|  |  |
| --- | --- |
|  | Министерство образования Калининградской области  Государственное бюджетное учреждение Калининградской области профессиональная образовательная организация  «Колледж информационных технологий и строительства» |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Разработка компьютерного симулятора для дальнобойщиков**

Выполнил (а):обучающийся группы ИСп 20-1 Якименко Илья Сергеевич

Специальность: 09.02.07 «Информационные системы и программирование»

(код, наименование)

Руководитель ВКР:

Большакова-Стрекалова А.В.

(Ф.И.О. преподавателя)

|  |  |
| --- | --- |
| Выпускная квалификационная работа допущена к защите  « » 20 г.  Заместитель директора по учебно – методической работе  \_\_\_\_\_\_\_\_\_\_ / Павленко Г.Я.  (подпись) (Ф.И.О) | Рецензент: Бровко Павел Николаевич, Общество с ограниченной ответственностью «РЛ Авто»  (Ф.И.О., место работы)  Председатель ГЭК: Наконечный А.Н.\_\_\_\_\_\_\_\_\_  (Ф.И.О.) |

Калининград

2024 г.

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**для выполнения выпускной квалификационной работы**

Специальность 09.02.07 «Информационные системы и программирование» Группа ИСп 20-1

Ф.И.О. обучающегося Якименко Илья Сергеевич

Тема ВКР Разработка компьютерного симулятора для дальнобойщиков

Дата выдачи задания: «22» апреля 2024 г.

Работа должна быть сдана не позднее «15» июня 2024 г.

**Перечень вопросов, подлежащих разработке в ВКР**

1. Подготовка материала и оформление технического задания и последующего технического проекта по теме дипломной работы.

2. Подбор и анализ материалов для определения того нового, что будет разрабатываться в дипломной работе.

3. Анализ и проектирование структуры и дизайна симулятора.

4. Программная реализация симулятора.

5. Подготовка методического обеспечения.

6. Расчет себестоимости разработки.

Графическая часть (при наличии):

Архитектура приложения, блок-схема алгоритма, структурная схема.

Руководитель ВКР: Большакова-Стрекалова А.В.

(подпись) (Ф.И.О.)

Задание получил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Якименко И.С.

(подпись) (Ф.И.О.)

«22» апреля 2024 г.

Содержание

[Введение 4](#_Toc168580917)

[1 Анализ информации и постановка задачи 7](#_Toc168580918)

[1.1 Структура и особенности разработки симулятора дальнобойщика 7](#_Toc168580919)

[1.1.1 Структура разработки симулятора дальнобойщика 7](#_Toc168580920)

[1.1.2 Особенности разработки симулятора дальнобойщика 8](#_Toc168580921)

[1.2 Анализ существующих аналогов 9](#_Toc168580922)

[1.2.1 Euro Truck Simulator 2 9](#_Toc168580923)

[1.2.2 City Car Driving 10](#_Toc168580924)

[1.2.3 STISIM Drive 11](#_Toc168580925)

[1.3 Обзор и обоснование выбранных средств разработки 12](#_Toc168580926)

[1.3.1 Godot Engine 13](#_Toc168580927)

[1.3.2 Unreal Engine 5 13](#_Toc168580928)

[1.3.3 Unity 15](#_Toc168580929)

[1.3.4 Язык программирования C# 16](#_Toc168580930)

[1.3.5 Visual Studio 19](#_Toc168580931)

[1.3.6 Adobe Experience Design 21](#_Toc168580932)

[1.3.7 Microsoft Word 22](#_Toc168580933)

[2 Проектирование компьютерного симулятора для дальнобойщиков 24](#_Toc168580934)

[2.1 Требования к компьютерному симулятору для дальнобойщиков 24](#_Toc168580935)

[2.2 Архитектура и компоненты симулятора дальнобойщика 24](#_Toc168580936)

[2.3 UI-дизайн симулятора 25](#_Toc168580937)

[2.4 Создание диаграммы классов 29](#_Toc168580938)

[3 Разработка компьютерного симулятора для дальнобойщиков 32](#_Toc168580939)

[3.1 Разработка функционала для пользователя 32](#_Toc168580940)

[3.3 Методическое обеспечение 40](#_Toc168580941)

[3.3.1 Руководство системного администратора 40](#_Toc168580942)

[3.3.2 Руководство пользователя 42](#_Toc168580943)

[4 Экономическая часть 47](#_Toc168580944)

[4.1 Теоретическая часть 47](#_Toc168580945)

[4.2 Практическая часть 49](#_Toc168580946)

[Заключение 51](#_Toc168580947)

[Список используемых источников 52](#_Toc168580948)

[Приложение А. Листинг кода 53](#_Toc168580949)

# Введение

В эпоху глобализации и высокого темпа производства и потребления человеком различных материальных благ остро стоит вопрос транспортировки грузов на большие расстояния как между странами, так и между регионами и предприятиями одной страны. В России поставки товаров и сырья имеют ключевое значение для функционирования экономики и качества жизни населения в целом ввиду большой площади государства и внушительного количество импорта и экспорта. Для Калининградской области перевозки извне являются жизненно необходимыми, так как регион экономически сильно зависит от основной территории Российской Федерации и в тоже время территориально от нее изолирован.

Транспортировка груза на грузовых автомобилях является наиболее популярной ввиду своей доступности. Для осуществления подобной работы необходим найм квалифицированных сотрудников, так как от них зависит не только экономика страны и благополучие компаний – работодателей, но и безопасность других участников дорожного движения. Важность качественного отбора кадров для выполнения подобной работы становится еще более очевидной если брать в расчет особенности и условия автомобильной транспортировки грузов между городами и регионами в условиях нашего государства, такие как необходимость транзитного проезда в другие страны при ввозе и вывозе товаров из Калининградской области на территорию остальной России, длительное движение экономически важных продуктов между удаленными друг от друга территориальными единицами, разнообразность рельефа и качества дорожного покрытия.

Один из лучших способов проверки навыков водителя большегрузного автомобиля – это тестирование при помощи симуляции его деятельности при помощи компьютерных программ. Данный метод не требует использования реального автомобиля, обеспечивает полную безопасность участников тестирования, а также позволяет быстро проводить как саму проверку, так и получение ее результатов. Кроме того, подобное ПО (программное обеспечение) можно применять с такой же успешностью для обучения и тренировки будущих работников перед стартом карьеры, а также для повышения квалификации либо дополнительной проверки водителей со стажем.

Объектом исследования являются симуляторы вождения, разработанные для персональных компьютеров. Подобные программы как правило служат для тренировки людей, обучающихся на водительские права различных категорий, либо же просто в качестве приложения для развлечения. В рамках дипломной работы будет сделан приоритет в сторону изучения наиболее реалистичных симуляторов с точки зрения физики перемещения транспорта и соблюдения участниками дорожного движения правил.

Предмет исследования – это особенности и требования компьютерных симуляторов вождения; а также их достоинства, на которые можно опираться при разработке собственного аналога, и недостатки, причину существования которых можно определить, а сами минусы исправить или свести к минимуму. Изучение данных аспектов подобных программ в основном будет проводиться с точки зрения их практической пользы для людей, применяющих их лично, сюда можно отнести навык определения положения транспорта относительно других объектов на проезжей части, а также соблюдения правил дорожного движения (особенно в спорных и критических ситуациях, происходящих при вождении).

Также важно указать, что компьютерная программа, являющаяся конечным результатом исследования, разрабатывается для использования в транспортной компании ООО «РЛ Авто» в целях проверки профессиональных качеств и навыков водителей большегрузных автомобилей перед наймом или уже работающих сотрудников на корректность выполнения работы в рамках должности.

Ссылка на GitHub - https://github.com/YakimenkoIlya/Truck-Driver-Simulator-Repository

1. Анализ информации и постановка задачи

1.1 Структура и особенности разработки симулятора дальнобойщика

1.1.1 Структура разработки симулятора дальнобойщика

1. Анализ требований

На этом этапе проводится сбор и анализ требований к симулятору. Включает следующие шаги:

1. Исследование рынка: Определение потребностей целевой аудитории и изучение существующих решений.
2. Определение целей и задач: Формулирование целей проекта и задач, которые симулятор должен решать.
3. Составление технического задания: Документирование всех требований, включая функциональные и нефункциональные.

2. Проектирование

На этапе проектирования создаются архитектура и дизайн симулятора:

1. Архитектура системы: Определение основных компонентов системы и их взаимодействий.
2. Технический дизайн: Разработка спецификаций для каждого компонента, включая пользовательский интерфейс, физическую модель, сценарии движения и другие элементы.
3. Прототипирование: Создание прототипов ключевых элементов для тестирования и оценки.

3. Разработка

Этап разработки включает реализацию всех компонентов симулятора:

1. Программирование: Написание кода для всех модулей системы. Обычно используется объектно-ориентированное программирование (например, на C++ или C#).
2. Графический дизайн: Создание моделей транспортных средств, дорог, окружающей среды и других визуальных элементов.
3. Интеграция физического движка: Реализация физической модели для расчета движений и взаимодействий объектов.
4. Создание сценариев: Разработка сценариев движения, имитирующих реальные дорожные условия и ситуации.

