурсов, частота кадров и стабильность работы при различных нагрузках;

– интерактивное пользовательское тестирование. На финальном этапе симулятор проходил проверку с участием стороннего пользователя, не участвовавшего в разработке. Это позволяло выявить потенциальные проблемы удобства и доступности, которые могли быть упущены при внутреннем тестировании.

Тестирование проводилось по заранее подготовленным чек-листам, включающим набор шагов и критериев оценки. Каждый тест сопровождался фиксацией результата: «пройдено», «не пройдено» или «ошибка», а также записью комментариев при возникновении отклонений от ожидаемого поведения.

Такой подход обеспечил всестороннюю проверку работы симулятора и создал основу для анализа его качества и стабильности, а также последующей доработки и оптимизации.

4.3 Подготовка контрольного примера

Для проведения тестирования был разработан контрольный пример, охватывающий весь основной функционал симулятора швейной машинки. Контрольный пример представляет собой заранее определенный сценарий использования, позволяющий оценить работоспособность ключевых компонентов приложения в условиях, приближенных к реальной эксплуатации.

Условия тестирования:

– аппаратная платформа: персональный компьютер с операционной системой Windows 10, процессором Intel Core i5, 8 ГБ оперативной памяти, видеокартой уровня NVIDIA GTX 1050 и установленным игровым движком Unity (режим запуска в редакторе и в собранном .exe-файле);

– разрешение экрана: 1920×1080 пикселей (Full HD);

– входные устройства: клавиатура и мышь;

– финальная версия, собранная после завершения этапа реализации.

Описание контрольного сценария:

– запуск симулятора через главное меню;

– перемещение пользователя в пределах симуляционного пространства;

– выбор ткани из инвентаря и ее размещение на рабочей поверхности;

– выбор трафарета и выполнение обводки по контуру;

– обрезка ткани по трафарету;

– перенос вырезанного фрагмента к швейной машинке;

– сшивание подготовленных фрагментов ткани;

– просмотр и взаимодействие с обучающей панелью (информация о игле, лапке, маховике и т.д.);

– завершение работы с приложением и возврат в главное меню.

Цель контрольного примера – обеспечить пошаговую проверку всех основных функций симулятора в логической последовательности, соответствующей предполагаемому учебному сценарию. Это позволяет выявить ошибки, возникающие как на отдельных этапах, так и в процессе перехода между ними.

Контрольный пример служит универсальной основой для сопоставления ожидаемых и фактических результатов, а также позволяет проводить регрессионное тестирование при внесении изменений в приложение.

4.4 Проведение тестирования основных функций

Проведение тестирования симулятора швейной машинки началось с оценки базовой функциональности, определенной на этапе проектирования. Проверка осуществлялась в логической последовательности, имитирующей действия реального пользователя – от запуска приложения до завершения сеанса работы. Это позволило не только оценить отдельные функции по отдельности, но и проверить связность интерфейса, переходов между этапами и общую устойчивость системы.

Одной из наиболее критичных функций является процесс сшивания ткани. При тестировании пользователь помещает вырезанный фрагмент под лапку швейной машинки и запускает шитье. Важно было проверить анимацию иглы и визуализацию создаваемого шва. Все компоненты работали синхронно: движение иглы и маховика совпадало с визуальным изменением ткани, а шов формировался в точке соприкосновения.

Тестированию также подверглась обучающая панель, содержащая описание компонентов швейной машинки. При нажатии на соответствующие кнопки, на экране появлялась информация об игле, лапке, маховике и других элементах. В процессе теста проверялись полнота текста, читаемость, своевременность появления и скрытия панели, а также возможность быстро вернуться к предыдущему этапу.

Финальный этап тестирования касался корректного завершения работы симулятора. Были проверены кнопки выхода и возврата в главное меню. Завершение текущего сеанса происходило без сбоев, все пользовательские

действия сбрасывались, а сохраненные данные корректно очищались при необходимости. Возврат в главное меню происходил с соблюдением всех анимаций и без заметных задержек, что подтверждало правильность реализации логики переходов между сценами.

В результате проведенных испытаний стало ясно, что основные функции симулятора работают стабильно, интерфейс интуитивно понятен, а визуальные и анимационные элементы корректно реагируют на действия пользователя. Все этапы взаимодействия протестированы на соответствие требованиям, описанным в проектной документации. Выявленные в процессе тестирования незначительные замечания, касающиеся, например, выравнивания элементов интерфейса, были оперативно устранены. По итогам можно сделать вывод о высокой степени готовности симулятора к эксплуатации и его соответствию учебным целям.

4.5 Обработка ошибок и внештатные ситуации

Одним из ключевых этапов тестирования симулятора швейной машинки стала проверка его поведения в условиях внештатных ситуаций и некорректных действий пользователя. Целью данного этапа было убедиться, что приложение способно адекватно реагировать на ошибки, не допускает критических сбоев и предоставляет пользователю понятные уведомления и подсказки.

Тестирование проводилось по сценарию, в рамках которого пользователь сознательно выполнял действия, выходящие за рамки стандартного использования. Среди таких действий:

– попытка начать процесс шитья без выбора ткани;

– попытка обвести трафарет без размещения на ткани;

– удаление предметов из инвентаря во время активной операции;

– повторное нажатие на кнопки запуска анимаций, не дождавшись завершения предыдущих;

– выход из симуляции в момент активного процесса (например, при работе машинки или во время обрезки ткани).

Результаты показали, что симулятор устойчив к большинству таких сценариев. В случае отсутствия выбранного материала при попытке сшивания система не выполняет действие.

При попытке начать обводку трафарета, не выбрав его, операция блокируется, а на экране появляется соответствующее предупреждение. Аналогично реализована защита от запуска обрезки без завершенной обводки – система проверяет наличие активного контура перед выполнением действия.

Были также проверены сценарии с быстрым переключением состояний, например, многократное быстрое нажатие на кнопки панели интерфейса. Симулятор корректно обрабатывает такие действия: повторные команды игнорируются до завершения текущей, что исключает конфликты между анимациями и логикой.

Особое внимание уделялось проверке сохранности состояния при аварийных выходах. Если пользователь покидает симуляцию в процессе выполнения действия, при повторном запуске приложение корректно возвращается

в начальное состояние, сбрасывая неоконченные операции. Это особенно важно для учебного применения симулятора, так как позволяет избежать не консистентных данных или зависаний.

Также была проверена устойчивость к отсутствию интернет-соединения и проблемам с графическим рендерингом (например, при понижении производительности ПК). В данных случаях приложение продолжает работу в офлайн-режиме, сохраняя базовую функциональность, а визуальные эффекты автоматически упрощаются для повышения производительности.

Таким образом, проведенное тестирование внештатных ситуаций показало, что симулятор обладает высоким уровнем отказоустойчивости. В большинстве случаев система либо корректно предотвращает ошибочное действие, либо уведомляет пользователя о невозможности его выполнения. Это значительно повышает общее качество пользовательского опыта и делает симулятор пригодным для использования в образовательных учреждениях, где важна не только функциональность, но и безопасность взаимодействия с программным обеспечением.

4.6 Выводы по разделу

Проведенное тестирование симулятора швейной машинки подтвердило его функциональную состоятельность и готовность к использованию в учебной среде. Проверка охватила все ключевые аспекты работы приложения: от базового взаимодействия с интерфейсом до сложных сценариев с обводкой, обрезкой и сшиванием ткани. Реализация каждой функции была протестирована на соответствие логике и ожидаемому результату.

В процессе проверки особое внимание уделялось устойчивости симулятора к ошибкам пользователя, а также поведению системы в нештатных ситуациях. Результаты показали, что приложение успешно предотвращает недопустимые действия, информирует пользователя о допущенных ошибках и обеспечивает стабильную работу при последовательном выполнении различных операций. Это подтверждает его надежность и продуманную архитектуру.

Кроме того, был подтвержден высокий уровень интерактивности и визуальной достоверности. Анимации, отклик интерфейса и поведение объектов сцены обеспечивают эффект присутствия и способствуют лучшему усвоению материала. Пользователь может последовательно выполнять все действия, необходимые для понимания работы швейной машинки, без необходимости прибегать к дополнительным инструкциям.

Таким образом, тестирование продемонстрировало, что разработанный симулятор соответствует поставленным целям проекта. Он является полноценным программным продуктом, способным эффективно решать обучающие задачи, повышая качество и безопасность образовательного процесса. Выявленные в процессе тестирования незначительные замечания были устранены, что дополнительно повысило надежность и удобство использования системы.

5 Руководство пользователя

5.1 Назначение и возможности симулятора

Разработанный симулятор швейной машинки представляет собой обучающее программное средство, предназначенное для ознакомления пользователей с основами работы бытовой швейной машины в интерактивной форме. Основная цель приложения – предоставить интуитивно понятный инструмент для изучения принципов работы швейной техники, без необходимости взаимодействия с реальным оборудованием.

