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

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

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___» _____ 2024 г.

МОДЕЛИРОВАНИЕ ХОДОВОЙ СИСТЕМЫ БУЛЬДОЗЕРА

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.492 ПЗ ВКР

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

Автор работы,
студентка группы КЭ-405
_____А.А. Протасова
«___» _____2024 г.

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

Челябинск-2024

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

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

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студентке группы КЭ-405
Протасовой Анастасии Андреевны
обучающейся по направлению
09.03.01 «Информатика и вычислительная техника»

1. Тема работы: «Моделирование ходовой системы бульдозера» утверждена приказом по университету от 22.04.2024 г. № 764-13/12

2. Срок сдачи студенткой законченной работы: 01 июня 2024 г.

3. Исходные данные к работе:

В качестве реальной модели используется бульдозер D12 с характеристиками:

а) масса 19 тонн;

б) гидростатическая трансмиссия;

в) двигатель ЯМЗ-238;

г) мощность 240 л.с.

3D модель бульдозера в формате FBX и её составные части в формате OBJ.

4. Перечень подлежащих разработке вопросов:

1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.

2. Определение требований и выбор средств реализации проекта.
3. Проектирование и реализация модели ходовой системы бульдозера.
4. Тестирование работы модели.

Дата выдачи задания: 1 декабря 2023 г.

Руководитель работы _____/ Ю.Г. Плаксина/

Студентка _____/ А.А. Протасова /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы	26.02.2024	
Определение требований и выбор средств реализации проекта	03.03.2024	
Проектирование и реализация модели ходовой системы бульдозера	24.03.2024	
Тестирование работы модели	07.04.2024	
Подготовка презентации и доклада	15.05.2024	

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

Студентка _____/ *А.А. Протасова* /

АННОТАЦИЯ

А.А. Протасова. Моделирование ходовой системы бульдозера. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 57 с., 16 ил., библиогр. список – 21 наим.

В рамках выпускной квалификационной работы проводится проектирование и реализация модели ходовой системы бульдозера. Проведен аналитический обзор по тематике работы, а также рассмотрены аналогичные решения моделирования движения бульдозера и составлена сравнительная таблица. Выявлены преимущества и недостатки существующих решений. Описан выбор средств реализации проекта и среды разработки, определены требования к модели бульдозера. Разработана архитектура проекта и блок-схема процесса моделирования движения бульдозера, описаны составные компоненты бульдозера, участвующие в моделировании. Результатом выпускной квалификационной работы является создание компьютерной модели, которая будет имитировать поведение ходовой системы бульдозера.

Оглавление

ВВЕДНИЕ 7

1. АНАЛИТИЧЕСКИЙ ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ, НОРМАТИВНОЙ И МЕТОДИЧЕСКОЙ ЛИТЕРАТУРЫ ПО ТЕМАТИКЕ РАБОТЫ 9

1.1. Обзор аналогов 11

1.2. Вывод 15

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ И ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ ПРОЕКТА 16

2.1. Функциональные требования 16

2.3. Выбор средств реализации 16

2.4. Выбор среды разработки 21

3. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ МОДЕЛИ ХОДОВОЙ СИСТЕМЫ БУЛЬДОЗЕРА 24

3.2. Класс ссылок (англ. link) 29

3.3. Система движения и расчет трансмиссии 32

3.4. Визуализация процесса 34

4. ТЕСТИРОВАНИЕ РАБОТЫ МОДЕЛИ 37

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 41

ПРИЛОЖЕНИЯ 44

ПРИЛОЖЕНИЕ А 44

ПРИЛОЖЕНИЕ Б 46

ПРИЛОЖЕНИЕ В 49

ВВЕДНИЕ

В современном мире, где технологии развиваются с невероятной скоростью, одной из основных задач является повышение эффективности и производительности труда. Важная роль в этом процессе отводится машинам, в частности специализированным машинам, таким как бульдозеры. Бульдозеры являются незаменимыми помощниками на строительных площадках, карьерах и других объектах, где требуется перемещение грунта. Однако, чтобы повысить эффективность работы таких машин, необходимо разработать и внедрить новые методы и подходы к их проектированию и эксплуатации.

Одним из таких подходов является компьютерное моделирование ходовой системы бульдозеров, которое позволяет изучить основные характеристики и особенности работы этой системы, а также выявить возможные недостатки и предложить пути их устранения. К основным элементам ходовой системы бульдозера относятся опорные и поддерживающие катки, направляющее колесо, звёздочка, механизм натяжения и гусеничная лента, состоящая из траков. Ходовая часть бульдозера работает в постоянном контакте с опорной поверхностью, поэтому подвержена наибольшим нагрузкам при работе. Моделирование ходовой системы бульдозера позволяет провести анализ нагрузок, действующих на машину во время работы, определить оптимальные параметры и тем самым снизить затраты на техническое обслуживание и ремонт.

Темой выпускной квалификационной работы является моделирование ходовой системы бульдозера со всеми физическими характеристиками, для того, чтобы получить информацию о нагрузках, о подходящих параметрах для наиболее эффективного использования техники.

Цель выпускной квалификационной работы заключается в создании компьютерной модели, которая будет имитировать поведение ходовой системы бульдозера в различных режимах работы (покоя и рабочий режим).

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести аналитический обзор научно-технической, нормативной и методической литературы по тематике работы и обзор аналогов.
2. Проанализировать и выбрать средства реализации проекта.
3. Спроектировать и реализовать модель ходовой системы гусеничного бульдозера.
4. Протестировать работу модели.

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

Стремление к увеличению конкурентоспособности техники заставляет производителей повышать качество изделий. С одной стороны, требуется увеличение прочности конструкций, их надежности, долговечности, с другой стороны, улучшение качества материалов и увеличение массы изделий приводит к увеличению себестоимости и потере конкурентоспособности. Одним из путей является повышение качества расчетов для уточнения нагрузок действующих на машину, систему или узел. Решение таких задач достижимо на стадии функционального проектирования, когда возможен выбор параметров машины, наиболее полно удовлетворяющих установленным целевым функциям [1]. Моделирование сложных систем является неотъемлемой частью процесса автоматизации проектирования. В статье [2] рассмотрена система автоматизации моделирования рабочего процесса на примере гусеничного трактора. В источнике [3] рассматривается актуальность исследования гусеничных машин при помощи современных средств компьютерного моделирования, позволяющих проводить анализ кинематики и динамики движения. Приведены примеры программ для такого анализа MSC ADAMS View, ADAMS Tracked Vehicle (ATV) Toolkit и UM Tracked Vehicles (Универсальный механизм) и т.д. [4].

Для моделирования поведения техники, в нашем случае гусеничного бульдозера, необходимо использовать физические законы, так как они позволяют описать и предсказать движение и взаимодействие объектов с учётом их массы, силы тяжести, инерции и других факторов. В общепринятом понимании, физику в виртуальном мире называют “физический движок” (англ. physics engine) – программное обеспечение для симуляции физических законов реального мира. В источнике [5] автор приводит описание двух видов движков: научные и игровые. Научные движки используются в симуляциях и расчетах, где важна именно точность вычислений. Физический движок обычно не является самостоятельной программой, а представляет собой

библиотеку, которую можно интегрировать в прикладной проект. Это позволяет применять физическое движение везде, где это необходимо.

Использование моделирования физический явлений необходимо как для анимации модели бульдозера, так и для мониторинга дополнительных данных, таких как скорости, крутящие моменты, силы, энергии и столкновения [6].

Наиболее сложным аспектом моделирования твердого тела является контактное моделирование. Двумя основными подзадачами контактного моделирования являются обнаружение контактов и вычисление контактных усилий. Моделирование столкновения двух тел заключается в определении параметров (точки, глубины и нормали) коллизии аппроксимирующих контейнеров двух объектов и в задании реакции тел на это столкновение [7]. Существует несколько алгоритмов определения коллизий: априорные и апостериорные. В работе [7] рассматриваются методы апостериорного определения коллизий двух сфер и сферы с прямоугольным параллелепипедом (боксом). В источнике [8] дается обзор алгоритма обнаружения столкновений V-Clip, который позволяет быстро и надежно находить пары наиболее близких объектов между многогранными моделями, также кратко описывается несколько методов вычисления контактных сил, действующих между телами. Также, можно использовать физический движок, в котором используются готовые библиотеки, на которых основан механизм коллизий, если использование своего алгоритма обнаружения коллизий не является главной задачей в проекте. Например, в физическом движке Project Chrono доступны две различные системы столкновений: индивидуальная версия Bullet (библиотека) и собственный многоядерный движок для столкновений ChCollisionSystemMulticore [9].

Моделирование ходовой части бульдозера нереализуемо без применения двух понятий: гидростатическая трансмиссия, крутящий момент. Гидростатическая трансмиссия (ГСТ) является замкнутой гидравлической системой с гидронасосом и гидромотором [10]. В данном источнике [10] изложены расчеты параметров и рабочих характеристик. При использовании

ГСТ можно добиваться устойчивой работы в широком диапазоне чисел оборотов, начиная с самых малых. Гидростатические трансмиссии позволяют наиболее простым способом реализовать бесступенчатое регулирование скорости, в том числе автоматическое [11]. В научной статье [11] авторы приводят пример расчета ГСТ погрузчика с получением тяговых характеристик. Основная необходимость, которую следует учесть при разработке гидростатической трансмиссии, заключается в минимизации количества гидравлических компонентов, через которые проходит высокое давление жидкости. Эти компоненты имеют значительные размеры и массу.