4. Тестирование

Тестирование является критически важным этапом разработки:

1. Юнит-тестирование: Проверка отдельных компонентов на соответствие спецификациям.
2. Интеграционное тестирование: Проверка взаимодействия между различными компонентами.
3. Системное тестирование: Полное тестирование симулятора для выявления ошибок и недостатков.
4. Бета-тестирование: Привлечение реальных пользователей для получения обратной связи и выявления скрытых проблем.

5. Внедрение и поддержка

После завершения разработки симулятор внедряется и поддерживается:

1. Развертывание: Установка симулятора на целевых платформах (например, учебные центры, автошколы).
2. Обучение пользователей: Проведение обучения для инструкторов и пользователей по работе с симулятором.
3. Техническая поддержка: Обеспечение поддержки пользователей и решение возникающих технических проблем.
4. Обновления и улучшения: Внесение изменений и дополнений на основе обратной связи и новых требований.

1.1.2 Особенности разработки симулятора дальнобойщика

1. Мультимодульность

Разработка симулятора включает создание и интеграцию множества различных модулей: графических, физических, управленческих и сценарных. Это требует тщательной координации и обеспечения совместимости всех компонентов.

2. Реализм и детализация

Для достижения реалистичности необходимо учитывать множество факторов, таких как физика движения, поведение грузовика в различных дорожных условиях, визуальные эффекты и звуковое сопровождение. Это требует использования современных технологий и мощных вычислительных ресурсов.

3. Пользовательский опыт (UX)

Особое внимание уделяется удобству использования симулятора. Интерфейс должен быть интуитивно понятным, а управление – отзывчивым и точным. Важно также обеспечить адекватную обратную связь пользователю для эффективного обучения.

4. Образовательная ценность

Симулятор должен быть не только реалистичным, но и образовательным. Это включает разработку сценариев, которые помогут пользователям отработать навыки в условиях, приближенных к реальным. Также необходимо предусмотреть систему оценки и анализа прогресса пользователей.

5. Сотрудничество и интеграция

Разработка таких сложных систем требует тесного сотрудничества между различными специалистами: программистами, дизайнерами, инженерами и педагогами. Важно также учитывать интеграцию с другими системами и программным обеспечением, используемым в учебных центрах.

* 1. Анализ существующих аналогов

1.2.1 Euro Truck Simulator 2

Компьютерная игра в жанре симулятора водителя-дальнобойщика с элементами экономической стратегии. Разработана для персональных компьютеров Windows, macOS и Linux. Благодаря качеству проработки данная игра стала одной из самых высокоценных и узнаваемых в жанре.



Рисунок 1 – Обложка симулятора Euro Truck Simulator 2

Плюсы:

1. Реалистичная физика и моделирование поведения грузовиков.
2. Обширная карта, включающая множество городов и достопримечательностей.
3. Возможность настройки и улучшения грузовиков.
4. Поддержка модификаций, что позволяет расширить игровой опыт.

Минусы:

1. Медленное развитие игрового процесса в начале игры.
2. Монотонный процесс заездов. [1]

1.2.2 City Car Driving

Данный симулятор вождения сосредотачивается на обучении и практике навыков вождения в городских условиях. Он предоставляет реалистичные сценарии вождения в различных городах с разными дорожными условиями. Разработан для персональных компьютеров Windows. Данный симулятор является одной из популярнейших игр в своем жанре.



Рисунок 2 – Обложка симулятора City Car Driving

Плюсы:

1. Отличный инструмент для обучения правилам дорожного движения и навыкам вождения.
2. Реалистичная физика автомобилей и поведение дорожного трафика.
3. Различные режимы вождения и сценарии, включая экзаменационные маршруты.
4. Поддержка виртуальной реальности для более погруженного опыта.

Минусы:

1. Имеются проблемы с оптимизацией.
2. Иногда появляются проблемы с поведением водителей, пешеходов и прочих элементов, управляемых искусственным интеллектом. [2]

1.2.3 STISIM Drive

Программное обеспечение, специально разработанное для использования на автомобильных тренажерах. Оно имитирует различные дорожные условия и сценарии для тренировки водителей в различных ситуациях.



Рисунок 3 – Тренажер STISIM Drive

Плюсы:

1. Реалистичные симуляции дорожных условий, включая погодные условия, дорожные знаки и сценарии аварий.
2. Поддержка различных видов тренажеров, включая стационарные.
3. Может быть настроен под конкретные требования обучения и оценки.

Минусы:

1. Высокие затраты на оборудование и программное обеспечение.
2. Требует профессиональной установки и настройки.
3. Ограниченная доступность для обычных потребителей. [3]

Данные аналоги были проанализированы с целью выявления их достоинств и недостатков. В качестве основного плюса приведенных выше симуляторов вождения большегрузных автомобилей можно выделить реалистичную физику, а в качестве минуса – либо проблемы, связанные с содержанием программы (монотонный процесс заездов, поведение искусственного интеллекта), либо с невозможностью использования симулятора на обычных персональных компьютерах.

1.3 Обзор и обоснование выбранных средств разработки

Так как разрабатываемый проект должен имитировать езду на реальном большегрузном автомобиле, реализовывать программу необходимо в программной среде, поддерживающей 3D-графику, из-за чего в качестве средства разработки должен использоваться игровой движок.

1.3.1 Godot Engine



Рисунок 4 – Логотип Godot

Открытый кроссплатформенный игровой движок для разработки 2D/3D-видеоигр и приложений для ПК, мобильных устройств, веб-платформ.

Достоинства движка:

1. Сравнительно простое освоение и разработка, в том числе благодаря GDScript (собственному диалекту языка программирования Python) используемому для написания кода, а также возможности использовать такие языки, как C#, C++, C или систему VisualScript. [4]
2. Открытый исходный код.
3. Низкие системные требования.

Недостатки движка:

1. Малые возможности работы с 3D графикой.
2. Относительно небольшой набор функций.

Godot Engine лучше всего подходит для создания небольших 2D проектов, что не подходит для реализации задачи дипломной работы.

1.3.2 Unreal Engine 5

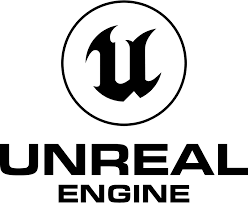


Рисунок 5 – Логотип Unreal Engine

Кроссплатформенный игровой движок, в основном применяемый в разработке AAA-игр (проектов с большим бюджетом) и неигровых проектов с высоким уровнем графики и детализации.

Достоинства движка:

1. Наличие технологии визуального программирования Blueprints, способного сократить большое количество времени на создании и настройке персонажа. [5]
2. Удобные средства для создания игровых средств и спецэффектов высокого качества.

Недостатки движка:

1. Высокая сложность интерфейса движка.
2. Высокая требовательность движка к производительности аппаратной основы ПК.
3. Медленная работа кода в процессе разработки.
4. Относительная длительность работы над проектами.

Unreal Engine 5 хорошо подходит для создания игр различных жанров под большое количество платформ, однако он имеет высокую требовательность к ресурсам ПК, высокую сложность и некоторое неудобство в работе с кодом.

1.3.3 Unity



Рисунок 6 – Логотип Unity

Кроссплатформенный игровой движок, как правило используемый для разработки AA-игр (проектов со средним бюджетом), мобильных игр, а также неигровых проектов. Позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. [6]

Движок разработан таким образом, чтобы разработка на нем придерживалась подхода модульности, а каждый объект может иметь набор собственных компонентов, из-за чего качестве языка написания скриптов используется объектно-ориентированный язык программирования C#.

В качестве интегрированной среды для разработки в Unity по умолчанию используется программа Visual Studio, которую можно установить при скачивании самого движка на компьютер.

Достоинства движка:

1. Большое количество игровых ассетов - компонентов, представляющих готовое решение, которое можно использовать в проекте (текстуры, материалы, модельки и т.д.), многие из которых доступны бесплатно.

2. Удобный и простой интерфейс.

3. Относительно низкие требования к производительности аппаратной основы ПК.

4. Большое количество учебных материалов, в том числе на русском языке.

5. Модульная система компонентов.

Недостатки движка:

1. Созданные проекты занимают относительно много памяти.

2. При создании больших проектов высока вероятность понижения производительности игры или приложения.

В качестве средства разработки был выбран игровой движок Unity версии 2022.3.7f1, поскольку он имеет удобные и эффективные средства для работы с графикой и физикой, необходимой для программной реализации симулятора дальнобойщика.

В качестве наиболее популярного языка написания скриптов в Unity используется C# (C Sharp), а в качестве интегрированной среды разработки по умолчанию – Visual Studio.

1.3.4 Язык программирования C#



Рисунок 7 – Логотип C#

C# – объектно-ориентированный язык программирования общего назначения, разработанный к началу 2000-х годов. Изначально он планировался для создания программ под Windows, но в итоге стал универсальным.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Достоинства языка:

1. Простота и удобство в использовании. Благодаря своей простой и понятной синтаксису язык C# легок для изучения и позволяет быстро разрабатывать приложения.
2. Мощная и эффективная сборка мусора. C# автоматически освобождает выделенную в памяти для объектов после того, как они не используются.
3. Объектно-ориентированное программирование. Язык C# предоставляет основные конструкции, необходимые для создания объектно-ориентированных приложений.
4. Безопасность. C# предоставляет механизмы для защиты от несанкционированных действий, таких как взлом или сбой в работе приложения.
5. Интеграция с .NET-платформой. C# разработан для совместной работы с .NET-платформой, что позволяет использовать библиотеки и компоненты, созданные на других языках .NET.
6. Поддержка параллельного программирования. Это позволяет создавать многопоточные приложения.
7. Возможности для веб-разработки. C# используется для создания веб-приложений, благодаря различным фреймворкам, таким как ASP.NET.