Симулятор ориентирован на студентов учебных заведений, начинающих специалистов, а также всех, кто интересуется основами шитья. Программа позволяет безопасно и без риска для оборудования освоить базовые операции, включая выбор материалов, подготовку ткани, выполнение обводки по трафарету, обрезку и сшивание.

Ключевые возможности симулятора:

– свободное перемещение пользователя по виртуальной мастерской;

– выбор вида ткани и цвета нитей из инвентаря;

– интерактивная обводка ткани по выбранному трафарету;

– выполнение виртуальной обрезки ткани по заданному контуру;

– сшивание подготовленных фрагментов ткани на швейной машинке с визуализацией процесса;

– изменение вида строчки с прямой на зигзагообразную и обратно;

– изучение конструктивных элементов швейной машинки через специальную обучающую панель.

Симулятор создан с использованием современных технологий (Unity, C#, Autodesk 3ds Max), что позволило добиться высокого уровня визуализации, интерактивности и реалистичности модели. Приложение может использоваться как самостоятельное обучающее средство, так и в составе комплексных курсов по обучению ручному труду, дизайну и прикладному творчеству.

5.2 Системные требования и запуск приложения

Для корректной и стабильной работы симулятора швейной машинки необходимо наличие персонального компьютера, соответствующего минимальным или рекомендуемым системным требованиям. Приложение не требует подключения к сети Интернет, легко запускается и работает в режиме одиночного пользователя.

Минимальные системные требования:

– операционная система: Windows 10 (64-разрядная);
– процессор: Intel Core i3 или аналогичный AMD;
– оперативная память: 4 ГБ;
– видеокарта: NVIDIA GeForce GTX 750 / AMD Radeon R7 260X;
– свободное место на диске: 2 ГБ;
– разрешение экрана: 1280×720 (HD).
Рекомендуемые системные требования:
– операционная система: Windows 10 / 11 (64-разрядная);
– процессор: Intel Core i5 и выше;
– оперативная память: от 8 ГБ;
– видеокарта: NVIDIA GeForce GTX 1050 Ti / AMD Radeon RX 560 и выше;
– свободное место на диске: 4 ГБ;
– разрешение экрана: 1920×1080 (Full HD) или выше.
Порядок установки и запуска:
– скопировать папку с симулятором на жесткий диск компьютера;
– перейти в директорию с программой и найти исполняемый файл SewingMachineSimulator.exe;
– дважды кликнуть по файлу для запуска приложения;
– после загрузки появится главное меню симулятора, с которого начинается работа пользователя.

Установка не требует дополнительных драйверов или программ. Все необходимые библиотеки включены в дистрибутив. При первом запуске создаются файлы пользовательских настроек, которые могут быть сброшены через соответствующий пункт меню.

5.3 Основное меню и навигация

После запуска симулятора швейной машинки пользователю отображается главное меню, предоставляющее доступ ко всем ключевым функциям приложения, таким как начало симуляции и настройки. Интерфейс оформлен в едином визуальном стиле, соответствующем тематике шитья, с использованием декоративных графических элементов и текстур, создающих атмосферу уютной мастерской. Все элементы интерфейса расположены интуитивно понятно и сопровождаются визуальными подсказками, что облегчает освоение функционала даже для пользователя без предварительного опыта.

Главное меню включает следующие элементы управления:

– кнопка «Начать симуляцию» – переводит пользователя в основную сцену симулятора, где доступно взаимодействие с элементами виртуальной мастерской, такими как: швейная машинка, инвентарь, ткани, нитки, трафареты и другие вспомогательные панели. Пользователь может передвигаться по симулятору с помощью клавиш WASD на клавиатуре, а также поворачивать камерой в разные стороны с помощью правой клавиши мыши. Пример экрана после перехода показан на картинке 5.1;

Рисунок 5.1 – Основная сцена симулятора

– кнопка «Настройки» – открывает меню конфигурации, позволяющее изменить громкость звука, переключить режим отображения (полноэкранный или оконный), а также сбросить пользовательские данные. Интерфейс настроек демонстрируется на рисунке 5.2;

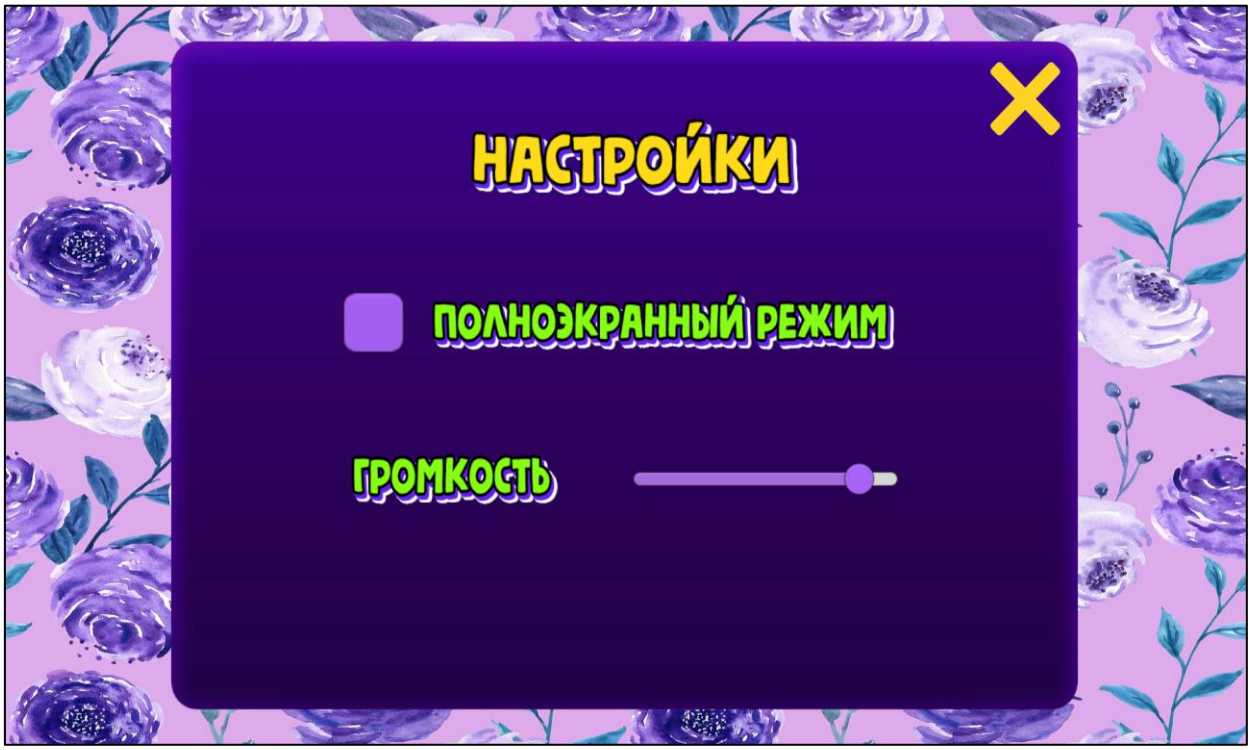

Рисунок 5.2 – Интерфейс настроек

– кнопка «Выход» – завершает работу приложения и закрывает симулятор.

Все элементы меню сопровождаются понятными подписями и визуальной обратной связью при наведении курсора, что делает навигацию простой даже для пользователей без опыта работы с подобными программами. Расположение элементов оптимизировано под стандартные разрешения экрана и остается удобным как в оконном, так и в полноэкранном режиме.

Переход между сценами (меню и симуляция) осуществляется мгновенно, без задержек и промежуточных загрузок, что обеспечивает плавный пользовательский опыт. Навигация внутри симулятора также реализована в виде простых и последовательных действий, подробно описанных в следующем разделе.

5.4 Работа в симуляторе: пошаговая инструкция

Процесс работы с симулятором швейной машинки представляет собой последовательность действий, направленных на подготовку и выполнение виртуального шитья. Для удобства пользователя весь цикл взаимодействия с системой структурирован и интуитивно понятен, что делает обучение безопасным и эффективным. Инструкция охватывает все этапы – от запуска приложения до завершения сеанса, обеспечивая плавное погружение в процесс.

Сначала пользователь запускает симулятор, после чего загружается стартовая сцена с главным меню, где доступна кнопка «Начать симуляцию» для перехода в рабочую среду. В симуляционной среде открывается доступ к инвентарю, где необходимо выбрать цвет ткани, подходящую для дальнейшей работы, в зависимости от желаемого результата. Далее пользователь переходит к разделу с трафаретами, как показано на рисунке 5.3, чтобы выбрать подходящий шаблон для раскроя материала, что позволяет создать заготовку нужной формы.