1.1. Обзор аналогов

В последние годы наблюдается рост спроса на использование машин и механизмов в различных отраслях промышленности и народного хозяйства. Это связано с развитием технологий, увеличением объемов производства и необходимостью повышения эффективности работы оборудования. Одним из ключевых факторов, влияющих на рост спроса на машинную технику, является развитие информационных технологий и их использование для моделирования процессов эксплуатации техники. Использование информационных технологий позволяет создавать более точные и эффективные модели, которые помогают оптимизировать работу оборудования и снижать затраты на его эксплуатацию.

В данном разделе будет произведен обзор аналогов – решений, которые предоставляют возможность моделирования сложных систем.

В процессе поиска открытых проектов в области проектирования и реализации модели гусеничного бульдозера, был найден программный комплекс Универсальный механизм и два примера моделирования в этой среде, которые будут рассмотрены ниже.

1.1.1. Моделирование движения лесозаготовительной гусеничной машины (ЛЗГМ)

В статье [12] приводится пример моделирования элементов ходовой системы гусеничной машины в программе Универсальный механизм с помощью модуля UM Tracked Vehicles. Создание включает в себя выбор структурной схемы гусеничного движителя, здание подвески, добавление ведущего колеса, добавление направляющего колеса, создание модели гусеничной ленты, добавление упругих элементов. Общий вид корпуса может быть импортирован из программы трехмерного моделирования формы. Большое внимание уделяется выбору модели грунта.

Моделируется гусеничный обвод, задав внешний вид и подставив все необходимые условия для моделирования ЛЗГМ, получается виртуальная модель, позволяющая проводить исследования кинематики, динамики движителя. Пример движения гусеничной машины показан на рисунке 1 [12].

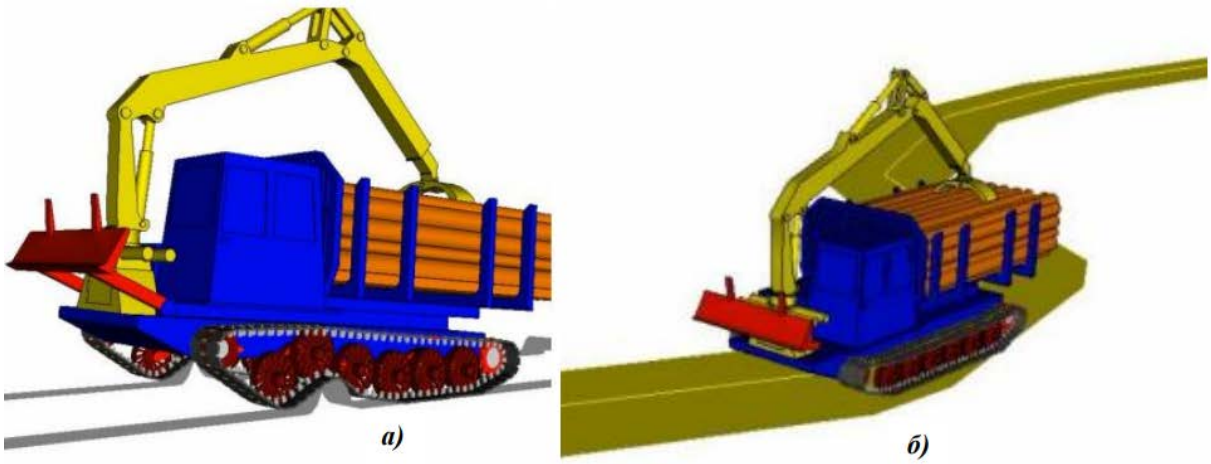

Рисунок 1 – Примеры виртуального эксперимента по движению через дискретные препятствия (а) и криволинейному движению (б)

В этом решении большее внимание уделяется моделированию именно движения, то есть просто визуализации процесса и взаимодействию грунта с гусеничным движителем, не проводятся расчеты нагрузок в узлах при движении, что является важным аспектом при моделировании ходовой системы. В программном комплексе Универсальный Механизм отсутствуют

программные инструменты, позволяющие сделать моделирование более четким и точным.

1.1.2. Моделирование динамики и нагруженности рамы гусеничного бульдозера ТМ10 ГСТ-12

Проект моделирования динамики и нагруженности рамы гусеничного бульдозера ТМ10 ГСТ-12, в котором произведен расчет сил, действующих на составные части бульдозера, и смоделировано поведение техники в транспортном режиме, выполнен в программном комплексе Универсальный Механизм. Модель бульдозера с указанием векторов силы в среде представлена на рисунке 2.

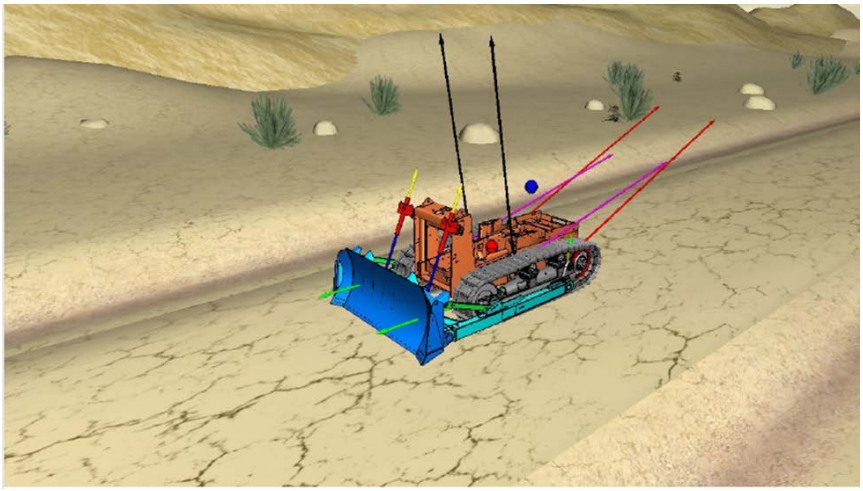

Рисунок 2 – Модель бульдозера в среде «Универсальный механизм»

Механизмы описываются как системы твердых тел, шарниров и силовых элементов. Модуль UM Tracked Vehicle предусмотрен для автоматизации процесса создания моделей гусеничных машин и анализа их динамики [11]. На рисунке 3 показан пример визуализации модели гусеничной техники.

Рисунок 3 – Пример моделирования динамики гусеничных машин [11]

Рассмотрим достоинства этого программного комплекса:

– анимация движения модели в процессе расчета;

– дополнительные модули: автомобильный модуль, железнодорожный модуль, модуль импорта из CAD программ и другие;

– многовариантные расчеты и оптимизация, экспорт результатов.

Но этот программный комплекс имеет и ряд недостатков:

– платный доступ к программным продуктам;

– ограниченные возможности и неудобство моделирования внутри программы;

– преимущественно конструкторский подход реализации модели.

Моделирование движения бульдозера должно включать не только графическое изображение и симуляцию, но и учёт физических свойств и возможность получать данные о том, как различные режимы работы влияют на ходовую систему бульдозера. Поэтому были выделены следующие критерии:

1. Аналог моделирует движение техники.
2. Аналог в ходе моделирования рассчитывает силы и нагрузки.
3. Аналог выполнен в доступном и удобном программном комплексе.
4. Аналог выводит необходимые параметры для анализа (например, мощность двигателя, скорость техники).
5. Аналог позволяет анализировать полученные результаты моделирования (таблицы нагрузок, отображение векторов силы).

Сравним имеющиеся аналоги по этим критериям в таблице 1:

Таблица 1 – Сравнение аналогов

Аналог / Критерии	Моделирование ЛЗГМ	Моделирование ТМ10 ГСТ-12
Моделирование движения	+	+
Точный расчет нагрузок	–	+–
Доступность среды разработки	–	–
Визуализация параметров	–	–
Анализ результатов	–	+

1.2. Вывод

Проведен аналитический обзор по тематике работы. Были рассмотрены существующие аналоги, в которых моделируется движение бульдозера, но не проводится расчет нагрузок на элементы ходовой части, отсутствует визуализация параметров. Необходимо создать свою модель ходовой части бульдозера. Выбрать подходящую среду разработки, предусмотреть возможность моделирования движения, вычисления нагрузок в узлах гусеничного движителя, направления сил.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ И ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ ПРОЕКТА

2.1. Функциональные требования

Для моделирования ходовой системы бульдозера были определены функциональные требования, которые соответствуют требованиям заказчика к итоговому результату.

– модель должна имитировать движение бульдозера с учетом его характеристик, веса бульдозера и других параметров;

– возможность расчета нагрузки на элементы ходовой системы бульдозера;

– во время запуска должна присутствовать визуализация моделирования работы ходовой системы бульдозера, включая отображение движения, параметров двигателя, нагрузки и других важных данных.

2.2. Нефункциональные требования

Модель должна соответствовать следующим нефункциональным требованиям:

– данные, полученные в результате моделирования, должны быть представлены в таблице во время симуляции.

Разработанная программа должна быть написана на языке программирования C++.

2.3. Выбор средств реализации

Для разработки модели бульдозера будет использоваться физический движок, который позволяют создавать реалистичные модели, учитывать все нюансы работы данной техники и ее взаимодействия с окружающей средой.

Физические движки представляют собой программные комплексы, которые имитируют законы физики и обеспечивают реалистичное взаимодействие объектов в виртуальном мире. Они используются для создания виртуальных прототипов различных устройств и машин, а также для проведения виртуальных испытаний и исследований. Физические движки позволяют моделировать не только работу механических систем, но и

взаимодействие объектов с окружающей средой, включая силы трения, аэродинамику, гидродинамику и другие факторы.