Недостатки языка:

1. Зависимость от платформы — C# выполняется в .NET-среде, поэтому зависит от платформы, которая может быть ограничением для некоторых разработчиков.
2. Ограниченные возможности управления памятью — в отличие от некоторых других языков, таких как С++, C# предоставляет ограниченные возможности управления памятью.
3. Сложность для разработки низкоуровневых приложений — C# не предназначен для разработки низкоуровневых приложений, из-за чего он может быть менее эффективным для некоторых задач.
4. Ограниченная поддержка функционального программирования — в C# недостаточно развита поддержка функционального программирования, в отличие от некоторых других языков, таких как Scala или Haskell.
5. Ограниченная поддержка многопоточности — хотя C# имеет поддержку многопоточности, она не настолько развита как у некоторых других языков, таких как Java или Kotlin.
6. Привязка к платформе Microsoft — поскольку C# был создан корпорацией Microsoft, он наиболее широко используется в операционной среде Windows и в экосистеме Microsoft.
7. Неудобство работы с низкоуровневым оборудованием — в некоторых случаях C# может быть неудобен для работы с низкоуровневым оборудованием, таким как микроконтроллеры. [7]

### 1.3.5 Visual Studio



Рисунок 8 – Логотип Visual Studio

Visual Studio – продукт компании Microsoft, включающий интегрированную среду разработки (IDE) программного обеспечения и ряд других инструментов для работы с кодом, написанном на большом количестве разнообразных языков. Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода.

Достоинства среды разработки:

1. Обширный набор инструментов и функций: Visual Studio предоставляет разработчику обширный выбор инструментов, включая отладчики, автозавершение кода, инструменты для профилирования приложений и многое другое. Надежная и расширяемая архитектура позволяет разрабатывать наиболее сложные и мощные программы.

2. Интеграция с платформами Microsoft: Visual Studio идеально подходит для разработки приложений под платформы Windows, Windows Mobile, .NET Framework и другие технологии Microsoft. Это значительно упрощает создание и развертывание приложений на этих платформах.

3. Отличная поддержка командной работы: Visual Studio предлагает мощные инструменты для совместной работы нескольких разработчиков над одним проектом. Встроенная система управления версиями позволяет эффективно сотрудничать, отслеживать изменения, резервировать данные и решать конфликты.

4. Большое сообщество и экосистема: Visual Studio - одна из самых популярных интегрированных сред разработки среди разработчиков по всему миру. Это создает огромное сообщество пользователей, готовых помочь разработчикам в решении проблем и предоставить свои знания. Кроме того, наличие экосистемы плагинов и расширений позволяет дополнить функциональность программы по мере необходимости.

Недостатки интегрированной среды разработки:

1. Высокие системные требования: Visual Studio является мощным и функциональным инструментом, но он также требует значительных вычислительных ресурсов. Некоторым пользователям может быть сложно установить и использовать Visual Studio на устаревших или слабых компьютерах.

2. Платность: В отличие от некоторых других сред разработки, Visual Studio является коммерческим продуктом с лицензионными сборами. На первых порах разработчикам может быть сложно оправдать расходы на покупку лицензии, особенно когда существуют бесплатные альтернативы.

3. Сложность и избыточность: Некоторые пользователи могут найти Visual Studio сложным в освоении из-за его обширного набора функций и настроек. Это может вызывать затруднения у начинающих разработчиков или тех, кто работает над простыми проектами. Кроме того, некоторые пользователи считают программу избыточно загруженной и перегруженной информацией.

4. Только для платформы Windows: Ограничением Visual Studio является то, что оно доступно только для операционных систем Windows. Если разрабатываются приложения для других платформ, таких как macOS или Linux, может потребоваться использовать другие среды разработки. [8]

1.3.6 Adobe Experience Design



Рисунок 9 – Логотоп Adobe XD

Adobe Experience Design (Adobe XD) — программа для разработки интерфейсов от Adobe Systems. Поддерживает векторную графику и веб-верстку и создает небольшие интерактивные прототипы.

Особенности Adobe Experience Design:

1. Adobe Experience Design — приложение для UI / UX дизайнеров, предоставляющее возможности как для разработки дизайна, так и для создания взаимодействия прототипа с переходами между монтажными областями.
2. Возможность создавать простые макеты и вайрфреймы с помощью мобильного приложения Adobe Comp для iOS и Android.
3. Возможность передать растровые изображения между Photoshop и Adobe Experience Design. [9]

Adobe Experience Design был выбран в качестве средства разработки дизайна, так как поддерживает продвинутые функции прототипирования и является стандартом в сфере создания дизайна и прототипов различных программных продуктов.

1.3.7 Microsoft Word



Рисунок 10 – Логотип Microsoft Word

Microsoft Word (часто — MS Word, WinWord или просто Word) — текстовый процессор, предназначенный для создания, просмотра, редактирования и форматирования текстов статей, деловых бумаг, а также иных документов, с локальным применением простейших форм таблично-матричных алгоритмов. Выпускается корпорацией Microsoft в составе пакета Microsoft Office. [10]

Особенности Microsoft Word:

* + - 1. Многочисленные инструменты для форматирования текста, включая стили, шрифты, цвета, абзацы и межстрочные интервалы.
      2. Бесшовная интеграция с другими приложениями Microsoft Office (Excel, PowerPoint, Outlook и т.д.), что облегчает работу с документами в различных форматах.
      3. Большое количество готовых шаблонов для различных типов документов, таких как резюме, отчеты, письма и приглашения.

Microsoft Word был выбран в качестве текстового редактора для написания отчета, так как он имеет весь необходимый и удобный функционал и используется практически повсеместно.

1.4 Техническое задание на разработку

Цель:

Разработать компьютерный симулятор с использованием игрового движка Unity.

Задачи:

1. Создать инструмент по созданию дорог при помощи импортируемой библиотеки сплайнов. В нем должна быть предусмотрена возможность добавления перекрестков.
2. Создать ландшафт, имитирующий неровности земной поверхности, а также путь между двумя точками, состоящий из различных участков дороги.
3. Разработать модель грузового автомобиля марки Renault с реалистичной физикой движения, включающего в себя модель внутреннего устройства кабины и панель управления.
4. Разработать события, имитирующие реальные ситуации, встречающиеся в работе водителя большегрузных автомобилей и служащие для проверки пользователя на знание ПДД и специфики профессии.
5. Подготовить понятный для пользователя интерфейс.

2 Проектирование компьютерного симулятора для дальнобойщиков

2.1 Требования к компьютерному симулятору для дальнобойщиков

Симулятор для дальнобойщиков должен соответствовать следующим требованиям:

1. Реалистичное отображение процесса вождения большегрузного автомобиля с точки зрения физики.
2. Возможность настроек характеристик транспорта, используемого в симуляции.
3. Вывод результатов о качестве прохождения водителем виртуальной трассы.
4. Наличие понятного и удобного интерфейса.
5. Своевременный и корректный отклик на действия пользователя.

## 2.2 Архитектура и компоненты симулятора дальнобойщика

Все классы в проекте подразделяются на два обширных группы:

1. Классы, созданные для создания трассы.

2. Классы, взаимодействующие напрямую с конечным пользователем.

Первая группа должна выполнять следующие функции:

1. Создание, редактирование и сохранение дорог в трехмерном пространстве.

2. Создание, редактирование и сохранение перекрестков на основе дорог, созданных в результате применения выше представленной функции.

Функционал первой группы должен быть доступен исключительно разработчику и только при непосредственной работе с проектом в движке Unity.

Вторая группа отвечает за функционал готовой программы и выполняет ее функции как симулятора, а именно:

1. Настройка транспортного грузового средства, используемого в симуляции.

2. Управление выше представленного транспортного средства.

3. Создание звуковых и визуальных эффектов, имитирующие эффекты из реального мира.

4. Оповещение пользователя о его действиях с явным разделением сообщений на негативные, нейтральные и позитивные.

5. Взаимодействие пользователя с интерфейсом и выполнение соответствующих команд, запрашиваемых пользователем.

6. Финальный подсчет баллов и определение степени готовности пользователя к использованию реального транспорта для выполнения работы водителя большегрузных автомобилей.

## 2.3 UI-дизайн симулятора

При входе в программу пользователь видит начальный экран с названием программы и кнопками с начальными функциями.



Рисунок 11 – Примерный макет начального экрана

При нажатии кнопки «Начать заезд» пользователь начинает сессию тестирования. Сессия имеет интерфейс, призванный отображать движение и текущие параметры транспортного средства, такие как скорость и передача, а также сообщать пользователю подсказки, сообщения об ошибках с его стороны, а также о пройденных этапах маршрута.



Рисунок 12 – Примерный макет сессионного экрана

Таймер начинает отсчет при преодолении пользователем этапа подготовки и останавливается при достижении финиша. Сообщения пользователю и оповещения о текущем этапе появляются только при определенных действиях пользователя, после чего исчезают через несколько секунд.

При достижении пользователем финиша появляется окно с таблицей результатов по различным аспектам вождения, финальный рейтинг и оценка, а также кнопка выхода в меню.