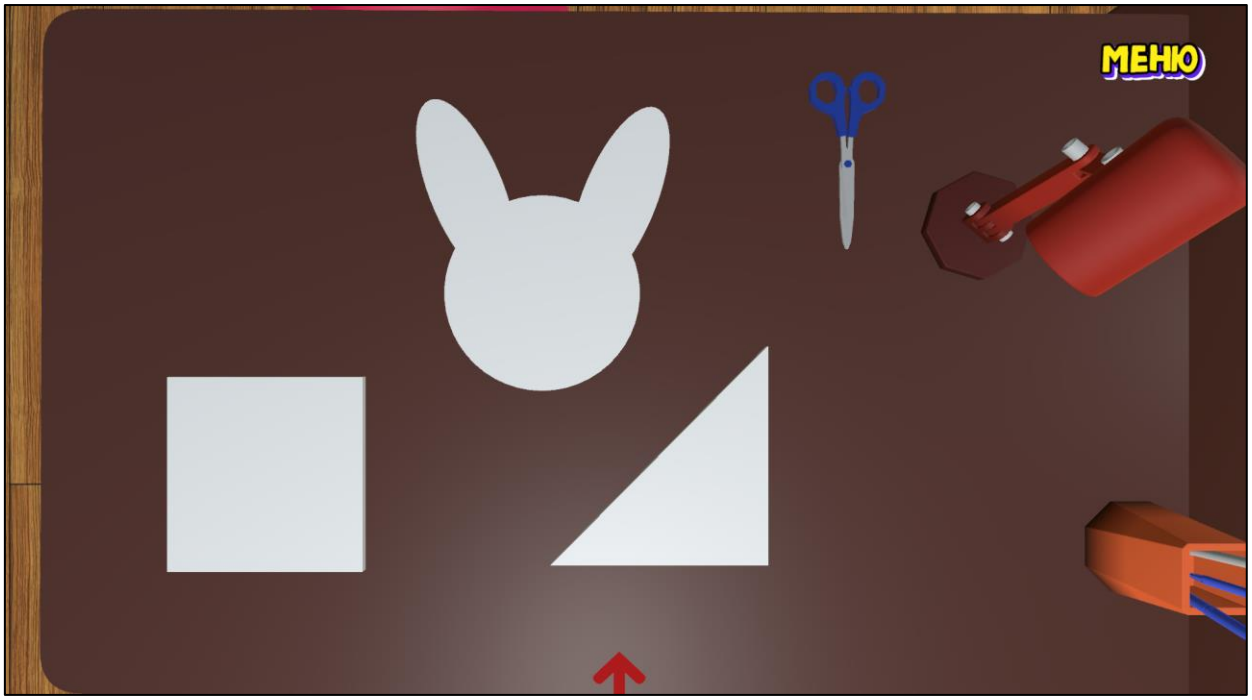

Рисунок 5.3 – Этап с трафаретами

На следующем этапе выполняется обработка трафарета, представляющая собой ключевой шаг в подготовке материала для дальнейшей работы. Пользователь сначала выбирает подходящий шаблон из доступного набора, затем обводит его по контуру фигуры, используя интерактивный инструмент, который позволяет точно следовать линиям трафарета. После завершения обводки пользователь переходит к обрезке ткани: для этого необходимо навести курсор на виртуальные ножницы и нажать, что запускает анимацию обрезки.

Этот процесс позволяет подготовить заготовку нужной формы, которая будет использоваться на следующих этапах шитья. Для большей наглядности данный этап проиллюстрирован на рисунке 5.4, где показаны все шаги обработки трафарета.

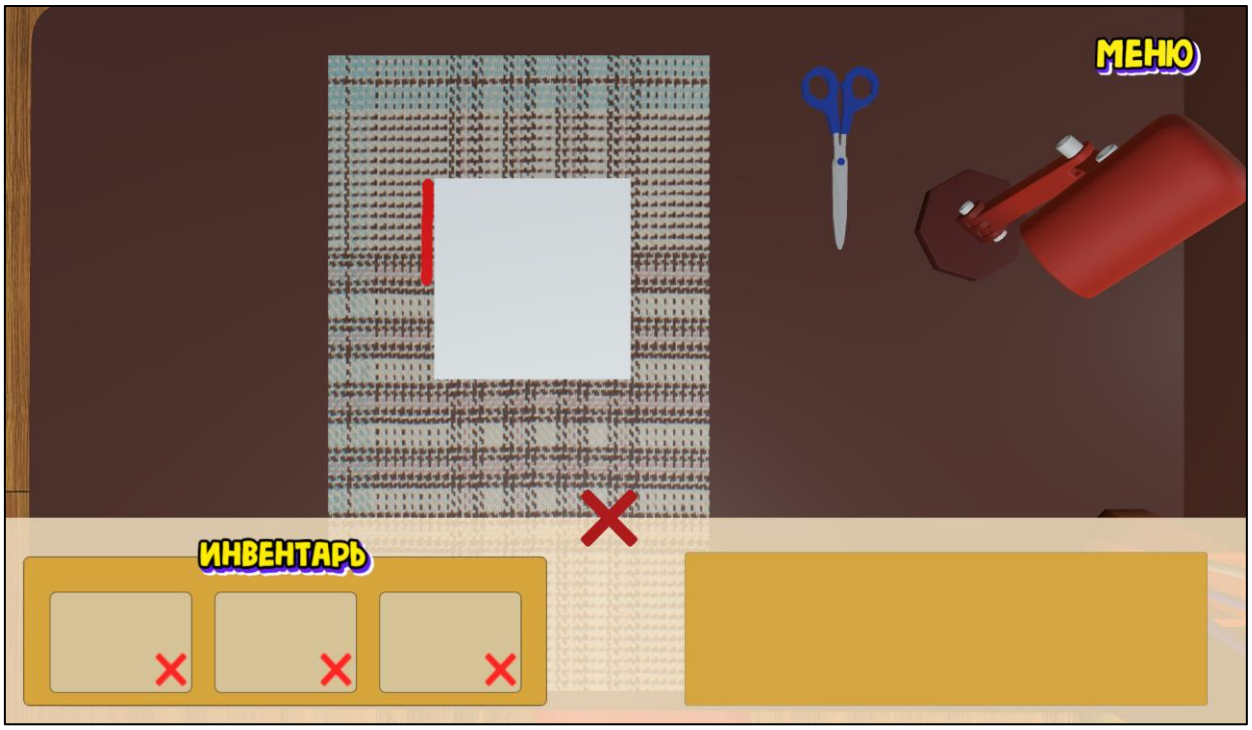

Рисунок 5.4 – Этап обводки и обрезки трафарета

После завершения подготовки материала пользователь переходит к основной сцене симулятора, где центральное место занимает виртуальная швейная машинка, полностью готовая к работе. На этом этапе из инвентаря, доступного через панель инструментов, извлекается ранее подготовленная выкройка, которая автоматически размещается в рабочей зоне машинки. Перед началом шитья пользователю предоставляется возможность активировать режим изучения устройства, специально разработанный для образовательных целей. В этом режиме фиолетовым цветом отображаются интерактивные элементы швейной машинки, такие как игла, прижимная лапка, маховик и шпульный механизм, каждый из которых сопровождается подробным описанием функций. Все элементы режима изучения наглядно представлены на рисунке 5.5, где показаны интерактивные части машинки с соответствующими пояснениями, что делает процесс обучения более доступным и эффективным.

Рисунок 5.5 – Все элементы режима изучения

Затем пользователь выбирает цвет нитки, вставляет шпульку и размещает ткань под иглой, опуская лапку для фиксации материала. После подготовки машины предлагается выбрать тип строчки – прямую или зигзагообразную, что влияет на визуальный стиль шва. Запускается процесс шитья, показанный на рисунке 5.6, сопровождаемый анимацией движения иглы, ткани и формирования прошитой линии, что создает реалистичное восприятие работы.