Физические движки условно разделяют на игровые и научные. Из игровых наиболее известными являются Unity и Unreal Engine. Их цель и применение понятны: облегчить процесс разработки игр, предоставляя готовые инструменты для создания графики, звука, перемещения персонажей и генерации физических эффектов. Применение игровых движков позволяет разработчикам сосредоточиться на основной идее игры, а не на технических деталях, и сокращает время разработки. В моделировании движения гусениц бульдозера и снятии нагрузок наиболее важным критерием при выборе физического движка является точность вычислений, что могут предоставить научные движки, которые применяются в научно-исследовательских расчётах и симуляциях, где скорость вычислений при этом не играет существенной роли.

Выделим два научных движка Project Chrono и ODE (Open Dynamics Engine), которые используются для моделирования физических процессов и поведения объектов в виртуальных средах. Оба движка предлагают широкий спектр функций и возможностей для создания реалистичных моделей и симуляций, однако они имеют некоторые различия в своих подходах и особенностях.

2.3.1. Open Dynamics Engine

ODE, также известный как Open Dynamics Engine, представляет собой библиотеку с открытым исходным кодом, предназначенную для моделирования динамики твердого тела. Этот мощный движок обладает простым в использовании API (программный интерфейс), обширными возможностями соединений, а также встроенным обнаружением столкновений с поддержкой функции трения.

ODE применяется для моделирования различных объектов, включая транспортные средства, объекты в виртуальной реальности и виртуальных существ. В настоящее время данная библиотека находит широкое применение

в компьютерных играх, инструментах 3D-разработки и программных средствах моделирования.

В ODE контакты представлены точками контакта, а не контактными поверхностями [13]. На рисунке 4 показан пример симуляции транспортного средства. ODE придаёт больше значение скорости и стабильности, чем физической точности. В некоторых случаях это может быть недостатком, особенно если требуется высокая точность в симуляции.

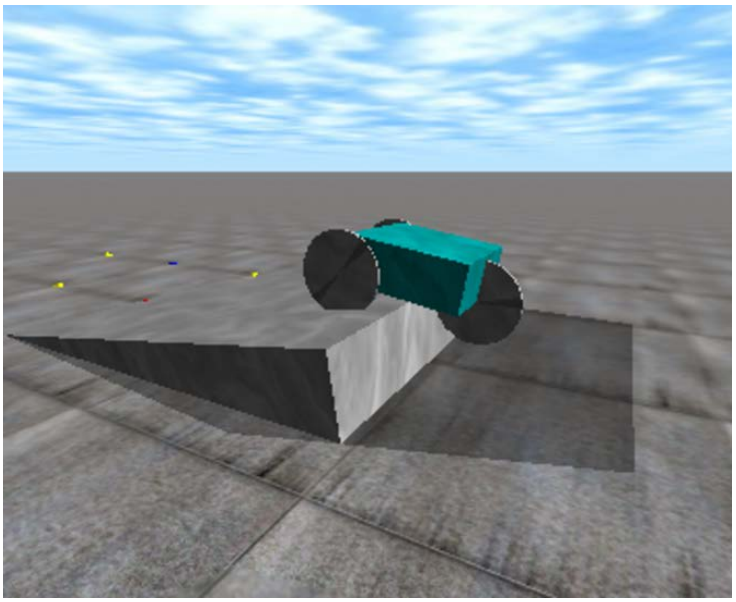

Рисунок 4 – Пример преодоления транспортным средством препятствий

Достоинства:

– возможность применять свою систему определения столкновений;

– наличие C и C++ интерфейсов;

– написано множество юнит тестов, и много пишутся сейчас.

Недостатки:

– ограниченная точность (что является проблемой для нашего проекта);

– отсутствие активной поддержки, устаревшая документация (что затруднит процесс разработки);

– ограниченная поддержка геометрии.

2.3.2. Project Chrono

Project Chrono является физическим движком с открытым исходным кодом, реализованном на c++. Проектная библиотека PROJECT CHRONO [14] предоставляет возможность интеграции в программное обеспечение для симуляции различных видов транспортных средств, работающих на деформируемой местности, роботов, мехатронных систем, совместимых механизмов и взаимодействия жидкости и твердого тела. Возможности систем включают жесткие и гибкие / совместимые компоненты с ограничителями, двигателями и контактами; компоненты могут иметь трехмерные формы для обнаружения столкновений. На рисунке 5 можно заметить точность отрисовки объектов и визуализацию процесса столкновения ковша с предполагаемой поверхностью.

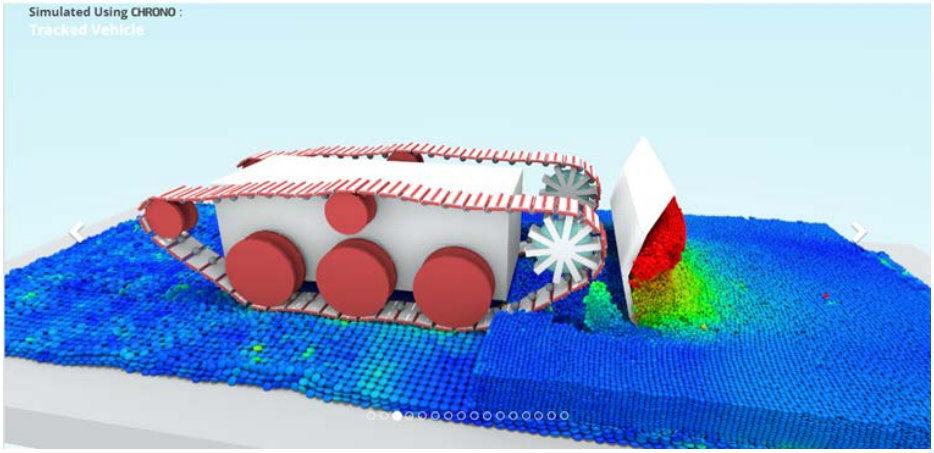

Рисунок 5 – Симуляция гусеничной техники

Преимущества:

1. Высокая точность моделирования: Project Chrono использует передовые физические модели и алгоритмы для создания высокоточных симуляций.
2. Широкий спектр поддерживаемых платформ: Project Chrono поддерживает множество операционных систем и платформ, включая Windows, Linux, macOS.
3. Простота использования: Project Chrono имеет простой и понятный интерфейс, который упрощает процесс создания и настройки симуляций.

4. Открытый исходный код: Project Chrono распространяется под лицензией GNU General Public License, что позволяет разработчикам свободно использовать и модифицировать код движка.
5. Доступно расширение для работы с языком программирования Python.
6. Доступная и понятная документация.

Недостатки:

1. Скорость выполнения симуляции.
2. Ограниченность сведений о практическом использовании.
3. Необходимость установки дополнительных библиотек для работы с некоторыми форматами файлов.

Один из ключевых отличий между этими движками заключается в их производительности. ODE часто считается более быстрым и эффективным движком, особенно для симуляции большого количества объектов или сложной динамики. Однако Project Chrono может предложить более высокую точность и детализацию моделирования, благодаря использованию более сложных физических моделей и алгоритмов.

Были определены следующие критерии, исходя из удобства пользования и соответствия требованиям к итоговому результату моделирования:

– кроссплатформенность - возможность использовать удобную для себя ОС: Windows, Linux и т.п.

– наличие актуальной технической документации повышает скорость разработки и процесс установки ПО;

– точность вычислений является самым важным критерием при вычислении нагрузок на ходовую систему бульдозера;

– открытый исходный код, чтобы была возможность разобраться в уже готовых написанных примерах.

В таблице 2 приведен сравнительный анализ двух физических движков.

Таблица 2 – Сравнение физических движков

Физический движок	Кроссплатформенность	Техническая документация	Точность вычислений	Открытый исходный код
Open Dynamics Engine	+	+–	–	–
Project Chrono	+	+	+	+

Для достижения поставленной цели проекта был выбран физический движок с открытым исходным кодом Project Chrono, так как в моделировании ходовой системы бульдозера важна точность вычислений нагрузок в узлах техники, а скорость симуляции является второстепенным критерием.

2.4. Выбор среды разработки

При выборе среды разработки необходимо учитывать тот факт, чтобы взаимодействие между компонентами было корректным и они работали без сбоев и ошибок. При написании проекта с помощью Project Chrono, нам необходимо использовать множество библиотек и компонентов, которые должны работать сообща и не конфликтовать друг с другом.

Выбор операционной системы (ОС) является важным этапом в разработке программного обеспечения. От выбора ОС зависит стабильность работы программы, ее производительность и безопасность.

ОС Linux была выбрана для разработки данного проекта, потому что она является одной из самых популярных, бесплатных и стабильных операционных систем. Она имеет открытый исходный код, что позволяет разрабатывать программы с учетом индивидуальных требований и возможностей. Библиотеки Project Chrono находятся в свободном доступе для разработки проектов под операционной системой Linux.

В качестве дистрибутива Linux был выбран Ubuntu 22.04 по принципу простоты и удобства для пользования при написании объемного проекта:

– комфортный графический интерфейс;

– удобство и простота использования;

– минимально необходимый функционал без сторонних приложений.

При работе с большим количеством данных, библиотек, пакетов необходим инструмент, который сможет автоматически собирать проект, позволяя описать его структуру и настроить параметры компиляции. В данной работе используется утилита CMake. Это инструмент с открытым исходным кодом, предназначенный для управления процессом сборки программных проектов. Этот генератор системы сборки является важным для разработчиков, которые работают с C, C++ и другими языками программирования. CMake используется для создания сборки файлов проектов для различных IDE и настройки процесса сборки программного обеспечения.