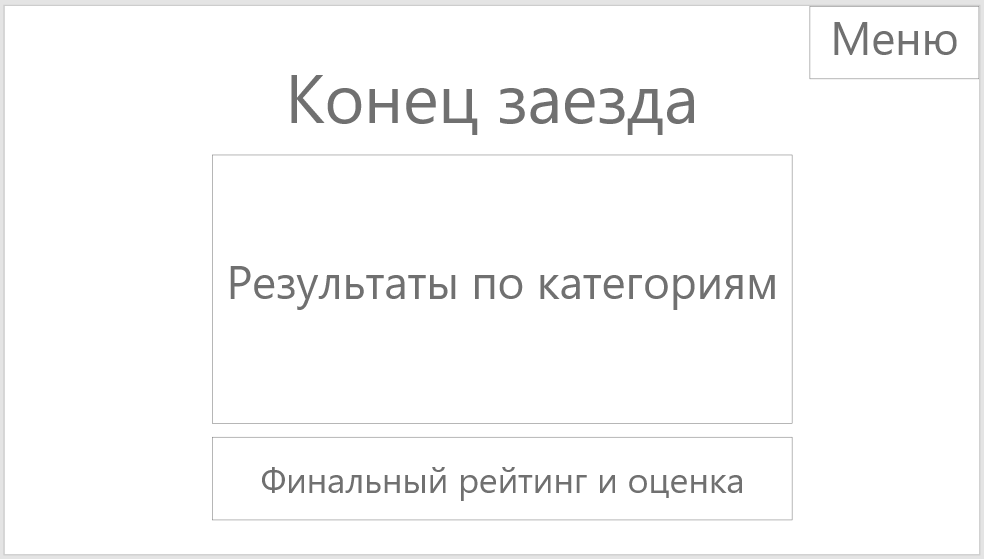


Рисунок 13 – Примерный макет экрана окончания заезда

При нажатии кнопки «Пауза» на сессионном экране пользователю выдается соответствующее окно с кнопками. Все процессы симуляции, в том числе таймер, останавливаются.



Рисунок 14 – Примерный макет экрана паузы

Нажатие кнопки «Продолжить» возобновляет сессию заезда и все её процессы, кнопка «Заново» начинает сессию с начала. «Меню» завершает сессию и отображает пользователю начальный экран без предварительного отображения экрана окончания. Кнопки «Настройки» и «Инструкция» осуществляют переход к соответствующим окнам и имеют тот же функционал, что и кнопки с этими названиями на стартовом экране.



Рисунок 15 – Примерный макет экрана настроек

Кнопка «Назад» окна настроек осуществляет переход к предыдущему экрану. На макете представлен пример настроек и их значений, который может менять человек, ответственный за тестирование прочих пользователей в зависимости от особенностей грузового транспорта, используемого в компании.

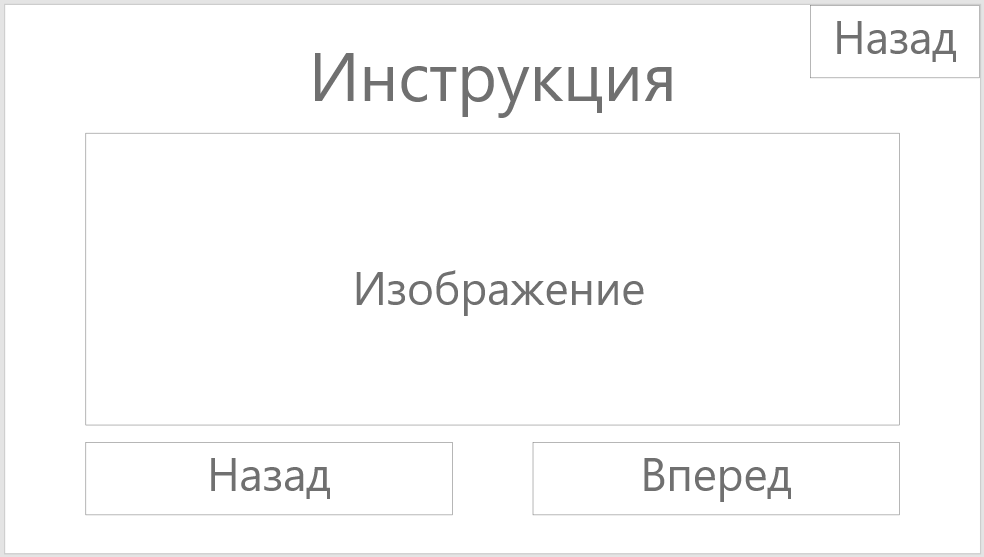


Рисунок 16 – Примерный макет экрана инструкции

Кнопка «Назад» окна инструкций также осуществляет переход к предыдущему экрану. Изображения, объясняющие управление и конечную цель сессии заезда переключаются при помощи кнопок «Вперед» и «Назад».

При нажатии кнопки «Выход» на начальном экране пользователь увидит окно подтверждения выхода с кнопками «Да» и «Нет». При нажатии первой кнопки программа закрывается, при нажатии второй – снова отображается начальный экран.

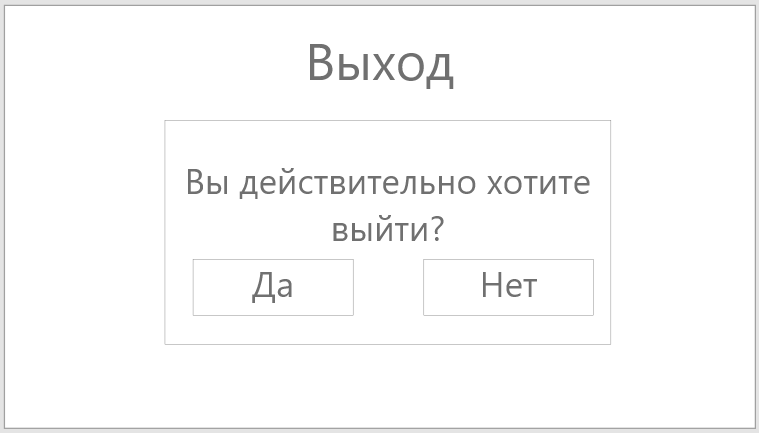


Рисунок 17 – Примерный макет экрана выхода

## 2.4 Создание диаграммы классов

Модуль для создания трассы содержит классы, отвечающие за создание дорог и перекрестков в трехмерном пространстве симулятора. Их функционал разделяется на отображение UI, видимого только в редакторе Unity при разработке программы, а также классы, одни из которых содержат информацию о физическом положении точек дороги или перекрестка, а другие отображают их и позволяют задавать различные настройки.

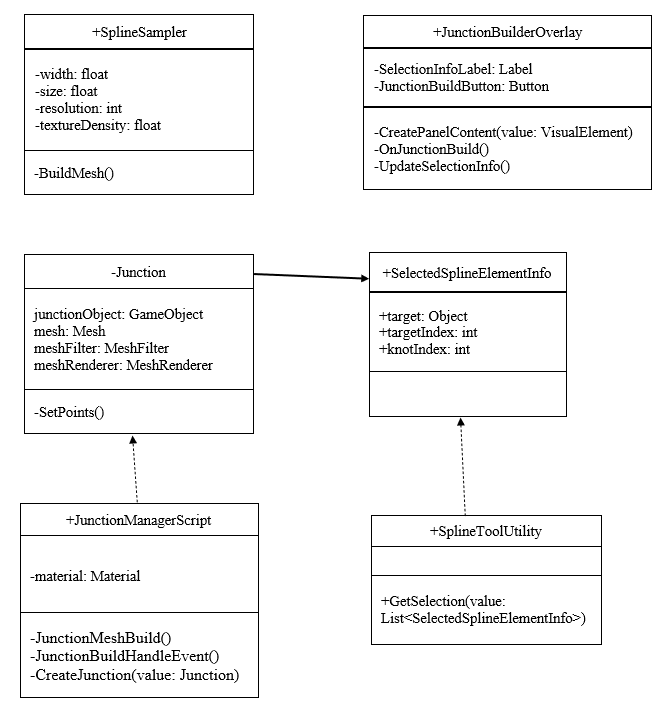


Рисунок 18 – Диаграмма классов для работы с трассой

Модуль, предназначенный для работы напрямую с пользователем, содержит классы, работающие с автомобилем (управление автомобилем; реализация его физики, а также звуковых и визуальных эффектов) и функционал, отвечающий за пользовательский интерфейс. Также в этот модуль входит класс, предназначенный для автоматического управления камерой.



Рисунок 19 – Пример классов программы для работы с пользователем

# 3 Разработка компьютерного симулятора для дальнобойщиков

## 3.1 Разработка функционала для пользователя

Из ассетов Unity была скачана модель грузового автомобиля, с которой в дальнейшем производилась вся работа.

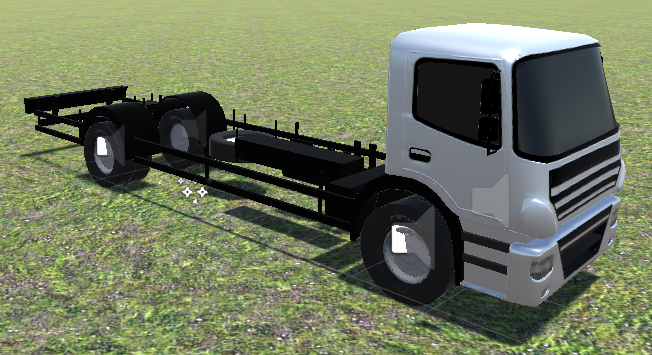


Рисунок 20 – Модель автомобиля

Физика обеспечена при помощи WheelCollider – специальных объектах Unity, имеющих необходимые параметры для настройки колес.