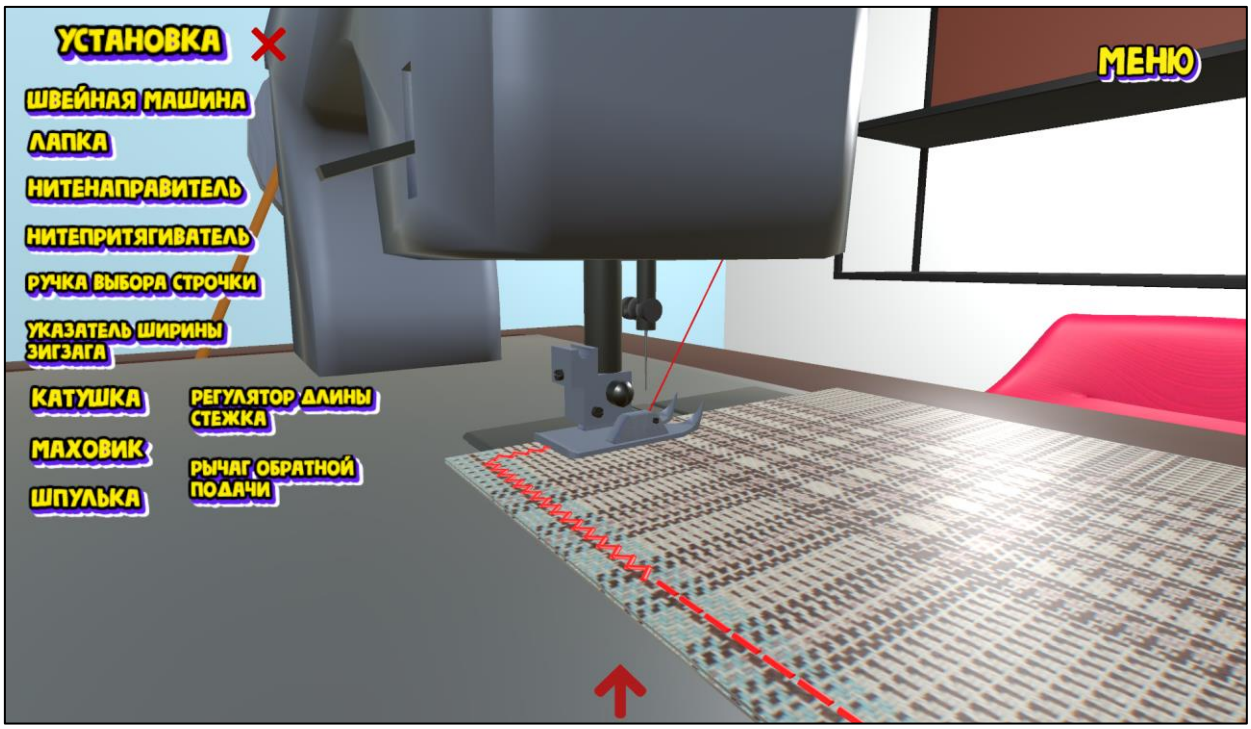

Рисунок 5.6 – Процесс шитья

После завершения процесса шитья пользователь получает свободу действий: он может выйти из активного режима работы с швейной машинкой, чтобы свободно перемещаться по виртуальной мастерской, внимательно рассматривая различные детали интерьера, такие как текстуры тканей, расположение инструментов или конструктивные элементы оборудования. Это позволяет глубже погрузиться в атмосферу симуляции и лучше понять контекст рабочего пространства. Альтернативно, если пользователь желает продолжить обучение или попробовать новый подход, он может вернуться в главное меню, где доступна опция для запуска симуляции заново, начиная цикл с выбора материалов и настройки параметров.

5.5 Выводы по разделу

Разработанное руководство пользователя и инструкция по работе с симулятором швейной машинки обеспечивают базовую информационную поддержку, необходимую для начала работы с программным продуктом. Представленные материалы позволяют ознакомиться с функциональными возможностями симулятора и общими принципами его использования. Тестирование и визуальная поддержка, оформленная в виде скриншотов, подтверждают стабильность и надежность работы приложения, что делает его полезным инструментом для образовательных учреждений и индивидуального изучения.

Дополнительно стоит отметить, что симулятор демонстрирует высокую степень автономности и простоту эксплуатации, благодаря чему может использоваться как в очной, так и в дистанционной форме обучения. Отсутствие необходимости подключения к сети Интернет делает его удобным для применения в различных условиях – от компьютерных классов до индивидуальных пользовательских систем.

Созданное руководство пользователя охватывает основные функции симулятора и обеспечивает общее представление о структуре управления. Это позволяет эффективно использовать программное средство в учебном процессе, повышая его доступность и практическую ценность.

6 Технико-экономическое обоснование проекта

Основной целью данного раздела является экономическое обоснование целесообразности разработки симулятора, предназначенного для интерактивного обучения принципам работы бытовой швейной машинки. Разработка средства актуальна в условиях необходимости снижения затрат на оборудование в образовательном процессе, обеспечения безопасности и повышения качества освоения практических навыков в швейном деле. Кроме того, симулятор позволяет расширить доступ к обучению вне зависимости от наличия материальной базы.

6.1 Общая характеристика разрабатываемого программного средства

Объектом разработки является 3D-симулятор швейной машинки, представляющий собой однопользовательское приложение, реализованный с использованием игровых технологий Unity – для создания интерактивной логики, визуализации и управления сценой, а также графического редактора 3ds Max – для моделирования трехмерных объектов, включая саму машинку и элементы виртуального окружения. Программа предназначена для студентов и начинающих пользователей, изучающих принципы работы швейной машинки без использования физического оборудования. Программное средство предоставляет доступ к интерактивной модели машинки, позволяет выполнять действия по обводке трафарета, обрезке ткани, выбору типа строчки и запуску процесса шитья.

Симулятор ориентирован на студентов учреждений среднего специального образования, а также на начинающих пользователей, заинтересованных в приобретении практических знаний без необходимости использования реального оборудования. Интуитивно понятный интерфейс и реалистичная визуализация делают программное средство удобным в освоении и применении в образовательной среде.

Способ монетизации – продажа разработанного программного средства и имущественных прав на него заказчику. В условиях цифровой трансформации образования данное решение может быть выгодно как с точки зрения прямой продажи лицензии, так и путем дальнейшего расширения программного продукта на основе индивидуальных требований заказчика.

В рамках данного раздела необходимо определить затраты, произведенные на всех стадиях разработки описанного программного средства. Это включает расчет затрат на разработку и тестирование симулятора, оценку стоимости материалов, оборудования, а также затрат на сопровождение и адаптацию.

6.2 Исходные данные для проведения расчетов и маркетинговый анализ

Источниками исходных данных для данных расчетов выступают действующие законы и нормативно-правовые акты. Расчеты будут проводится для различных экономических показателей. Исходные данные для расчета приведены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчета

Наименование показателя	Условные обозначения	Норматив
Численность разработчиков, чел	$Ч_р$	1
Норматив дополнительной заработной платы, %	$Н_{дз}$	15
Ставка отчислений в Фонд социальной защиты населения, %	$Н_{фсзн}$	34
Ставка отчислений в БРУСП «Белгосстрах», %	$Н_{бгс}$	0,6
Норматив прочих прямых затрат, %	$Н_{пз}$	25
Норматив расходов на сопровождение и адаптацию, %	$Н_{рса}$	10
Норматив накладных расходов, %	$Н_{опб,опх}$	25

Далее эти данные понадобятся для расчета всех необходимых параметров в процессе экономического анализа приложения.

6.3 Методика обоснования цены

Выбор эффективных программных решений в первую очередь определяется их экономической целесообразностью и возможностью получения положительного результата от внедрения. В рамках оценки эффективности разрабатываемого симулятора швейной машинки основное внимание уделяется расчету затрат на его создание и анализу потенциальной прибыли от реализации программного средства.

Экономический эффект для разработчика выражается в виде чистой прибыли, остающейся после реализации программного продукта. Для заказчика и конечного пользователя выгода проявляется в снижении затрат на приобретение и обслуживание физического оборудования, повышении безопасности учебного процесса, а также в возможности эффективного освоения базовых навыков без дополнительных материальных расходов.

Разработка симулятора направлена на автоматизацию и визуализацию обучения в области начального швейного дела. Использование такого программного продукта позволяет:

– снизить трудоемкость подготовки обучающихся к работе с реальной техникой за счет предварительного знакомства с ее конструкцией и функционалом в виртуальной среде;

– минимизировать риск повреждения дорогостоящего оборудования;

– сократить расходы на закупку учебных машинок и материалов;

– повысить качество и наглядность образовательного процесса.

Основные статьи затрат, Которые формируют себестоимость программного средства, включают:

– основную и дополнительную заработную плату исполнителей;

– отчисления в фонд социальной защиты населения и на обязательное страхование от несчастных случаев;

– расходы на материалы;

– затраты на оборудование, используемое в процессе разработки (манипуляторы, периферия);

– прочие прямые расходы, связанные с организационными или вспомогательными затратами;

– накладные расходы;

– затраты на сопровождение и адаптацию программного продукта после сдачи его заказчику.

На основе совокупных затрат определяется полная себестоимость разработки симулятора.

6.3.1 Объем программного средства

Для определения временных и трудовых затрат, связанных с разработкой симулятора швейной машинки, был составлен перечень основных этапов работ. В таблице ниже представлены укрупненные этапы, а также соответствующие им затраты рабочего времени и количество исполнителей.