CMake широко используется с Qt. Qt Creator – это кроссплатформенная интегрированная среда разработки (IDE), разработанная для повышения эффективности работы программистов. Qt предлагает множество функций, которые упрощают процесс разработки, такие как встроенный отладчик, редактор кода с подсветкой синтаксиса, менеджер проектов и многое другое. Кроме того, Qt Creator является бесплатным и открытым исходным кодом. Qt оснащен визуальной средой разработки для графического интерфейса, который позволяет создавать диалоги и формы [15].

ВЫВОД

Проанализировав средства разработки, можно сделать вывод о том, что моделирование ходовой системы будет реализовано в фреймворке Qt, с использованием сборщика проекта CMake, на базе физического движка Project Chrono, в операционной системе Linux.

Выбор Qt обусловлен тем, что данный фреймворк предоставляет широкий спектр инструментов, возможность режима отладки.

Использование CMake в качестве сборщика проекта позволяет автоматизировать процесс сборки и настройки проекта. CMake обеспечивает гибкость в определении структуры проекта и параметров компиляции, а также позволяет легко интегрировать сторонние библиотеки и модули.

Физический движок Project Chrono был выбран благодаря его высокой точности моделирования физических процессов и широкому спектру поддерживаемых платформ. Кроме того, движок обладает открытым исходным кодом, что упрощает процесс адаптации и модификации кода.

В качестве операционной системы для разработки и запуска проекта была выбрана Linux, так как она обладает рядом преимуществ, таких как открытость, бесплатность, надежность и производительность.

3. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ МОДЕЛИ ХОДОВОЙ СИСТЕМЫ БУЛЬДОЗЕРА

3.1. Архитектура проекта

Проектирование представляет собой процесс, в ходе которого определяется структура, элементы, интерфейсы и прочие особенности системы или ее отдельных компонентов. В итоге проектирования формируется проект – комплекс моделей, свойств и характеристик, изложенных в форме, пригодной для последующей реализации [16].

3.1.1. Блок-схема процесса моделирования

Процесс создания модели бульдозера сложен, так как он включает анализ и проектирование множества взаимосвязанных систем, таких как двигатель, трансмиссия, движитель, взаимодействие объектов и внешняя среда. На рисунке 6 представлена блок схема процесса моделирования.

На первых этапах моделирования происходит импортирование и инициализация объектов (в формате .obj) в проект, в главный файл D12. Исходный код инициализации объектов представлен в листинге А1 приложении А. После чего создаются отдельно классы каждого компонента (например, тележка, звёздочка, ведущее колесо и т.п.), в которых далее детально будут прописаны методы и функции: соответствующие коллизии, соединения с другими объектами.

На этапе расчета гидростатической трансмиссии прописываются параметры объемных постоянных насоса и мотора, максимальное давление в системе, обороты двигателя, передаточное число редуктора и производится расчет крутящего момента.

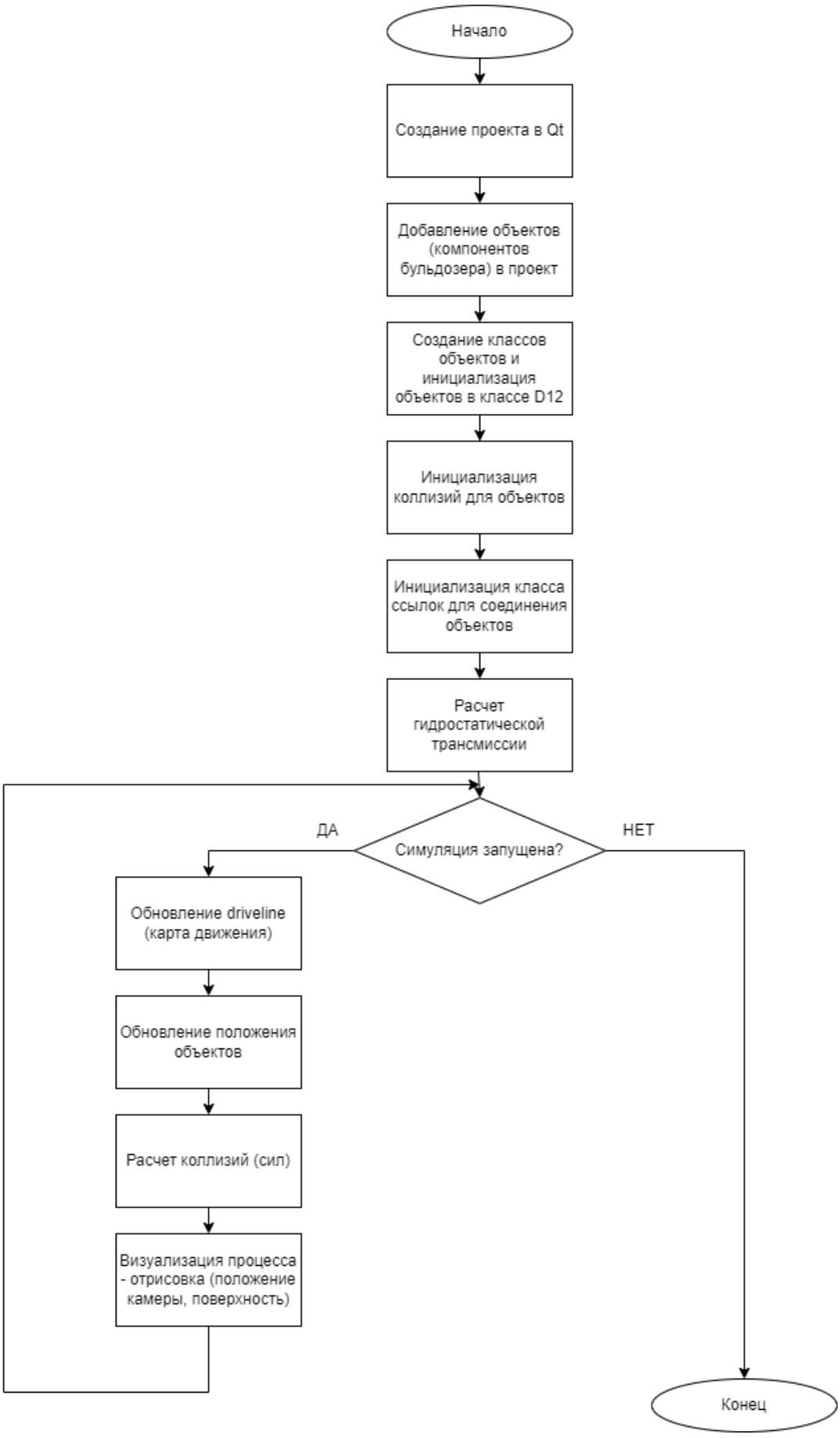

Рисунок 6 – Блок-схема процесса моделирования

Опишем составляющие компоненты модели бульдозера, которые участвуют в процессе моделирования.

1. Рама (от англ. chassis)

Рама — это основная несущая конструкция, которая служит для крепления всех рабочих органов и агрегатов бульдозера. К раме крепится тележка, а также она соединяется с балансирной балкой.

2. Тележка (от англ. trolley)

Тележка — это элемент ходовой части, который состоит из рамы, опорных и поддерживающих катков. Тележка соединяется с рамой в двух местах. Через опорные и поддерживающие катки обводится гусеничная лента.

3. Балансирная балка (от англ. balanced beam)

Балансирная балка представляет собой длинную горизонтальную компоненту, размещаемую под шасси бульдозера. Основная цель этой детали заключается в обеспечении равномерного распределения веса машины путем балансировки нагрузки между передней и задней частями.

4. Переднее натяжное колесо (от англ. idler)

Натяжное колесо выполняет функцию натяжного устройства, а также задает направление движения и ликвидирует провисания ленты. Располагается с каждой стороны в начале ходовой части, соединено с тележкой.

5. Звезда (от англ. sprocket)

Звёздочка – элемент гусеничного движителя, который осуществляет перематывание гусеничной ленты и преобразует собственное вращательное движение в поступательное движение гусеничного бульдозера. Располагается с каждой стороны в конце ходовой части.

6. Отвал (от англ. blade)

Отвал — навесное оборудование для бульдозеров, применяется для выравнивания территории и уплотнения грунта. Соединяется с тележкой и рамой.

7. Гусеничный башмак (от англ. track shoe)

Гусеничные башмаки — это металлические пластины, которые соединены болтами со звеньями. Их главная функция — обеспечивать сцепление с грунтом и плавный ход. Так как гусеничные башмаки — единственные элементы, непосредственно контактирующие с грунтом, они первыми принимают на себя ударные нагрузки и являются основой всей системы.

8. Траки (от англ. track)

Гусеничный трак бульдозера D12 состоит из 45 башмаков. Гусеничная цепь служит для передачи веса бульдозера на опорную поверхность и обеспечения сцепных качеств, необходимых для реализации тяговой силы бульдозера.

9. Опорные и поддерживающие катки (от англ. road wheel)

Опорные катки представляют собой металлические колеса, установленные на нижнюю часть тележки, чтобы поддерживать вес машины и направлять гусеницу.

Поддерживающие катки представляют собой металлические колеса, установленные на верхнюю часть тележки. Именно они направляют и поддерживают гусеницы между звездочкой и передним натяжным колесом.

На рисунке 7 представлена модель бульдозера D12 в симуляции.

Рисунок 7 — Общий вид модели в симуляции

После того, как определены этапы моделирования и составные компоненты модели, для наглядного представления архитектуры данного проекта было решено использовать диаграмму классов.