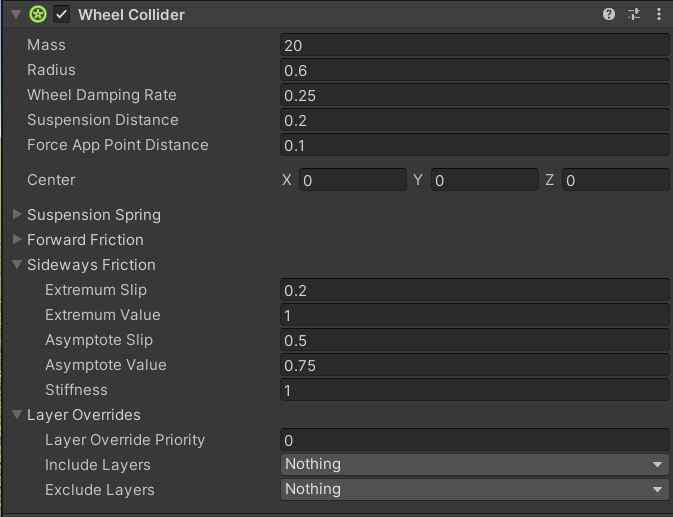


Рисунок 21 – Отображение элемента WheelCollider

Управление автомобилем было реализовано таким образом, что движении автомобиля обеспечивается нажатием клавиш W,A,S,D, а переключение передач, при помощи клавиш с цифрами на клавиатуре, совместно с некоторыми особыми клавишами, имитирующие определенные педали и рычаги, находящиеся внутри кабины реального автомобиля.

В программе было реализовано главное (стартовое) меню, которое открывается при запуске программы.



Рисунок 22 – Главное меню

При нажатии на кнопку «Начать заезд» начинается работа симуляции вождения.

Настройки открывают параметры автомобиля, которые можно менять, тем самым имитируя грузовики с разным управлением, характеристиками и физикой.



Рисунок 23 – Окно настроек

При нажатии кнопки «Инструкция» появляется окно с табличками с информацией об устройстве управления автомобилем и целям заезда, которые переключаются кнопками внизу.

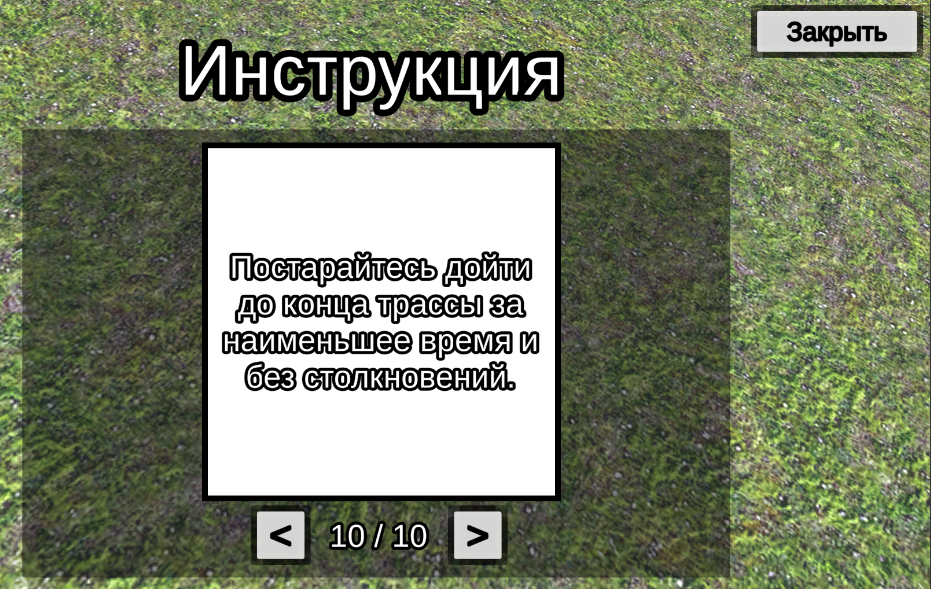


Рисунок 24 – Окно инструкций

При нажатии кнопки «Выход» появится окно с подтверждением выхода. При нажатии кнопки «да» программа будет закрыта.

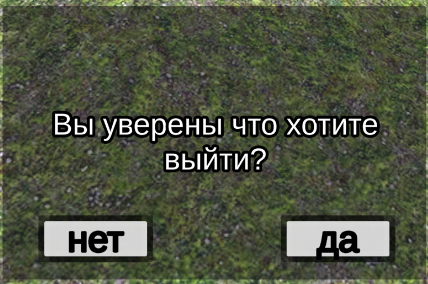


Рисунок 25 – Окно подтверждения выхода

При нажатии клавиши Escape во время проведения заезда время в симуляции будет остановлено, а также появится окно паузы.

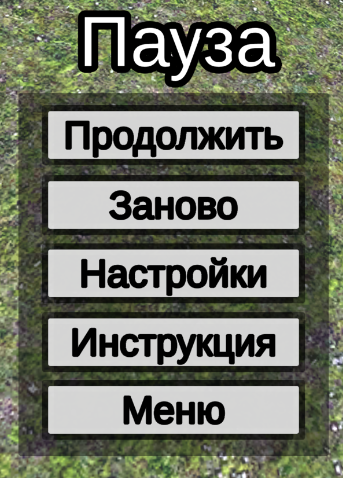


Рисунок 26 – Окно паузы

Кнопки «Продолжить» и «Заново» позволяют возобновить заезд или начать его заново. Остальные кнопки в окне идентичны кнопкам с тем же названием в стартовом меню.

Для обеспечения оповещения пользователя об ошибках во время прохождения виртуальной трассы (столкновения, выезд за проезжую часть и т.д.), прохождении этапов дороги и подсказок для лучшего результата, был реализован функционал, отображающий надписи различного содержания и внешнего вида в зависимости от действий пользователя.

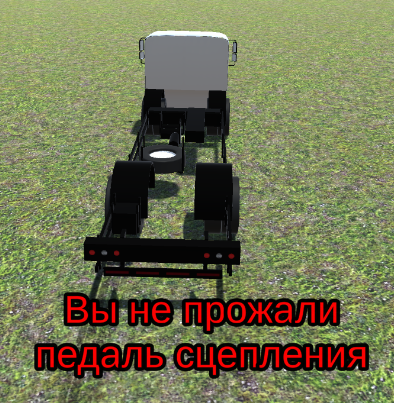


Рисунок 27 – Пример оповещения об ошибке



Рисунок 28 – Пример оповещения о прохождении этапа

Текущая передача и ее тип (повышенная или пониженная) указаны на экране и меняются в зависимости от действий пользователя.



Рисунок 29 – Отображение передачи

Одной из целей прохождения трассы является достижение финиша за наименьшее количество времени. Поэтому в программе был реализован таймер, отображающийся в верхней части экрана.

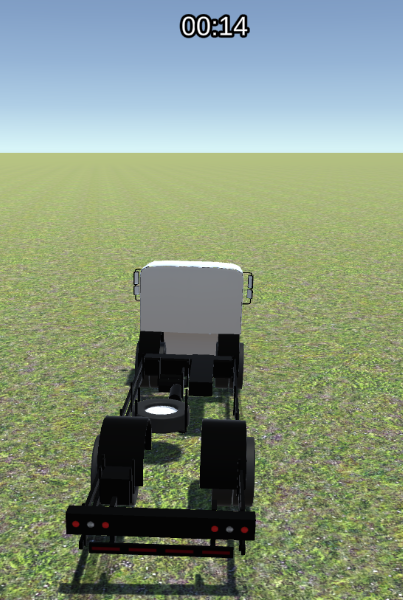


Рисунок 30 – Отображение таймера

При достижении финиша трассы на экране появляется окно с подсчетом баллов и рейтингом на их основе.

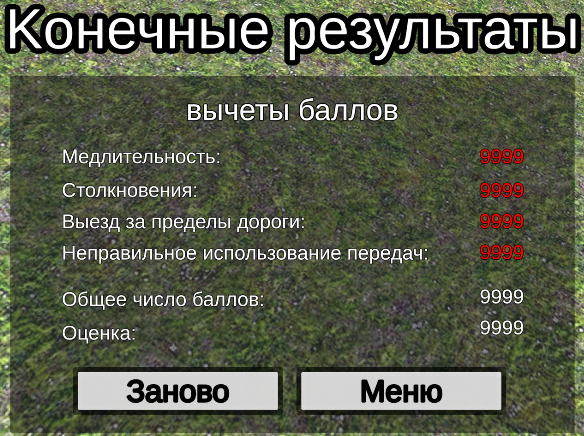


Рисунок 31 – Окно конечных результатов

## 3.2 Методическое обеспечение

3.2.1 Руководство системного администратора

1. Общие сведения о программе

Программа «Truck Driver Simulator» предназначена для обучения и проверки навыков вождения большегрузным автомобилем водителей – сотрудников предприятия ООО «РЛ Авто».

Условия применения программы:

1. Оперативная память: 2 ГБ и выше.

2. Видеокарта: GT1030 и выше или встроенная графика уровня Intel HD Graphics 610 или выше, или Vega8 и выше.