Таблица 6.2 – Затраты рабочего времени на разработку ПС

Содержание работ	Должность исполнителя	Количество исполнителей	Затраты рабочего времени, дней
1. Разработка 3D-модели швейной машинки	3D-дизайнер	1	7
2. Моделирование окружения (рабочее место, стол, интерьер)	3D-дизайнер	1	5
3. Импорт и сборка сцены в Unity	Unity-разработчик	1	4
4. Программирование логики взаимодействия	Unity-разработчик	1	14
5. Реализация анимаций (игла, маховик, обрезка ткани и т.д.)	Unity-разработчик	1	25
6. Разработка пользовательского интерфейса	Unity-разработчик	1	10
7. Создание функционала выбора ткани и ниток	Unity-разработчик	1	4
8. Тестирование основных функций	Тестировщик	1	4
9. Рефакторинг и исправление выявленных ошибок	Unity-разработчик	1	5
Всего			69

Результат (значение «всего») по данной таблице будет использовано далее для расчетов. Всего было затрачено 69 дней на разработку программного средства.

6.3.2 Основная заработная плата

Для расчета основной заработной платы использовались данные по средней месячной оплате труда специалистов начального уровня (Junior) в области 3D-моделирования и разработки на движке Unity. На основе анализа рынка установлено:

– средняя заработная плата 3D-дизайнера – 2 000 руб. в месяц;

– средняя заработная плата Unity-разработчика – 2 400 руб. в месяц;

– средняя заработная плата тестировщика – 1 900 руб. в месяц.

Согласно данным таблицы 6.2, общая продолжительность работы 3D-дизайнера составила 12 календарных дней, что эквивалентно 0,57 месяца. Unity-разработчик работал над проектом в течение 62 дней (включая сборку сцены, реализацию логики, анимаций, интерфейса и выбора параметров), что составляет 2,95 месяца. Тестировщик был задействован на протяжении 4 рабочих дней, или 0,19 месяца.

Основная заработная плата будет рассчитываться по формуле 6.1

$$С_{оз\textit{i}} = Т_{раз\textit{i}} \cdot С_{зп\textit{i}} \cdot К_{раз\textit{i}}, \quad (6.1)$$

где $С_{оз\textit{i}}$ – основная заработная плата i-го исполнителя, руб.;

$Т_{раз\textit{i}}$ – время раработки, месяцев;

$С_{зп\textit{i}}$ – средняя месячная заработная плата;

$К_{раз\textit{i}}$ – количество разработчиков, человек.

Рассчитаем основную заработную плату специалистов, привлекаемых для реализации программного средства:

$$С_{оз3Dдиз} = 0,57 \cdot 2\,000 \cdot 1 = 1140 \text{ руб.},$$

$$С_{озUP} = 2,95 \cdot 2\,400 \cdot 1 = 7\,080 \text{ руб.},$$

$$С_{озТест} = 0,19 \cdot 1\,900 \cdot 1 = 361 \text{ руб.},$$

Суммарная основная заработная плата составит:

$$С_{оз} = 1\,140 + 7\,080 + 361 = 8\,581 \text{ руб.}$$

Исходя из приведенных выше расчетов, основная заработная плата составит 8 581 рублей. Эта сумма будет использована в дальнейших расчетах.

6.3.3 Дополнительная заработная плата

Дополнительная заработная плата включает в себя выплаты, предусмотренные трудовым законодательством: оплата отпусков, выходных, праздничных дней и другие гарантированные выплаты, не включенные в основную зарплату. В экономических расчетах такие выплаты определяются по установленному

нормативу в процентах к основной заработной плате по формуле 6.2

$$С_{д.з.} = \frac{С_{о.з.} \cdot Н_{д.з.}}{100} \quad (6.2)$$

где $С_{оз}$ – основная заработная плата, руб.;

$Н_{дз}$ – норматив дополнительной заработной платы, %.

$$С_{дз} = 8\,581 \cdot 15 / 100 = 1287{,}15 \text{ руб.}$$

Таким образом, дополнительная заработная плата составляет 1 287,15 рублей, и включается в совокупные затраты на оплату труда при расчете себестоимости проекта.

6.3.4 Отчисления в Фонд социальной защиты населения

В соответствии с действующим законодательством, на фонд оплаты труда необходимо производить обязательные социальные отчисления.

К числу таких обязательных отчислений относятся:

– взносы в Фонд социальной защиты населения (ФСЗН), определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле 6.3

$$С_{фсзн} = \frac{(С_{оз} + С_{дз}) \cdot Н_{фсзн}}{100}, \quad (6.3)$$

где $С_{оз}$ – основная заработная плата, руб.;

$С_{дз}$ – дополнительная заработная плата на конкретное ПС, руб.;

$Н_{фсзн}$ – норматив отчислений в Фонд социальной защиты населения, %.

– отчисления в БРУСП «Белгосстрах» – систему государственного страхования от несчастных случаев на производстве и профессиональных заболеваний, вычисляются по формуле 6.4

$$С_{бгс} = \frac{(С_{оз} + С_{дз}) \cdot Н_{бгс}}{100}, \quad (6.4)$$

$$С_{фсзн} = \frac{(8\,581 + 1287{,}15) \cdot 34}{100} = 3355{,}17 \text{ руб.}$$

$$С_{бгс} = \frac{(8\,581 + 1287{,}15) \cdot 0{,}6}{100} = 59{,}21 \text{ руб.}$$

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 59,21 рублей, а в фонд социальной защиты населения – 3355,17 рублей.

6.3.5 Расходы на материалы

Несмотря на то, что разработка программного обеспечения в значительной степени осуществляется в цифровой среде, в процессе выполнения проекта все же возникла необходимость в использовании некоторых физических материалов, обеспечивающих подготовку технической документации и визуализацию концепции будущего симулятора.

Сумма расходов на материалы $С_М$ определяется из таблицы 6.3, в которой собраны понесенные затраты на разработку программного средства.

Таблица 6.3 – Материальные затраты на разработку ПС

Вид материальных затрат	Фактические затраты, руб.
Бумага офисная, 1 пачка	11,50
Образец ткани (демонстрационный кусок), 1 шт.	9,80
Всего	21,03

Сумма расходов на материалы составляет 21,03 рубль.

6.3.6 Расходы на специальное оборудование и платные услуги

Сумма расходов на специальное оборудование и платные услуги $С_{СОПУ}$ определяется как итог таблицы 6.4. был приобретен манипулятор «Мышь компьютерная» ценой с НДС 87,00 рублей и USB-накопитель ценой 21,00 рубль. Платных услуг со стороны иных юридических лиц разработчикам рассматриваемого программного средства оказано не было.

Таблица 6.4 – Стоимость оборудования, специальных программ и платных услуг для разработки ПС

Вид затрат	Фактические затраты, руб.
Устройство ввода «Мышь компьютерная»	87,00
USB-накопитель (флешка) 32 ГБ	21,00
Всего	108,00

Таким образом, расходы на специальное оборудование и платные услуги составили 108,00 рубль, и эта сумма будет учтена при формировании общей себестоимости программного средства.

6.3.7 Прочие прямые затраты

Сумма прочих затрат $С_{пз}$ рассчитывается как произведение основной заработной платы $С_{оз}$, затраченной на разработку программного средства, и установленного норматива прочих прямых расходов $Н_{пз}$, действующего в организации. Расчеты проводились по формуле 6.5

$$С_{пз} = \frac{С_{оз} \cdot Н_{пз}}{100} \quad (6.5)$$

$$С_{пз} = 8\ 581 \cdot 25 / 100 = 2\ 145{,}25 \text{ руб.}$$

Таким образом, прочие прямые затраты составили 2 145,25 рублей.

6.3.8 Накладные расходы

Сумма накладных расходов $С_{общ,опх}$ определяется как произведение основной заработной платы исполнителей на конкретное программное средство $С_{оз}$ на норматив накладных расходов в целом по организации $Н_{общ,опх}$ и рассчитывается по формуле 6.6

$$С_{общ,опх} = \frac{С_{оз} \cdot Н_{общ,опх}}{100}, \quad (6.6)$$

$$С_{общ,опх} = 8\ 581 \cdot 25 / 100 = 2\ 145{,}25 \text{ руб.}$$

При сумме основной зарплаты 8 581 рублей накладные расходы составили 2 145,25 рублей.

6.3.9 Расходы на разработку программного средства

Сумма расходов на разработку программного средства $С_р$ определяется как сумма основной заработной платы и дополнительной заработной платы исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов (формула 6.7)

$$С_р = С_{оз} + С_{дз} + С_{фсзн} + С_{бгс} + С_м + С_{сопу} + С_{пз} + С_{общ,обх} \quad (6.7)$$

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

$$С_р = 8\ 581 + 1287{,}15 + 3355{,}17 + 59{,}21 + 21{,}03 + 108{,}00 + 2145{,}25 + 2145{,}25 =$$
$$= 17\ 702{,}06 \text{ руб.}$$

Общая сумма расходов на разработку средства составит 17 702,06 рублей. Эта величина отражает совокупные затраты на все этапы создания симулятора. Полученный результат используется для дальнейшего расчета себестоимости и оценки экономической эффективности проекта. Он также служит ориентиром при планировании аналогичных разработок в будущем. Важно учитывать, что на практике фактические расходы могут варьироваться в зависимости от масштабов проекта и применяемых технологий. Проведённые расчёты позволяют заранее оценить затраты и обосновать бюджет проекта.