3.1.2. Диаграмма классов

Цель UML (Unified Modeling Language) диаграммы можно определить как реализацию простого механизма моделирования всех возможных практических систем в сложной современной среде.

Диаграмма классов представляет структуру системы, показывая классы, их атрибуты и методы, а также связи между классами [17].

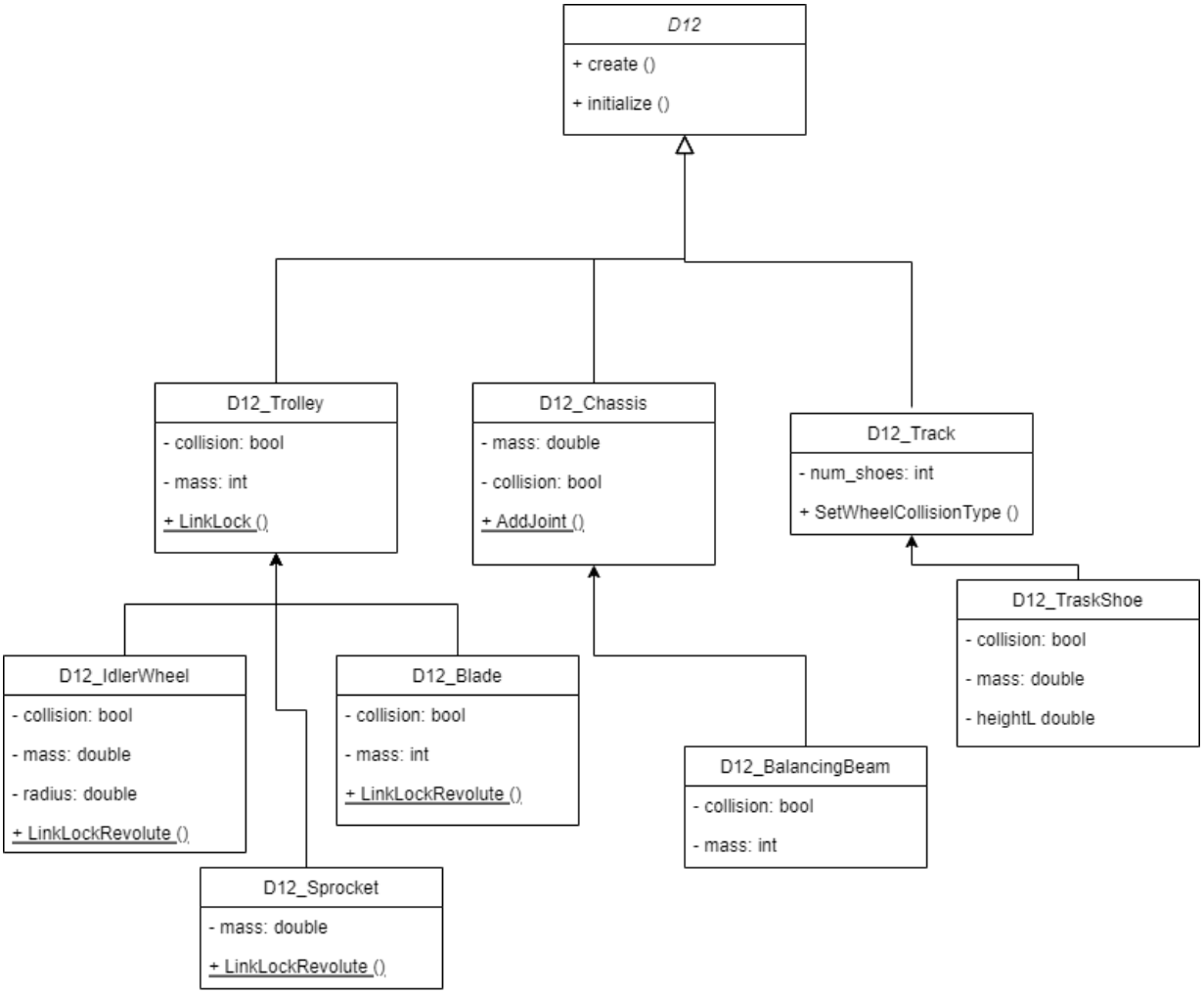

Рисунок 8 — Диаграмма классов

Исходя из данных диаграммы, представленной на рисунке 8, получена информацию о том, что проект имеет иерархическую структуру, а программный класс D12 является главным в проекте и именно в нём происходит сборка всех компонентов.

В проекте описано больше классов, каждый из которого выполняют важную функцию (например, класс ссылок, который ограничивает тело (body) в движении), о них далее будет рассказано подробнее.

3.2. Класс ссылок (от англ. link)

Рассмотрим компонент, который является важным фактором при разработке проекта.

Корпус (body) в Chrono относительно другого тела или грунта может быть ограничен в движении, что достигается благодаря классу ссылок [18].

Класс ChLink является основным, из которого создаются различные наборы ссылок. Наиболее популярными являются:

– ChLinkLockLock (позволяет задавать предписанное движение в любом из 6 направлений (3 перемещения и 3 поворота по осям X,Y,Z))

– ChLinkLockRevolute (класс для моделирования вращательного соединения между двумя объектами)

При соединении рамы и балансирной балки (рисунок 9) будет использовано шарнирное стопорное соединение (Link Lock Joint).

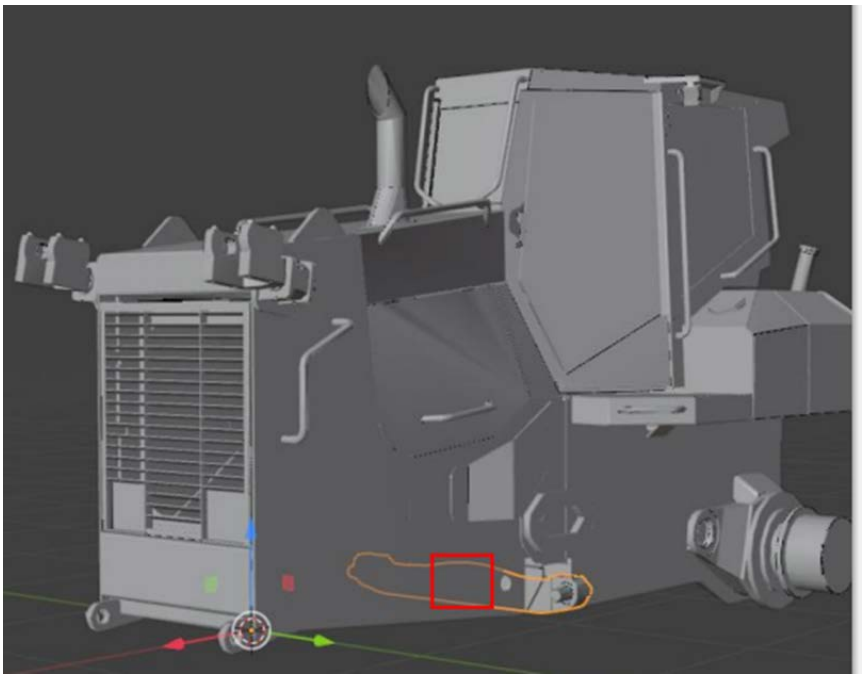

Рисунок 9 — соединение рамы и балансирной балки

Тележка соединяется с рамой в двух местах на рисунке 10: A) через вращательный шарнир между тележкой и рамой; B) через шарнирное стопорное соединение между опорной балкой и тележкой.

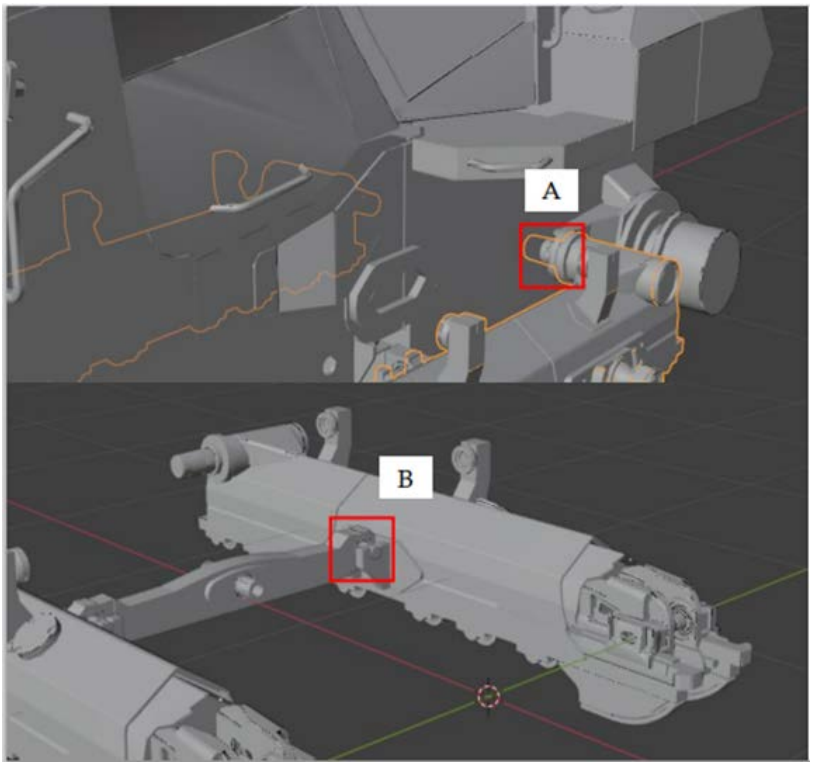

Рисунок 10 — Соединение рамы и тележки

В методе BuildLinkType класса ChLinkLock устанавливаются ограничения для координат ссылки (X, Y, Z, E0, E1, E2, E3): первые три координаты отвечают за поворот относительно осей, оставшиеся за вращение. Ограничения прописываются с помощью параметров true — разрешение движения, false — запрет.