3.Операционная система: Windows 10.

2. Структура программы

Программа состоит из двух модулей:

1. Модуль для создания трассы

Отвечает за создание объектов трассы симулятора (дороги и перекрестки). Открыт к использованию и применяется только при разработке и изменению программы в движке Unity. Скрыт от обычных пользователей и напрямую не влияет на работу другого модуля программы.

2. Модуль для взаимодействия с пользователем

Напрямую отвечает за взаимодействие программы с пользователем, а именно:

1. Управление автомобилем
2. Отображение оповещений об ошибках пользователя – водителя виртуального транспорта симулятора
3. Работа интерфейса
4. Прочие действия

Взаимодействий с прочими программами симулятор не имеет.

3. Настройка программы

Откройте программу.

Нажмите кнопку «Настройки» на главном меню (рисунок 22). Откроется окно настроек (рисунок 23). В ней можно изменять различные характеристики автомобиля, чтобы приблизить их к свойствам реального транспорта, применяемого водителями при работе.

Тип коробки передач: делится на механическую и автоматическую, меняет управление автомобилем.

Масса кабины, масса прицепа, максимальная скорость: измеряются целочисленными величинами, влияют на поведение физики транспорта.

Показывать обучение: если данный параметр включен, то при нажатии кнопки главного меню «Начать заезд» будет показываться обучение по управлению виртуальным транспортом и правила проведения заезда.

Кнопки:

Сбросить: возвращает значения всех настроек на те, что указаны на рисунке23.

Отмена и Назад: закрывает окно настроек.

Применить: сохраняет измененные настройки и применяет их к программе.

4. Проверка программы

Для реализации проверки работоспособности программы, достаточно запустить ее и подождать непродолжительное время. Если симулятор работает корректно, то спустя некоторое время после его запуска на экране должна появиться заставка «Made with Unity», а затем главное меню, идентичное представленному на рисунке22.



Рисунок 32 – Заставка Unity

1. Дополнительные возможности:

Помимо возможности начинать заезд с виртуальной грузовой машиной и настраивать ее, в программе имеется инструкция, объясняющая правила проведения заезда и особенности управления автомобилем. Открыть инструкцию можно нажав соответствующую кнопку на главном меню (рисунок 22).

Выход из программы осуществляется нажатием соответствующей кнопки на главном меню и выбором варианта «да» после появления окна подтверждения выхода (рисунок 25).

3.3.2 Руководство пользователя

1. Введение

Программа «Truck Driver Simulator» используется для тестирования навыков вождения водителей большегрузных автомобилей, а также может использоваться для дополнительной подготовки водителей, объяснения и практики вождения грузовиков путем применения трехмерной компьютерной симуляции езды.

Для работы с программой требуются навыки вождения большегрузных автомобилей, если программа применяется для тестирования. В случае, если программа используется для подготовки, подобные знания не требуются.

2. Назначение и условия применения

Программа выполняет следующие функции:

1. Симуляция реальной езды на большегрузном автомобиле в трехмерном пространстве.
2. Оповещение пользователя об ошибках и событиях, вызванных им во время прохождения трассы.
3. Вывод конечных результатов и общего рейтинга качества прохождения маршрута.
4. Настройка свойств виртуального автомобиля и самой программы.

Особых условия для выполнения работы программы не требуются.

3. Подготовка к работе

Программа состоит из единого исполняемого файла Truck Driver Simulator.exe.

Установка симулятора производится скачиванием исполняемого файла на компьютер. Подключения дополнительных программ и выполнения дополнительных действий по установке не требуется.

Для реализации проверки работоспособности программы, достаточно запустить ее и подождать непродолжительное время. Если симулятор работает корректно, то спустя некоторое время после его запуска на экране должна появиться заставка Unity (рисунок 42), а затем главное меню (рисунок 22).

4. Описание операций

Проведение заезда:

При запуске программы нажмите «Начать заезд» в главном меню. Если появилось окно «Инструкция» (рисунок 24), нажмите кнопку «Назад». После чего управляете автомобилем до тех пор, пока не дойдете до конца трассы.

Если коробка передач механическая:

Управление передачами осуществляется с помощью клавиш, где 1- 8 переключаются нажатием на соответствующие клавиши с цифрами на клавиатуре, нейтральная передача – клавиши «0», а задняя – клавиши «Минус».

Перед нажатием нужной клавиши из представленных выше следует прожать клавишу «Q» отвечающую за сцепление, и не отпускать до нажатия клавиши для переключения передачи. Также нужно чтобы передача была в правильном ряду (верхнем или нижнем), который можно сменить, нажав клавишу «R». Кроме того, передачу можно сделать пониженной или обычной, прожав клавишу «E».

Если коробка передач автоматическая:

Управление передачами осуществляется с помощью клавиш с цифрами, где каждая из них равна соответствующему положению:

1. P

1. R
2. N
3. D
4. S
5. L

Перед сменой передачи нужно предварительно прожать и не отпускать тормоз и переключатель (клавиши «S» и «Q»).

Руль и управление скоростью также реализованы клавишами, где:

1. A – поворот влево.
2. D – поворот вправо.
3. W – газ.
4. S – тормоз.

Если вверху экрана работает таймер (рисунок 40), значит идет время, потраченное на заезд. Чем оно меньше, тем лучше результат.

Сообщения о действиях пользователя, такие как ошибки и прохождения этапов, отображаются внизу экрана и вверху экрана (рисунки 27 и 28).

Положение коробки передач отображается на экране справа (рисунок 29).

При нажатии клавиши Escape во время проведения заезда время в симуляции будет остановлено, а также появится окно паузы (рисунок 26). Кнопки «Продолжить» и «Заново» позволяют возобновить заезд или начать его заново. Остальные кнопки в окне идентичны кнопкам с тем же названием в главном меню (рисунок 22).

При нажатии на кнопку «Начать заезд» в главном меню, начинается работа симуляции вождения.

Настройки открывают параметры автомобиля, которые можно менять, тем самым имитируя грузовики с разным управлением, характеристиками и физикой (рисунок 23).

Тип коробки передач: делится на механическую и автоматическую, меняет управление автомобилем.

Масса кабины, масса прицепа, максимальная скорость: измеряются целочисленными величинами, влияют на поведение физики транспорта.

Показывать обучение: если данный параметр включен, то при нажатии кнопки главного меню «Начать заезд» будет показываться обучение по управлению виртуальным транспортом и правила проведения заезда.

Кнопки:

Сбросить: возвращает значения всех настроек на те, что указаны на рисунке23.

Отмена и Назад: закрывает окно настроек.

Применить: сохраняет измененные настройки и применяет их к программе.

При нажатии кнопки «Инструкция» на главном меню появляется окно с табличками с информацией об устройстве управления автомобилем и целям заезда, которые переключаются кнопками внизу.

При нажатии кнопки «Выход» (рисунок 25) появится окно с подтверждением выхода. При нажатии кнопки «да» программа будет закрыта.

При достижении финиша трассы на экране появляется окно с подсчетом баллов и рейтингом на их основе (рисунок 41).

Кнопка «Заново» начинает прохождение трассы с начала. А кнопка «Меню» выводит главное меню.

5. Аварийные ситуации

Предупреждения об аварийных ситуациях не предусмотрены, так как возможности пользователя ограничены так, чтобы вызов таких ситуаций от его действий во время использования программы был исключён.

6. Рекомендации по освоению

Материалы по использованию программы, помимо руководств системного администратора и пользователя, отсутствуют.

# 4 Экономическая часть

## 4.1 Теоретическая часть

При оценке экономической эффективности разработки программного обеспечения следует сравнивать затраты на выполнение задачи вручную с затратами на её автоматизацию. Если разрабатываемое решение заменяет уже существующее или является модификацией, необходимо сопоставить затраты на создание и эксплуатацию старого и нового решения.

Для определения годового экономического эффекта от сокращения ручного труда при обработке информации используется следующая последовательность действий. Единовременные капитальные затраты представляют собой стоимость программного продукта (ПП), которая может быть оптовой и отпускной. Все финансовые расчёты между покупателем и продавцом, включая программные продукты, ведутся на основе отпускных цен. [11]

Оптовая цена разработки определяется трудоёмкостью создания программного продукта.

Классификация затрат

Затраты на производство и реализацию продукции можно классифицировать по нескольким признакам:

По роли в производственном процессе: основные и накладные затраты. Основные затраты непосредственно формируют создаваемый продукт и включают сырьё, материалы и заработную плату. Накладные затраты связаны с обслуживанием процесса производства, например, содержание оборудования и персонала.

По способу включения в себестоимость продукции: прямые и косвенные затраты. Прямые затраты относятся на себестоимость единицы продукции, включая сырьё и заработную плату. Косвенные расходы объединяются по признакам и распределяются по группам продукции пропорционально, например, на основе прямых затрат или заработной платы основных производственных рабочих.

По зависимости от объёма выпуска продукции: постоянные и переменные затраты. Постоянные затраты остаются неизменными при изменении объёма производства, тогда как переменные изменяются пропорционально объёму выпуска.

По срокам использования в производстве: текущие и единовременные затраты. Текущие затраты осуществляются регулярно, тогда как единовременные происходят реже одного раза в месяц.

По методам планирования, учёта и распределения: затраты по экономическим элементам и по месту их осуществления (статьи калькуляции).