6.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработку на расходы на сопровождение и адаптацию Нрса. В соответствии с установленным нормативом, данный показатель составляет 10% от суммы ранее рассчитанных расходов.

Таким образом сумма расходов на сопровождение и адаптацию программного средства вычисляется по формуле 6.8

$$С_{рса} = \frac{С_р \cdot Н_{рса}}{100}. \quad (6.8)$$

Подставим значения для определения суммы расходов на сопровождение и адаптацию программного средства.

$$С_{рса} = 17\,702{,}06 \cdot 10 / 100 = 1770{,}21 \text{ руб.}$$

Таким образом, расходы на сопровождение и адаптацию симулятора составляют 1 770,21 рублей, и будут учтены при расчете общей стоимости программного продукта.

6.3.11 Полная себестоимость

Общая сумма расходов включает все затраты, связанные как с непосредственной разработкой программного средства, так и с его последующим сопровождением и адаптацией. Она отражает полную себестоимость проекта и используется для определения отпускной цены программного продукта.

Общая сумма расходов на разработку с затратами на сопровождение и адаптацию программного средства $С_п$ определяется как сумма двух элементов: суммы расходов на разработку $С_р$ и суммы расходов на сопровождение и адаптацию программного средства $С_{рса}$ (формула 6.9).

$$С_п = С_р + С_{рса}. \quad (6.9)$$

$$С_п = 17702{,}06 + 1770{,}21 = 19\,472{,}27 \text{ руб.}$$

Размер полной себестоимости симулятора швейной машинки составила 19 472,27 рубль. Этот показатель ляжет в основу расчета отпускной цены программного продукта.

6.3.12 Расчет экономического эффекта

После расчета полной себестоимости разработки программного средства необходимо определить его рыночную цену и оценить экономическую эффективность проекта. Для этого сравниваются затраты на создание

симулятора с ценами аналогичных решений на рынке, а также рассчитываются потенциальная прибыль и уровень рентабельности.

Таблица 6.5 – Анализ аналогов

Приложение	Адрес	Цена	Краткое описание
Учебный тренажер по лапароскопии	https://urteks.ru/store/medicinskie-trenazhery-maksim-ilyusha/medical/virtualnye-trenazhery-simulyatory/uchebnyy-trenazher-po-laparoskopii/	42 958 руб.	Тренажер предназначен для отработки базовых навыков проведения лапароскопических операций. Станция оснащена стойкой с сенсорным монитором, камерой с функцией фиксации и коробок для проведения процедур.
Комплексный образовательный симулятор	https://globural.ru/katalog/robototekhnika-i-konstruktory/kvadrokoptery-i-drony/kvadrokoptery-dlya-shkoly/kompleksnyy-obrazovatelnyy-simulyator/	35 560 руб.	Данный тренажерный комплекс предназначен для подготовки операторов беспилотных воздушных судов мультироторного и самолетного (летающее крыло) типов
Симулятор вождения Logitech G29 + H-шифтер	https://prom.ua/p2603141506-simulyator-vozhdeniya-logitech.html	47 197 руб.	Комплект содержит все необходимое для начала виртуальных гонок.

Анализ существующих приложений, связанных с обучающими симуляторами, показал, что стоимость программных продуктов с подобным уровнем графики, интерактивности и образовательной направленности может варьироваться от 30 000 до 50 000 руб. в зависимости от функционала, уровня детализации и условий лицензирования.

В рамках настоящего проекта целесообразно установить отпускную цену с учетом НДС на уровне 35 000 рублей. Эта сумма попадает в нижний диапазон рыночной стоимости и обеспечивает конкурентоспособность продукта при продаже учебным заведениям.

Рассчитаем цену без НДС (20%):

$$Ц_{без\ НДС} = 35\ 000 \cdot (100 / 120) = 29\ 166{,}67 \text{ руб.}$$

Рассчитаем прибыль от реализации:

$$П_{рп} = 29\ 166{,}67 – 19\ 472{,}27 = 9\ 694{,}40 \text{ руб.}$$

Рентабельность определяется по формуле 6.10:

$$Р_{пс} = \frac{П_{рп}}{С_{п}} \cdot 100 \qquad (6.10)$$

$$Р_{пс} = (9\ 694{,}40 / 19\ 472{,}27) \cdot 100 = 49{,}78\ \%.$$

Таким образом, реализация симулятора швейной машинки по установленной цене обеспечивает положительный экономический эффект. Проект является коммерчески оправданным, демонстрирует высокий уровень рентабельности и может быть выгоден при разовой продаже.

6.4 Выводы по разделу

В результате выполнения технико-экономического обоснования были рассчитаны все основные показатели, отражающие затраты на разработку симулятора швейной машинки, его полную себестоимость, а также экономическую эффективность проекта. Общая сумма расходов на создание программного средства, включая сопровождение и адаптацию, составила 19 472,27 рублей.

На основании анализа цен аналогичных обучающих программных продуктов на рынке была установлена рекомендованная отпускная цена в размере 35 000 руб. с НДС, что соответствует нижней границе рыночной стоимости. Это обеспечивает конкурентоспособность программного средства при продаже в учебные учреждения и организации, осуществляющие подготовку по направлениям, связанным с швейным делом.

В таблице 6.6 приведены результаты расчетов экономических показателей, проведенных в данном разделе.

Таблица 6.6 – Результаты расчетов

Наименование показателя	Значение показателя
Количество исполнителей, чел.	1
Время разработки, дней	69
Основная заработная плата, руб.	8 581
Дополнительная заработная плата, руб.	1287,15
Отчисления в Фонд социальной защиты населения, руб	3355,15
Отчисления в БРУСП «Белгосстрах», руб.	59,21
Расходы на материалы	21,03
Расходы на специальное оборудование и платные услуги	108,00
Прочие прямые затраты, руб.	2145,25
Накладные расходы, руб.	2145,25
Себестоимость разработки программного средства, руб.	17 702,06
Расходы на сопровождение и адаптацию, руб.	1 770,21
Полная себестоимость, руб.	19 472,27
Отпускная цена без НДС, руб.	29 166,67
Прибыль от реализации программного средства, руб.	9 694,40
Уровень рентабельности, %	49,78

При расчете прибыли и рентабельности было установлено, что реализация программного продукта по заявленной цене позволяет получить прибыль в размере 9 694,40 руб., а уровень рентабельности составляет 49,78%. Таким образом, разработка симулятора швейной машинки является экономически оправданной и может быть эффективно внедрена как в учебный процесс.

Заключение

В ходе выполнения дипломного проекта была поставлена и успешно реализована задача создания интерактивного симулятора швейной машинки, ориентированного на учебные цели. Проект охватывает все этапы программной разработки: от анализа аналогов и проектирования архитектуры до реализации, тестирования и экономической оценки.

Симулятор разработан с применением современных инструментов и технологий, включая игровой движок Unity, 3ds Max и язык программирования C#. В рамках проекта была создана 3D-модель швейной машинки, реализованы пользовательский интерфейс, анимации механических элементов, функции выбора и обработки ткани, а также сценарии взаимодействия с виртуальной швейной машинкой. При проектировании особое внимание уделялось удобству использования, визуальной наглядности и соответствию интерфейса образовательной задаче.

Проведенное тестирование подтвердило стабильность работы симулятора, корректность выполнения основных функций и адекватную реакцию на внештатные ситуации. Благодаря пошаговой логике взаимодействия пользователь может интуитивно осваивать базовые приемы работы с швейной машинкой в безопасной виртуальной среде.

Экономическое обоснование показало, что самостоятельная разработка программного продукта в рамках дипломного проекта позволяет существенно снизить затраты по сравнению с коммерческой разработкой аналогичных решений, сохранив при этом высокое качество и функциональность.

Разработанный симулятор может быть использован в учебном процессе в учреждениях начального образования, а также служить основой для дальнейшего расширения: добавления новых видов швов, встроенных подсказок и расширенного обучающего модуля.

Таким образом, цель дипломного проекта достигнута, а его результаты могут быть применены как в образовательной, так и в демонстрационной среде.

Список использованных источников

1 Машинист и метро 3D [Электронный ресурс]. – Режим доступа: https://vseigru.net/igry-poezda/42934-igra-mashinist-moskovskogo-metro-3d.html (дата обращения: 24.03.2025).

2 Flight Simulator [Электронный ресурс]. – Режим доступа: https://yandex.ru/games/app/198745 (дата обращения: 25.03.2025).