Шарнирное стопорное соединение также закрепляет гидроцилиндры отвала с рамой (рисунок 11A), также брусьями соединяется с тележкой (рисунок 11B).

Связь отвала с тележкой инициализируется с помощью класса ChLinkLockRevolute, которое определяет возможность вращения, а также связь отвала с рамой — класс ChLinkLockLock, что подразумевает полное отсутствие движения.

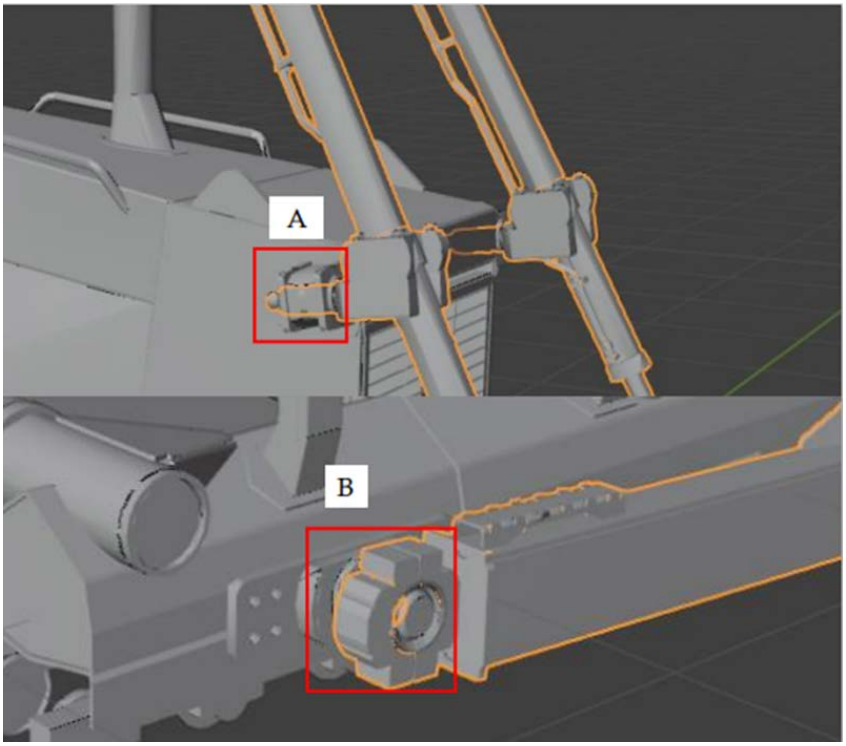

Рисунок 11 — Соединение тележки и отвала

Исходный код инициализации связей объектов представлен в листингах Б1-Б3 приложения Б.

3.3. Система движения и расчет трансмиссии

Трансмиссия бульдозера играет ключевую роль в работе бульдозера, передавая усилия от двигателя к рабочим органам машины. Благодаря трансмиссии бульдозер способен выполнять основные операции, такие как движение, разворот и поднятие-опускание отвала. Раньше гусеничные бульдозеры были в группе наиболее сложных в управлении видов техники, работающей с землей. Поэтому в последние годы шло активное внедрение новых технологий управления, одной из которых является гидростатическая трансмиссия (ГСТ) – передача энергии статического давления жидкости, создаваемого в объемном гидронасосе. Главные достоинства ГСТ: бесступенчатое плавное изменение крутящего момента, упрощенная механическая разводка по машине.

Бульдозер D12 оснащен гидростатической трансмиссией, в состав которой входит: двигатель, два гидронаноса, два гидромотора, редукторы. На рисунке 12 представлен набор ГСТ [19].

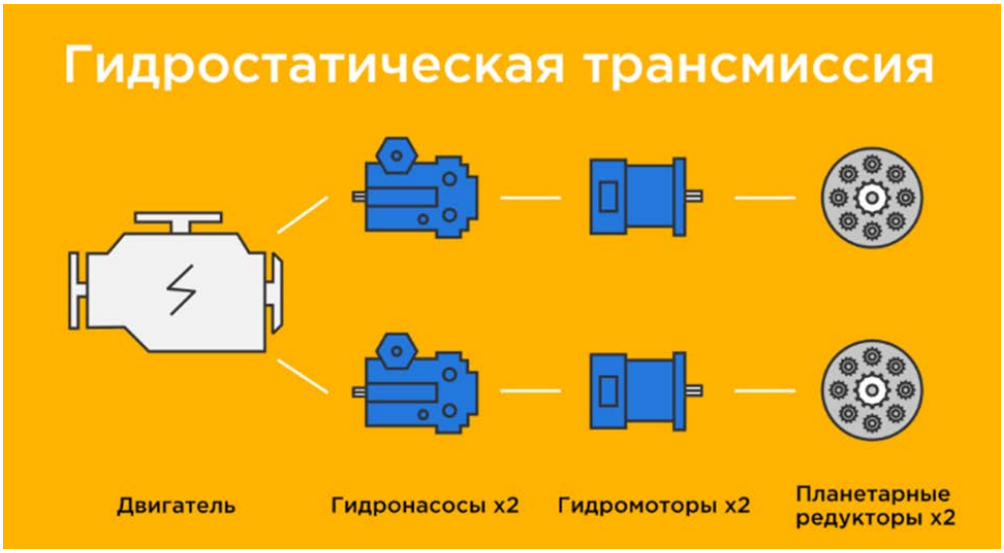

Рисунок 12 — схема гидростатической трансмиссии

Чтобы смоделировать движение бульдозера, необходимо разобраться с этой составляющей движения, а именно с параметрами составных частей ГСТ.

Расчет крутящего момента непосредственно в программном коде производиться по формуле (1):

m_shaft_torque = (m_motor_torque * 0.9 * 0.9 * (1)

(m_q_motor * motor_param)) / (m_q_pump * pump_param),

где m_q_motor — объемная постоянная мотора (165), m_q_pump — объемная постоянная насоса (135).

Двигатель через болтовое соединение соединяется с гидронасосами и передает им крутящий момент, гидронасосы создают давление в системе и тем самым создают крутящий момент на гидромоторах, которые, в свою очередь, через планетарный редуктор вращают звездочку и приводят гусеницу в движение.

Максимальное давление в системе — 400 Па, максимальный крутящий момент достигается при значении 2000 оборотов двигателя. Передаточное число редуктора — 55.

Ходовая часть большинства машин строительного класса оборудована натяжным устройством в сборе с пружиной. Механизм натяжения используется для регулировки натяжения цепи. Пружина выполняет функцию амортизатора, который абсорбирует удары и смягчает их воздействие во время использования при воздействии ударных нагрузок. Ударные нагрузки, возникающие при движении по неровной поверхности, поглощаются пружиной.

3.4. Визуализация процесса

Для отображения бульдозера на экране во время симуляции необходима визуализация среды выполнения, которая в Project Chrono выполняется с помощью модуля Irrlicht. Irrlicht Engine — трёхмерный графический движок, который является бесплатным свободным программным продуктом, написанный на C++.

Модуль IRRLICHT используется для отображения симуляций Chrono в интерактивном 3D-представлении [20]. К основным функциям относится: поддержка всех форм визуализации Chrono physics (тела, ссылки и т.д.), возможность управления мышью и клавиатурой (поворот камеры, управление движения камеры, панель настроек), возможность дополнительно нарисовать контакты векторами в 3D-виде.

В проекте система визуализации описывается в отдельном классе с помощью подключенной библиотеки Irrlicht. Отображение параметров движения бульдозера (рулевое управление, газ, тормоз), определение позиции камеры, а также отображение информации от системы движения (параметры двигателя) — всё это прописывается в файле ChVehicleVisualSystemIrrlicht. На рисунке 13 представлены параметры, которые выводятся во время симуляции. Исходный код отображения параметров двигателя представлен в листинге Б4 приложения Б.

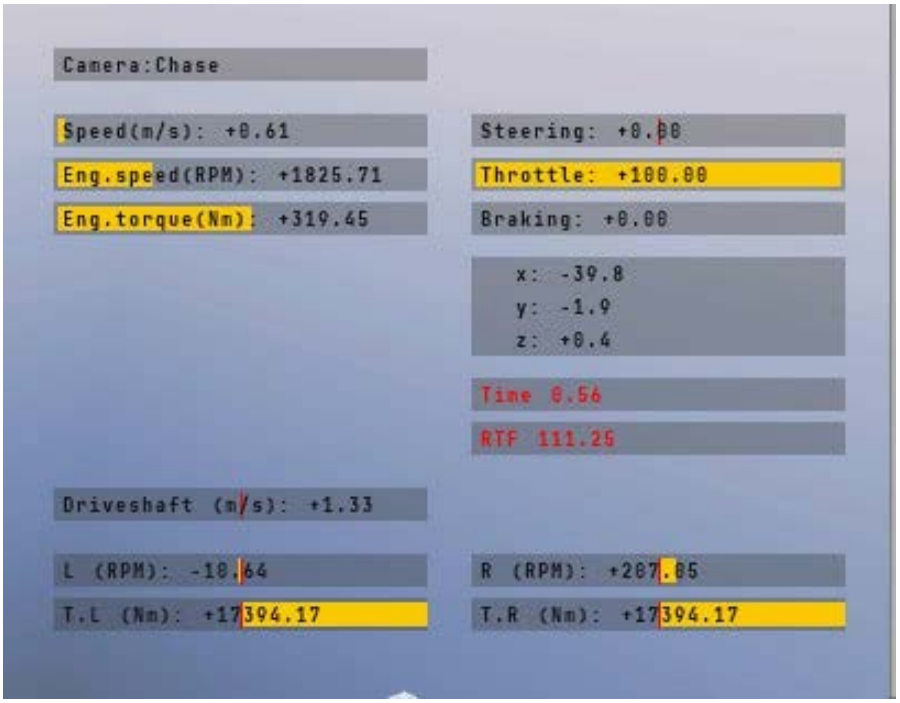

Рисунок 13 – Отображаемые параметры во время симуляции

Рассмотрим параметры, которые выведены на экран и которые следует учитывать при анализе работы бульдозера во время симуляции.