Группировка затрат по экономическим элементам

Амортизация на полное восстановление включает в себестоимость продукции в суммах, определяемых на основе балансовой стоимости фондов и действующих норм амортизационных отчислений. Износ начисляется как на собственные фонды, так и арендованные, а также на стоимости помещений, предоставляемых бесплатно предприятиям общественного питания и медицинского обслуживания трудовых коллективов предприятия.

Затраты по экономическим элементам собираются в смете на производство и реализацию продукции. Смета включает общий объём ресурсов, используемых в производственном процессе, и позволяет определить производственную себестоимость продукции, вычитая затраты, не связанные с производством. Эта классификация важна для предприятия, так как позволяет определить общий объём потребляемых ресурсов и спланировать бизнес-процессы, включая материально-техническое снабжение и потребность в оборотных средствах.

Группировка затрат по статьям калькуляции

Классификация затрат по статьям калькуляции помогает определить себестоимость единицы продукции, распределить затраты по ассортиментным группам и выявить резервы снижения затрат. Эта классификация лежит в основе построения плана счетов бухгалтерского учёта и формирования отчётности. Затраты группируются по направлениям их использования, таким как процесс изготовления продукции, обслуживание производства и управление предприятием.

Пример группировки затрат по статьям калькуляции:

1. Материалы.
2. Основная заработная плата программиста.
3. Дополнительная заработная плата программиста.
4. Отчисления на социальные нужды от заработной платы программиста.
5. Амортизация компьютера.
6. Амортизация программного обеспечения.
7. Накладные расходы.

Расчёт себестоимости единицы продукции или работ осуществляется путём калькулирования по установленным статьям затрат.

## 4.2 Практическая часть

Таблица 1 – Технико-экономические показатели проекта

|  |  |  |
| --- | --- | --- |
| **Наименование показателя** | **Варианты** | |
| **Базовый** | **Проектный** |
| 1. Трудоемкость решения задачи, час | 4 | 0,6 |
| 2. Периодичность решения задачи, раз в год | 511 | 511 |
| 3. Годовые текущие затраты, связанные с решением задачи, руб | 0 | 0 |
| 4. Отпускная цена программы, руб |  | 200 000 |
| 5. Степень новизны программы |  | В |
| 6. Группа сложности алгоритма |  | 2 |
| 7. Прирост условной прибыли, руб./ год |  | 130 000 |
| 8. Годовой экономический эффект пользователя, руб |  | 230 000 |
| 9. Срок возврата инвестиций, лет |  | 1,53 |

Разработанное программное обеспечение к дипломному проекту на тему «Разработка компьютерного симулятора для дальнобойщиков» обеспечивает получение годового экономического эффекта в сумме 230 000 руб. при отпускной цене программы 200 000 руб. Проект обеспечивает возврат инвестиций за 1,53 лет.

Данный программный продукт рентабельный и конкурентоспособный с точки зрения экономики и может быть использован для обучения и проверки навыков управления большегрузным автомобилем водителей – сотрудников предприятия ООО «РЛ Авто».

# Заключение

За время работы над дипломным проектом были разработан компьютерный симулятор для дальнобойщиков, состоящий из модуля, предназначенного для разработки виртуального ландшафта, а также функционала, отвечающего за взаимодействие с пользователем и реализующего основной функционал программы, имеющий фактическую ценность для места прохождения практики – организации ООО «РЛ Авто».

Весь функционал, как созданный для разработки, так и взаимодействующий напрямую с конечным пользователем работает исправно и предоставляет широкие возможности для настройки и изменения под нужды пользователя. Управление ими является интуитивно понятным и простым, а также экономно расходуют ресурсы компьютера и не вызывают проблем с производительностью.

Все элементы программы соответствуют своим изначальным требованиям и являются корректно работающими частями программных модулей. Все определенные задачи по их созданию были успешно выполнены.

# Список используемых источников

1. Euro Truck Simulator 2: Обзор - [Электронный ресурс]. – Режим доступа: - <https://stopgame.ru/show/46730/euro_truck_simulator_2_review>
2. City Car Driving (обзор) - [Электронный ресурс]. – Режим доступа: - <https://gamehag.com/ru/%d1%81%d1%82%d0%b0%d1%82%d1%8c%d0%b8/city-car-driving->
3. STISIM Drive - [Электронный ресурс]. – Режим доступа: - <https://www.systemstech.com/simulation-products/sti-sim-drive/>
4. Godot Engine - skillfactory media [Электронный ресурс]. – Режим доступа: - <https://blog.skillfactory.ru/glossary/godot-engine/>
5. Что такое Unreal Engine: история развития, преимущества и ключевые функции - skillbox media [Электронный ресурс]. – Режим доступа: <https://skillbox.ru/media/gamedev/chto-takoe-unreal-engine-istoriya-razvitiya-preimushchestva-i-klyuchevye-funktsii/>
6. Unity (игровой движок) - [Электронный ресурс]. – Режим доступа: - <https://w.wiki/4wcE>
7. Язык программирования С#: где используют и чем хорош - [Электронный ресурс]. – Режим доступа: - <https://www.tgu-dpo.ru/news/2023/04/06/yazyk-programmirovaniya-s/>
8. Microsoft Visual Studio - [Электронный ресурс]. – Режим доступа: - <https://w.wiki/A6gv>
9. Adobe Experience Design - [Электронный ресурс]. – Режим доступа: - <https://w.wiki/AHH$>
10. Microsoft Word - [Электронный ресурс]. – Режим доступа: - <https://w.wiki/9aEa>
11. Экономика: Учебное пособие - [Электронный ресурс]. – Режим доступа: - <https://reader.lanbook.com/book/369899#1>

# Приложение А. Листинг кода

Класс SplineSampler:

using System.Collections.Generic;

using Unity.Mathematics;

using UnityEngine;

using UnityEngine.Splines;

using UnityEditor;

[ExecuteInEditMode] //Назначение работы кода в режиме редактирования Unity

[RequireComponent(typeof(MeshRenderer), typeof(MeshFilter))] //Автоматическое создание элементов MeshRenderer

//и MeshFilter при добавлении данного скрипта на объект

//Класс создания дорог

public class SplineSampler : MonoBehaviour

{

private SplineContainer m\_splineContainer; //Контейнер для сплайна

[SerializeField] [Range(0f, 3f)] private float width = 0.5f; //Расстояние вершин полигонов по обе стороны от сплайна

[SerializeField] [Range(0f, 1f)] private float size = 0.2f; //Размер элементов Gizmos

[SerializeField] [Range(0, 1000)] private int resolution = 30; //Количество вершин полигонов по обе стороны сплайна

[SerializeField] [Range(1, 10)] private float textureDensity = 1f; //Сжатость материала

private List<Vector3> vertsP1; //Список вершин полигонов справа от сплайна

private List<Vector3> vertsP2; //Список вершин полигонов слева от сплайна

private Mesh mesh; //Меш сплайна для отображение графики дороги на его основе

private float3 position; //Позиция вершины полигона

private float3 tangent; //Касательная к сплайну

private float3 upVector; //Вспомогательный вектор для нахождения крайней точки создающейся дороги

public float Width => width; //Свойство переменной width

//Метод, срабатывающий при инициализации скрипта на объекте. Отвечает за приравнивае SplineContainer

//с этим скриптом к splineContainer и подписке на событие OnUpdate

private void OnEnable()

{

m\_splineContainer = GetComponent<SplineContainer>();

EditorApplication.update += OnUpdate;

}

//Метод, срабатывающий при отключении скрипта на объекте. Отвечает за отписку от события OnUpdate

private void OnDisable()

{

EditorApplication.update -= OnUpdate;

}

//Метод, отвечающий за постоянную перерисовку меша на объекте и изменение положения вершин полигонов

private void OnUpdate()

{

GetVerts();

BuildMesh();

EditorApplication.QueuePlayerLoopUpdate();

}

//Метод, срабатывающий перед первым кадром, отвечает за инициализацию меша на объекте

void Start()

{

mesh = new Mesh();

MeshFilter meshFilter = GetComponent<MeshFilter>();

meshFilter.mesh = mesh;

}

//Покадровая отрисовка элементов Gizmos для отображения вершин полигонов. Применяется для отладки

private void OnDrawGizmos() {

foreach (Vector3 position in vertsP1)

{

Handles.SphereHandleCap(0, position, Quaternion.identity, size, EventType.Repaint);

}

Handles.color = Color.black;

foreach (Vector3 position in vertsP2)

{

Handles.SphereHandleCap(0, position, Quaternion.identity, size, EventType.Repaint);

}

}

//Получение двух точек по обе стороны от сплайна

public void SampleSplineWidth(int m\_splineIndex, float time, out Vector3 p1, out Vector3 p2)

{

m\_splineContainer.Evaluate(m\_splineIndex, time, out position, out tangent, out upVector);

float3 right = Vector3.Cross(tangent, upVector).normalized;

p1 = position + (right \* width);

p2 = position + (-right \* width);

}

//Нахождение позиций вершин полигонов

private void GetVerts()

{

vertsP1 = new List<Vector3>();

vertsP2 = new List<Vector3>();

float step = 1f / resolution;

for (int i = 0; i <= resolution; i++)

{

float time = step \* i;

SampleSplineWidth(0, time, out Vector3 p1, out Vector3 p2);

AddingUniquenessPositionSpheres(p1, p2);