3 Метро симулятор [Электронный ресурс]. – Режим доступа: https://yandex.ru/games/app/297165 (дата обращения: 25.03.2025).

4 draw.io – Онлайн-редактор диаграмм [Электронный ресурс]. – Режим доступа: https://app.diagrams.net/ (дата обращения: 28.03.2025).

5 Вайсфельд М. Объектно-ориентированное мышление. – 4-е изд. – М.: Вильямс, 2019. – 352 с.

6 Autodesk 3ds Max 2025. Справочная система [Электронный ресурс]. – Режим доступа: https://help.autodesk.com/view/3DSMAX/2025/ENU (дата обращения: 06.04.2025).

7 Горелик А. Г. Самоучитель 3ds Max 2020. – СПб.: БХВ-Петербург, 2020. – 544 с

8 Unity Manual. Официальная документация Unity [Электронный ресурс]. – Режим доступа: https://docs.unity3d.com/Manual (дата обращения: 22.04.2025).

9 TextDrom – онлайн-генератор текстов и логотипов [Электронный ресурс]. – Режим доступа: https://textdrom.com/ (дата обращения: 02.05.2025).

10 Ферроне Х. Изучаем C# через разработку игр на Unity. 5-е изд. – СПб.: Питер, 2025. – 400 с.

Приложение А Диаграмма вариантов использования

Приложение Б Общая архитектура симулятора

Приложение В Блок-схема алгоритма работы симулятора

Приложение Г Выбор ткани

Приложение Д Обводка и обрезка ткани

Приложение Е Сшивание ткани

Приложение Ж Таблица технико-экономических показателей

Приложение И Листинг SewingMachineController.cs – обрезка ткани

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;

public class SewingMachineController : MonoBehaviour
{
    public Transform needle;
    public LineRenderer threadLineRenderer;
    public Transform[] pathPoints;
    public Button startButton;
    public GameObject stitchPrefab;
    public Transform cube1;
    public Transform cube2;
    public GameObject finishPanel;
    public GameObject finishObgect;

    private AudioSource audioSource;
    public AudioSource sewingAudioSource;
    public float chunkLength = 0.5f;
    private float audioProgress = 0f;
    private Coroutine audioCoroutine;

    public Sprite rewardSprite;
    private Inventory inventory;

    public float stitchSpacing = 0.1f;
    public float threadSpeed = 3f;
    public float fabricMoveDistance = 0.05f;
    public float sideLength = 1.243f;

    private readonly Vector3 fixedStitchPosition = new Vector3(-
6.5511f, 5.3392f, 7.2906f);

    private bool isSewing = false;
    private Animator needleAnimator;
    public Animator secondAnimator;
    private int currentPointIndex = 0;
    private float threadProgress = 0f;
    private int threadPositionCount = 1;

    private Vector3[] moveDirections;
    private int currentDirectionIndex = 0;
    private float distanceOnSide = 0f;

    private bool isZigzagMode = false;
    private bool zigzagLeft = true;
    public float zigzagOffset = 0f;

    void Start()
    {
```

```
        finishPanel.SetActive(false);
        needleAnimator = needle ? needle.GetComponent<Anima-
tor>() : null;
        audioSource = GetComponent<AudioSource>();

        if (!audioSource)
        {
            Debug.LogError("AudioSource не найден на объекте
SewingMachineController!");
        }
        if (!needleAnimator) Debug.LogError("Needle Animator not
found!");
        if (!threadLineRenderer) Debug.LogError("Thread LineRen-
derer not assigned!");
        if (pathPoints.Length == 0) Debug.LogError("Path points
not assigned!");
        if (!stitchPrefab) Debug.LogError("Stitch Prefab not as-
signed!");
        if (!cube1 || !cube2) Debug.LogError("Cube1 or Cube2 not
assigned!");
        if (!startButton) Debug.LogError("Start Button not as-
signed!");

        threadLineRenderer.positionCount = 1;
        threadLineRenderer.SetPosition(0,   pathPoints[0].posi-
tion);
        startButton.onClick.AddListener(StartSewing);

        moveDirections = new Vector3[]
        {
            new Vector3(0, 0, -1),
            new Vector3(0, 0, -1),
            new Vector3(0, 0, -1),
            new Vector3(0, 0, -1)
        };

        inventory   =   GameObject.FindGameObjectWithTag("Inven-
toryPanel")?.GetComponent<Inventory>();
        if (inventory == null)
        {
            Debug.LogError("Inventory не найден!");
        }
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space) && isSewing)
        {
            needleAnimator.SetTrigger("on");
            secondAnimator?.SetTrigger("on");
            CreateStitch();
            PlayNextAudioChunk();
        }
```

```
        if (isSewing && currentPointIndex < pathPoints.Length -
1)
        {
            threadProgress += threadSpeed * Time.deltaTime;
            if (threadProgress >= 1f)
            {
                threadProgress = 0f;
                currentPointIndex++;
                threadPositionCount++;
                threadLineRenderer.positionCount  =  threadPosi-
tionCount;
                threadLineRenderer.SetPosition(threadPosi-
tionCount - 1, pathPoints[currentPointIndex].position);
            }
            else
            {
                Vector3 newPos = Vector3.Lerp(
                    pathPoints[currentPointIndex].position,
                    pathPoints[currentPointIndex + 1].position,
                    threadProgress
                );
                threadLineRenderer.positionCount  =  threadPosi-
tionCount;
                threadLineRenderer.SetPosition(threadPosi-
tionCount - 1, newPos);
            }
        }
    }

    void PlayNextAudioChunk()
    {
        if (sewingAudioSource == null || sewingAudioSource.clip
== null)
        {
            Debug.LogWarning("AudioSource  или  аудиоклип  не
назначены!");
            return;
        }

        if (audioProgress >= sewingAudioSource.clip.length)
        {
            audioProgress = 0f;
            return;
        }

        if (audioCoroutine != null)
        {
            StopCoroutine(audioCoroutine);
        }

        sewingAudioSource.time = audioProgress;
        sewingAudioSource.Play();
```

```
        audioCoroutine    =    StartCoroutine(StopSoundAfterSec-
onds(chunkLength));

        audioProgress += chunkLength;
    }

    private IEnumerator StopSoundAfterSeconds(float seconds)
    {
        yield return new WaitForSeconds(seconds);
        sewingAudioSource.Stop();
    }

    void StartSewing()
    {
        isSewing = true;
        currentPointIndex = 0;
        threadProgress = 0f;
        threadPositionCount = 1;
        threadLineRenderer.positionCount = 1;
        threadLineRenderer.SetPosition(0,   pathPoints[0].posi-
tion);

        currentDirectionIndex = 0;
        distanceOnSide = 0f;

        EventSystem.current.SetSelectedGameObject(null);
    }

    void CreateStitch()
    {
        if (isZigzagMode)
        {
            CreateZigzagStitch();
        }
        else
        {
            CreateStraightStitch();
        }

        MoveFabric();
    }

    void CreateStraightStitch()
    {
        GameObject  stitch1  =  Instantiate(stitchPrefab,  fixed-
StitchPosition, Quaternion.Euler(90, 0, 0));
        GameObject  stitch2  =  Instantiate(stitchPrefab,  fixed-
StitchPosition, Quaternion.Euler(90, 0, 0));

        stitch1.transform.localScale = new Vector3(0.01f, 0.02f,
0.01f);
        stitch2.transform.localScale = new Vector3(0.01f, 0.02f,
0.01f);
```

```
        stitch1.transform.SetParent(cube1,  worldPositionStays:
true);
        stitch2.transform.SetParent(cube2,  worldPositionStays:
true);

        stitch1.name = "Stitch1_" + Time.time;
        stitch2.name = "Stitch2_" + Time.time;
    }

    void CreateZigzagStitch()
    {
        float xOffset = zigzagLeft ? -zigzagOffset : zigzagOff-
set;
        float rotationAngle = zigzagLeft ? -50f : 50f;
        zigzagLeft = !zigzagLeft;

        Vector3  zigzagPos  =  fixedStitchPosition  +  new  Vec-
tor3(xOffset, 0, 0);

        GameObject stitch1 = Instantiate(stitchPrefab, zigzag-
Pos, Quaternion.Euler(90, 0, rotationAngle));
        GameObject stitch2 = Instantiate(stitchPrefab, zigzag-
Pos, Quaternion.Euler(90, 0, rotationAngle));

        stitch1.transform.localScale = new Vector3(0.01f, 0.02f,
0.01f);
        stitch2.transform.localScale = new Vector3(0.01f, 0.02f,
0.01f);

        stitch1.transform.SetParent(cube1, true);
        stitch2.transform.SetParent(cube2, true);

        stitch1.name = "ZigzagStitch1_" + Time.time;
        stitch2.name = "ZigzagStitch2_" + Time.time;
    }

    void MoveFabric()
    {
        if   (cube1.GetComponent<Animator>())   cube1.GetCompo-
nent<Animator>().enabled = false;
        if   (cube2.GetComponent<Animator>())   cube2.GetCompo-
nent<Animator>().enabled = false;