а) speed – скорость движения бульдозера;

б) engine speed – обороты двигателя;

в) engine torque – крутящий момент двигателя;

г) driveshaft – скорость вращения звездочки;

д) time – время реальной симуляции;

е) RTF (Real Time Factor) – это соотношение между реальным и моделируемым временем. Например, значение RTF, равное 0,5, указывает на то, что для симуляции одной минуты в реальном времени потребуется две минуты, и каждая секунда симуляции будет занимать примерно две секунды реального времени;

ж) throttle – газ;

з) breaking – тормоз.

Значения параметров газа или тормоза дают понимание в каком состоянии находится бульдозер (в движении или в покое).

ВЫВОД

Для наглядного представления архитектуры проекта были использованы блок-схема процесса моделирования движения бульдозера и диаграмма классов. Показано соединение объектов с помощью класса ссылок. Движение бульдозера происходит за счет трансмиссии, которая передает энергию от двигателя к рабочим органам техники. Описан принцип гидростатической трансмиссии, а также отображения параметров во время симуляции.

4. ТЕСТИРОВАНИЕ РАБОТЫ МОДЕЛИ

Для проверки работы симуляции и сбора результатов моделирования будет использоваться основанное на требованиях тестирование. Основная задача тестирования, основанного на требованиях, заключается в том, чтобы убедиться, что в процессе тестирования требования к проверяемому объекту были учтены («покрыты») для определения соответствия этого объекта ожиданиям конечного пользователя [21]. В основанном на требованиях тестировании преимущественно используется тестирование по сценарию. Контрольные примеры пишутся заранее перед выполнением тестирования. Для проверки работоспособности модели и соответствию требованиям, был написан блок “driveline”, считывающий текстовый файл, в котором прописана карта движения с такими параметрами как: газ, тормоз, направление движения. На рисунке 14 представлена модель бульдозера во время симуляции с отображением параметров в правом верхнем углу.

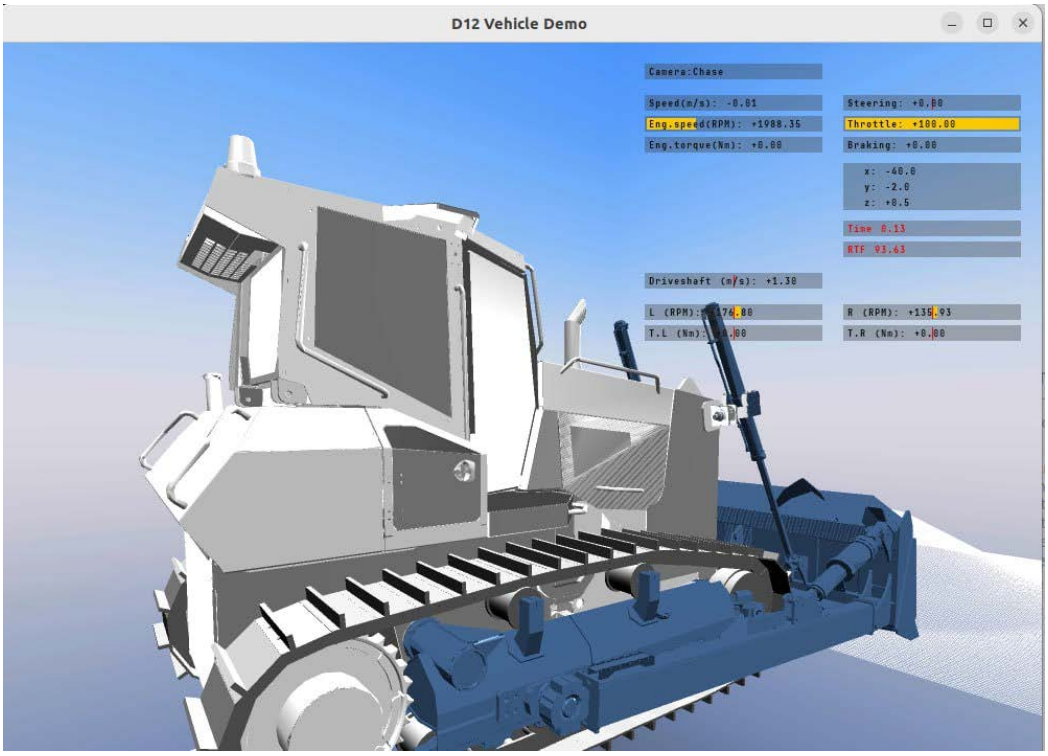

Рисунок 14 – Модель бульдозера с отображением параметров во время симуляции

Важным требованием, предъявляемым к моделированию бульдозера, была возможность снятия показателей нагрузки на элементы ходовой системы. На рисунке 15 показан результат отображения нагрузок в числовых значениях во время запуска симуляции. Для анализа полученных значений необходимо вывести их в текстовый файл и создать диаграмму по этим данным, пример такой диаграммы представлен на рисунке 16.

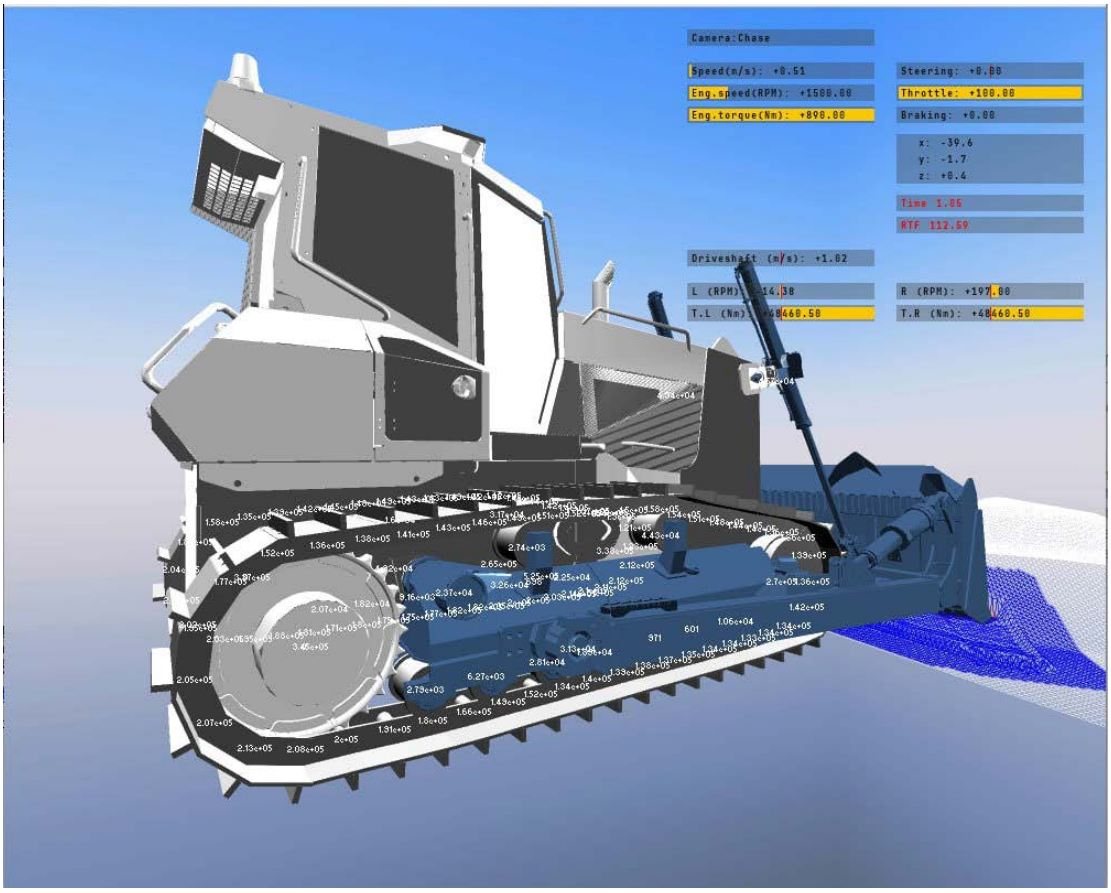

Рисунок 15 – Модель бульдозера с отображением нагрузки на ходовую систему бульдозера

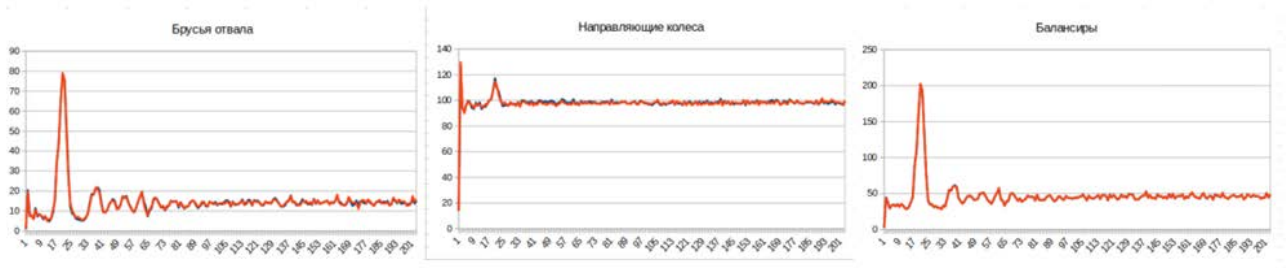

Рисунок 16 – Силы, действующие на тележки

ВЫВОД

Для проверки результатов моделирования движения ходовой системы бульдозера было выбрано основанное на требованиях тестирование, которое подразумевает проверку итоговой симуляции на соответствие заявленным требованиям по написанному заранее сценарию.