}

}

//Занесение точек по сторонам от сплайна в соответствующие списки

private void AddingUniquenessPositionSpheres(Vector3 positionA, Vector3 positionB)

{

vertsP1.Add(positionA);

vertsP2.Add(positionB);

}

//Построение сетки меша для придания ему необходиой формы и UV-координат для текстурирования

private void BuildMesh()

{

List<Vector3> verts = new List<Vector3>();

List<int> tris = new List<int>();

List<Vector2> uvs = new List<Vector2>();

int offset = 0;

float uvOffset = 0f;

int length = vertsP2.Count;

for (int i = 1; i <= length; i++)

{

Vector3 p1 = vertsP1[i - 1];

Vector3 p2 = vertsP2[i - 1];

if (i != length)

{

Vector3 p3 = vertsP1[i];

Vector3 p4 = vertsP2[i];

offset = 4 \* (i - 1);

int t1 = offset + 0;

int t2 = offset + 1;

int t3 = offset + 2;

int t4 = offset + 1;

int t5 = offset + 3;

int t6 = offset + 2;

approachToSpline(ref p1);

approachToSpline(ref p2);

approachToSpline(ref p3);

approachToSpline(ref p4);

verts.AddRange(new List<Vector3> { p1, p2, p3, p4 });

tris.AddRange(new List<int> { t6, t5, t4, t3, t2, t1 });

float distance = Vector3.Distance(p1, p3) \* textureDensity;

float uvDistance = uvOffset + distance;

uvs.AddRange(new List<Vector2> { new Vector2(0, uvOffset), new Vector2(1, uvOffset),

new Vector2(0, uvDistance), new Vector2(1, uvDistance) });

uvOffset += distance;

}

}

mesh.SetVertices(verts);

mesh.SetTriangles(tris, 0);

mesh.SetUVs(0, uvs);

}

//Назначение позициям вершин полигонов координат в мировом пространстве

private void approachToSpline(ref Vector3 position)

{

position -= transform.position - new Vector3(0, 0.001f, 0);

}

}

Класс Notification:

using System.Collections;

using TMPro;

using UnityEngine;

// Класс оповещений пользователя

public class Notification : MonoBehaviour

{

[SerializeField] private AnimationCurve brightnessAnimationCurve; // Анимационная кривая яркости надписи

[SerializeField] private TextMeshProUGUI simpleAndErrorNotificationTMP; // Текстовое поле для подсказок и сообщений об ошибках

[SerializeField] private TextMeshProUGUI chapterNotificationTMP; // Текстовое поле для сообщений о предолении этапов трассы

[SerializeField] private TextMeshProUGUI subChapterNotificationTMP; // Текстовое поле для дополнительных сообщений при предолении этапов трассы

private Coroutine brightnessAnimationCoroutine; // Сопрограмма отображения сообщения

private float totalTime; // Время отображение сообщения

private Color subInitialColor; // Цвет текста в таблице дополнительных сообщений

// Варианты сообщений

public enum MessageVariant

{

IncorrectShiftingIntoReverse,

Clash,

SharpGearChanges,

ShiftingGearsWithoutShifter,

StartPlace

}

// Типы сообщений

internal enum MessageType

{

Simple,

Error,

Chapter

}

// Метод, срабатывающий при появлении объекта с этим скриптом

private void Awake()

{

totalTime = brightnessAnimationCurve.keys[brightnessAnimationCurve.keys.Length - 1].time;

subInitialColor = subChapterNotificationTMP.color;

}

// Вызов сообщения

public void ShowMessage(MessageVariant messageVariant)

{

if (brightnessAnimationCoroutine != null)

StopCoroutine(brightnessAnimationCoroutine);

MessageType messageType = MessageType.Simple;

string messageString = "";

string subMessageString = "";

TextMeshProUGUI textMeshProUGUI = null;

TextMeshProUGUI subTextMeshProUGUI = null;

switch (messageVariant)

{

case MessageVariant.SharpGearChanges:

messageType = MessageType.Error;

messageString = "Вы не можете переключать более 2-х передач за раз";

break;

case MessageVariant.Clash:

messageType = MessageType.Error;

messageString = "Столкновение";

break;

case MessageVariant.StartPlace:

messageType = MessageType.Chapter;

messageString = "Этап 1-й\nСтартовая площадка";

subMessageString = "Изучите и привыкните к управлению";

break;

}

switch (messageType)

{

case MessageType.Error:

textMeshProUGUI = simpleAndErrorNotificationTMP;

textMeshProUGUI.color = Color.red;

subTextMeshProUGUI = null;

break;

case MessageType.Simple:

textMeshProUGUI = simpleAndErrorNotificationTMP;

textMeshProUGUI.color = Color.white;

subTextMeshProUGUI = null;

break;

case MessageType.Chapter:

textMeshProUGUI = chapterNotificationTMP;

subTextMeshProUGUI = subChapterNotificationTMP;

break;

}

textMeshProUGUI.gameObject.SetActive(true);

textMeshProUGUI.enabled = true;

textMeshProUGUI.text = messageString;

if(subTextMeshProUGUI != null)

{

subTextMeshProUGUI.gameObject.SetActive(true);

subTextMeshProUGUI.enabled = true;

subTextMeshProUGUI.text = subMessageString;

}

brightnessAnimationCoroutine = StartCoroutine(BrightnessAnimationCoroutine(textMeshProUGUI, subTextMeshProUGUI));

}

// Отображение сообщения

private IEnumerator BrightnessAnimationCoroutine(TextMeshProUGUI notificationTMP, TextMeshProUGUI subNotificationTMP)

{

Color initialColor = notificationTMP.color;

float startTime = Time.time;

while (Time.time - startTime < totalTime)

{

float elapsedTime = Time.time - startTime;

float brightness = brightnessAnimationCurve.Evaluate(elapsedTime);

notificationTMP.color = new Color(initialColor.r, initialColor.g, initialColor.b, brightness);

if (subNotificationTMP != null) subNotificationTMP.color = new Color(subInitialColor.r, subInitialColor.g, subInitialColor.b, brightness);

yield return null;

}

notificationTMP.enabled = false;

if (subNotificationTMP != null) subNotificationTMP.enabled = false;

}

}

Класс TimerDisplay:

using System.Collections;

using UnityEngine;

using TMPro;

// Класс тайсера

public class TimeDisplay : MonoBehaviour

{

private TextMeshProUGUI m\_TimerText; // Текстовое поле таймера

private static float m\_ElapsedTime; // Время таймера

private static bool m\_IsTiming = true; // Статус таймера

// Метод, срабатывающий при появлении объекта с данным скриптом

private void Awake()

{

m\_TimerText = GetComponent<TextMeshProUGUI>();

}

// Метод, срабатывающий каждый кадр

private void Update()

{

UpdateTimeDisplay();

}

// Отображение пройденного времени на таймере

private void UpdateTimeDisplay()

{

if (!m\_IsTiming) return;

m\_ElapsedTime += Time.deltaTime;

int minutes = Mathf.FloorToInt(m\_ElapsedTime / 60F);

int seconds = Mathf.FloorToInt(m\_ElapsedTime % 60F);

m\_TimerText.text = string.Format("{0:00}:{1:00}", minutes, seconds);

}

// Назначение статуса таймера

public static void SetTimingStatus(bool status)

{

m\_IsTiming = status;

}

// Сброс времени таймера

public static void ResetTiming()

{

m\_ElapsedTime = 0f;

}

}

Класс GearStatusUI:  
using System.Collections;

using System.Collections.Generic;

using System.Text;

using TMPro;

using UnityEngine;

using UnityStandardAssets.Vehicles.Car;

// Класс отображения статуса коробки передач

public class GearStatusUI : MonoBehaviour

{

[SerializeField] private GameObject m\_Car; // Объект автомобиля

private TextMeshProUGUI m\_GearStatusTMP; // Текстовое поле коробки передач

private StringBuilder m\_GearStatusStringBuilder = new StringBuilder(); // Тест поля коробки передач

private CarUserControl m\_CarUserControl; // Класс управления автомобилем

// Подписка на событие вызова этого класса из класса CarUserControl

private void OnEnable()

{

CarUserControl.OnGearStatusEvent += SetGearStatusText;

}

// Отписка от события вызова этого класса из класса CarUserControl

private void OnDisable()

{

CarUserControl.OnGearStatusEvent -= SetGearStatusText;

}

// Метод, срабатывающий при появлении объекта с данным скриптом

private void Awake()

{

m\_CarUserControl = m\_Car.GetComponent<CarUserControl>();

m\_GearStatusTMP = GetComponent<TextMeshProUGUI>();

}

// Создание теста статуса коробки передач

private void SetGearStatusText()

{

m\_GearStatusStringBuilder.Clear();

m\_GearStatusStringBuilder.Append("Передача: ");

int carGear = m\_CarUserControl.CurrentGear;

CarUserControl.GearType gearType = m\_CarUserControl.CurrentGearType;

switch (carGear)

{

case -1:

m\_GearStatusStringBuilder.Append("Задняя");

break;

case 0:

m\_GearStatusStringBuilder.Append("Нейтральная");

break;

default:

m\_GearStatusStringBuilder.Append($"{carGear} - ая");

break;

}

if (gearType == CarUserControl.GearType.Reduced) m\_GearStatusStringBuilder.Append(" пониженная");

m\_GearStatusTMP.text = m\_GearStatusStringBuilder.ToString();

}

}