        Vector3 move = moveDirections[currentDirectionIndex] *
fabricMoveDistance;

        cube1.position += move;
        cube2.position += move;
        distanceOnSide += fabricMoveDistance;

        if (distanceOnSide >= sideLength)
        {
```

```
            distanceOnSide = 0f;
            currentDirectionIndex++;

            if (currentDirectionIndex >= 4)
            {
                isSewing = false;

                // Скрываем cube1 перед показом finishPanel
                if (cube1 != null)
                {
                    cube1.gameObject.SetActive(false);
                    Debug.Log("cube1 скрыт перед появлением fin-
ishPanel");
                }

                if (inventory != null && rewardSprite != null)
                {
                    int targetSlot = 2;
                    if (targetSlot < inventory.slots.Length &&
!inventory.isFull[targetSlot])
                    {
                        Image slotImage = inventory.slots[tar-
getSlot].GetComponent<Image>();
                        if (slotImage != null)
                        {
                            slotImage.sprite = rewardSprite;
                            slotImage.enabled = true;
                            inventory.isFull[targetSlot]      =
true;
                            inventory.itemTags[targetSlot]    =
rewardSprite.name;
                        }
                    }
                }

                if (finishPanel != null)
                {
                    finishPanel.SetActive(true);
                }

                cube2.SetParent(cube1, true);
                return;
            }

            cube1.Rotate(0, -90, 0, Space.World);
            cube2.Rotate(0, -90, 0, Space.World);

            cube1.position   =   new   Vector3(-5.902f,   5.33f,
7.912f);
            cube2.position   =   new   Vector3(-5.902f,   5.338f,
7.912f);
        }
```

```
        Debug.Log($"Fabric moved {move}. Distance on side: {dis-
tanceOnSide:F3}");
    }

    public void CloseFinishPanel()
    {
        if (finishPanel != null)
        {
            finishPanel.SetActive(false);
        }
    }

    public void EnableZigzagMode()
    {
        isZigzagMode = true;
        zigzagOffset = 0f;
        fabricMoveDistance = 0.02f;
        Debug.Log("Режим зигзага ВКЛ — параметры установлены");
    }

    public void DisableZigzagMode()
    {
        isZigzagMode = false;
        zigzagOffset = 0f;
        fabricMoveDistance = 0.05f;
        Debug.Log("Режим зигзага ВЫКЛ — параметры установлены");
    }
}
```

Приложение К Листинг TexturePainter.cs – обводка ткани

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

public class TexturePainter : MonoBehaviour
{
    [SerializeField] private Camera mainCamera;
    [SerializeField] private float brushSize = 0.75f;
    [SerializeField] private Color brushColor = Color.red;
    [SerializeField] private int textureSize = 1024;

    [Header("Cursor Settings")]
    [SerializeField] private Texture2D paintCursor;
    [SerializeField] private Texture2D defaultCursor;
    [SerializeField] private Vector2 cursorHotspot = new Vec-
tor2(0, 200);

    private Texture2D paintTexture;
    private Renderer objectRenderer;
    private Material paintingMaterial;
    private bool isPainting = false;
    private Color[] originalPixels;
    private List<Vector2> drawPoints = new List<Vector2>();

    void Start()
    {
        objectRenderer = GetComponent<Renderer>();
        if (objectRenderer == null)
        {
            Debug.LogError("TexturePainter:     Renderer     не
найден!");
            return;
        }

        paintingMaterial = objectRenderer.material;
        if    (paintingMaterial    ==    null    ||    painting-
Material.mainTexture == null)
        {
            Debug.LogError("TexturePainter: Материал не поддер-
живает '_MainTex' или текстура не назначена!");
            return;
        }

        if (!TryGetComponent(out Collider _))
        {
            Debug.LogError("TexturePainter:    Коллайдер    не
найден!");
            return;
        }

        mainCamera ??= Camera.main;
        SetDefaultCursor();
```

```
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.T))
        {
            ResetToOriginal();
        }

        if (paintTexture == null) return;

        if (Input.GetMouseButtonDown(0))
        {
            isPainting = true;
        }
        else if (Input.GetMouseButtonUp(0))
        {
            isPainting = false;
        }

        if (isPainting)
        {
            Ray ray = mainCamera.ScreenPointToRay(Input.mousePo-
sition);
            if  (Physics.Raycast(ray,  out  RaycastHit  hit)  &&
hit.transform == transform)
            {
                SetPaintCursor();
                PaintAtUV(hit.textureCoord);
                RecordDrawPoint(hit.textureCoord);
            }
            else
            {
                SetDefaultCursor();
            }
        }
    }

    void SetPaintCursor()
    {
        if (paintCursor != null)
            Cursor.SetCursor(paintCursor,  cursorHotspot,  Cur-
sorMode.Auto);
    }

   public void SetDefaultCursor()
    {
        Cursor.SetCursor(defaultCursor,  cursorHotspot,  Cursor-
Mode.Auto);
    }

    public void LoadImage(Texture2D baseImage)
    {
```

```
        Debug.Log("TexturePainter: Загружается изображение из
Pickup");

        paintTexture = new Texture2D(textureSize, textureSize,
TextureFormat.RGBA32, false);
        paintTexture.filterMode = FilterMode.Bilinear;
        paintTexture.wrapMode = TextureWrapMode.Clamp;

        if (baseImage.width != textureSize || baseImage.height
!= textureSize)
        {
            Texture2D resized = new Texture2D(textureSize, tex-
tureSize, TextureFormat.RGBA32, false);
            RenderTexture rt = RenderTexture.GetTemporary(tex-
tureSize, textureSize);
            Graphics.Blit(baseImage, rt);

            RenderTexture prev = RenderTexture.active;
            RenderTexture.active = rt;
            resized.ReadPixels(new Rect(0, 0, textureSize, tex-
tureSize), 0, 0);
            resized.Apply();
            RenderTexture.active = prev;
            RenderTexture.ReleaseTemporary(rt);

            originalPixels = resized.GetPixels();
            paintTexture.SetPixels(originalPixels);
        }
        else
        {
            originalPixels = baseImage.GetPixels();
            paintTexture.SetPixels(originalPixels);
        }

        paintTexture.Apply();
        paintingMaterial.mainTexture = paintTexture;

        Debug.Log("TexturePainter: Изображение применено и со-
хранено как оригинал");
    }

    public void ResetToOriginal()
    {
        if (paintTexture == null || originalPixels == null)
        {
            Debug.LogWarning("TexturePainter: Нет оригинала для
отката");
            return;
        }

        paintTexture.SetPixels(originalPixels);
        paintTexture.Apply();
        drawPoints.Clear();
```

```
        Debug.Log("TexturePainter: Текстура сброшена до исход-
ного изображения");
    }

    void PaintAtUV(Vector2 uv)
    {
        int x = (int)(uv.x * paintTexture.width);
        int y = (int)(uv.y * paintTexture.height);

        int radius = (int)(brushSize * paintTexture.width /
100f);

        for (int i = -radius; i <= radius; i++)
        {
            for (int j = -radius; j <= radius; j++)
            {
                if (i * i + j * j <= radius * radius)
                {
                    int pixelX = x + i;
                    int pixelY = y + j;

                    if (pixelX >= 0 && pixelX < paintTex-
ture.width &&
                        pixelY >= 0 && pixelY < paintTex-
ture.height)
                    {
                        paintTexture.SetPixel(pixelX, pixelY,
brushColor);
                    }
                }
            }
        }

        paintTexture.Apply();
    }

    void RecordDrawPoint(Vector2 uv)
    {
        if (drawPoints.Count == 0 || Vector2.Distance(draw-
Points[drawPoints.Count - 1], uv) > 0.02f)
        {
            drawPoints.Add(uv);
            Debug.Log($"TexturePainter: Recorded point UV:
{uv}");
        }
    }

    public void ClearTexture()
    {
        if (paintTexture == null) return;

        Color[] pixels = paintTexture.GetPixels();
        for (int i = 0; i < pixels.Length; i++)
```

```
        {
            pixels[i] = Color.white;
        }

        paintTexture.SetPixels(pixels);
        paintTexture.Apply();
        drawPoints.Clear();
    }

    public Texture2D GetPaintTexture()
    {
        return paintTexture;
    }

    public Color GetBrushColor()
    {
        return brushColor;
    }

    public Renderer GetRenderer()
    {
        return objectRenderer;
    }

    public List<Vector2> GetDrawPoints()
    {
        return new List<Vector2>(drawPoints);
    }
}
```