В результате проведенного тестирования, можно сделать вывод, что моделирование движения бульдозера отвечает всем требованиям, а именно движение техники за счет вращения звезды и гусениц, возможность получения результатов нагрузок в узлах ходовой системы бульдозера, отображение параметров (скорость движения, обороты двигателя, крутящий момент двигателя, газ, тормоз).

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была спроектирована и реализована модель ходовой системы бульдозера.

В процессе реализации была сформулирована цель работы — создание компьютерной модели, которая будет имитировать поведение ходовой системы бульдозера.

Для достижения поставленной цели были сделаны следующие шаги:

1. Выполнен обзор литературы и аналогов.
2. Выбраны средства реализации проекта.
3. Выполнена разработка архитектуры программы и реализация модели.
4. Проведено тестирование модели ходовой системы бульдозера.

В ходе выполнения данной работы был проведен аналитический обзор по тематике работы, рассмотрены аналоги моделирования движения бульдозера и представлен их сравнительный анализ.

В качестве средства реализации был выбран физический движок Project Chrono, который отличается высокой точностью моделирования физических процессов и широким спектром поддерживаемых платформ.

Выполнено проектирование архитектуры проекта. В результате которого была построена блок-схема процесса моделирования движения бульдозера и диаграмма классов, которая представляет структуру системы, показывая классы, их атрибуты и методы, а также связи между классами. Реализация выполнена на языке программирования C++.

Выполнено тестирование моделирования, в ходе которого было определено соответствие установленным требованиям.

Результатом моделирования является движение бульдозера, вращение звездочки, следовательно, вращение гусениц. При работе симуляции на экране отображаются характеристики двигателя, параметры движения. Есть возможность посмотреть направление сил, а также снять нагрузки, проявляемые при движении на ходовую систему.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Deng, Y-M Constraint-based functional design verification for conceptual design / Y-M Deng, G.A Britton, S.B Tor // ELSEVIER LTD. – 2000. — 899 p.

2. Щербаков, В.С. Система автоматизации моделирования рабочего процесса бульдозерного агрегата / В.С. Щербаков, В.А. Кулибаба // Вестник Воронежского государственного технического университета. – 2011. – 4 с.

3. Погорелов, Д. Ю. Компьютерное моделирование динамики технических систем с использованием программного комплекса «Универсальный механизм» / Д.Ю. Погорелов // Вестник компьютерных и информационных технологий. – 2005. – № 4. – С. 27–34.

4. Клубничкин, Е. Е. Динамическое моделирование движения гусеничной лесозаготовительной машины с использованием прикладных пакетов компьютерных программ / Е. Е. Клубничкин [и др.] // Вестник Московского государственного университета леса – Лесной вестник. – 2012. – № 8. – С. 41–47.

5. Кондаков, С.А. Активизация физических знаний через моделирование в виртуальном мире / С.А. Кондаков // Современная высшая школа: инновационный аспект. – 2021. – №4. – 8 с.

6. Weistroffer, V. Using Physics-Based Digital Twins and Extended Reality for the Safety and Ergonomics Evaluation of Cobotic Workstations / V. Weistroffer, F. Keith, A. Bisiaux, C. Andriot, A. Lasnier // Virtual Reality in Industry. – 2022. — 18 p.

7. Трушин, А.М. Определение коллизий аппроксимирующих сфер и прямоугольных параллелепипедов в системах трехмерного моделирования / А.М. Трушин // Программные продукты и системы. – 2015. – 5 с.

8. Mirtich, Brian Rigid Body Contact Collision Detection to Force Computation / Brian Mirtich // TR98-01. – 1998. — 20 p.

9. Collisions [Электронный ресурс]. – URL: https://api.projectchrono.org/development/collisions.html (Дата обращения: 17.04.2024)

10. Петров, В.А. Гидрообъемные трансмиссии самоходных машин: учебник / В.А. Петров. – М.: Машиностроение, 1988. – 248 с.

11. Брилевский, О. В. Выбор схем и расчет функциональных характеристик гидростатических трансмиссий мобильных машин / О. В. Брилевский, А. П. Стецко, В. С. Шевченко // Вестник Белорусского национального технического университета : научно-технический журнал. – 2008. – № 3. – С. 43-47.

12. Универсальный механизм – программный комплекс для моделирования динамики механических систем [Электронный ресурс]: – URL: https://www.umlab.ru/ (дата обращения: 02.03.2024)

13. Kooijman, R.C. Evaluation Of Open Dynamics Engine Software / R.C.Kooijman // Eindhoven University of Technology - Traineeship report. – 2010. – 69 p.

14. PROJECTCHRONO An Open Source Multi-physics Simulation Engine [Электронный ресурс]: – URL: https://projectchrono.org/ (дата обращения: 17.01.2024)

15. Краткий обзор кроссплатформенного фреймворка Qt [Электронный ресурс]: – URL: https://nicknixer.ru/programmirovanie/kratkij-obzor-krossplatformennogo-frejmvorkaqt/ (дата обращения: 03.03.2024)

16. Термин проектирование [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Проектирование (Дата обращения: 21.03.2024)

17. UML Class Diagram Tutorial [Электронный ресурс]. – URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ (Дата обращения: 21.03.2024)

18. Project Chrono: Links [Электронный ресурс]. – URL: https://api.projectchrono.org/links.html (Дата обращения: 23.03.2024)

19. Бульдозер с ГСТ: разбираемся в особенностях гидростатической трансмиссии [Электронный ресурс]. – URL: https://tm10.ru/press-center/news/1666/ (Дата обращения: 23.03.2024)

20. Install the IRRLICHT module [Электронный ресурс]. – URL: https://api.projectchrono.org/module_irrlicht_installation.html (Дата обращения: 07.04.2024)

21. ГОСТ Р 56920-2016. Системная и программная инженерия. Тестирование программного обеспечения. – М.: Стандартинформ, 2016. – 53 с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Листинг А1 – Создание и инициализация объектов

```
void D12_Vehicle::Create(bool fixed,
                         TrackShoeType shoe_type,
                         DoublePinTrackShoeType
shoe_topology,
                         ChTrackShoeBandANCF::ElementType
element_type,
                         bool constrain_curvature,
                         int num_elements_length,
                         int num_elements_width,
                         DrivelineTypeTV driveline_type,
                         BrakeType brake_type,
                         bool use_track_bushings,
                         bool use_suspension_bushings,
                         bool use_track_RSDA,
                         CollisionType
chassis_collision_type) {

m_chassis =
chrono_types::make_shared<D12_Chassis>("Chassis", fixed,
chassis_collision_type);
    m_blade =
chrono_types::make_shared<D12_Blade>("D12_Blade");
    m_trolley =
chrono_types::make_shared<D12_Trolley>("D12_Trolley");
    m_balancing_beam =
chrono_types::make_shared<D12_BalancingBeam>("D12_BalancingB
eam");

m_tracks[0] =
chrono_types::make_shared<D12_TrackAssemblySinglePin>(
        LEFT, brake_type, use_track_bushings,
use_suspension_bushings, use_track_RSDA);
    m_tracks[1] =
chrono_types::make_shared<D12_TrackAssemblySinglePin>(
        RIGHT, brake_type, use_track_bushings,
use_suspension_bushings, use_track_RSDA);

       switch (driveline_type) {
        case DrivelineTypeTV::SIMPLE:
            m_driveline =
chrono_types::make_shared<D12_SimpleDriveline>();
            break;
        case DrivelineTypeTV::BDS:
            m_driveline =
chrono_types::make_shared<D12_DrivelineBDS>();
            break;
    }
```

```
     GetLog() << "D12 vehicle mass = " << GetMass() << "
kg.\n";
}

void D12_Vehicle::Initialize(const ChCoordsys<>& chassisPos,
double chassisFwdVel) {

    m_chassis->Initialize(m_system, chassisPos,
chassisFwdVel, WheeledCollisionFamily::CHASSIS);
    m_balancing_beam->Initialize(m_chassis,
m_chassis->GetBody(), ChVector<>(0,0,0));
    m_trolley->Initialize(m_chassis,
m_balancing_beam->GetBody(), ChVector<>(0,0,0));
    m_blade->Initialize(m_chassis, m_trolley->GetBody(),
ChVector<>(0,0,0));

    double track_offset = 1;
    m_tracks[0]->Initialize(m_chassis, m_trolley,
ChVector<>(0, track_offset, 0), m_create_track);
    m_tracks[1]->Initialize(m_chassis, m_trolley,
ChVector<>(0, -track_offset, 0), m_create_track);

    m_driveline->Initialize(m_chassis, m_tracks[0],
m_tracks[1]);

    ChTrackedVehicle::Initialize(chassisPos, chassisFwdVel);
}
```

ПРИЛОЖЕНИЕ Б

Листинг Б1 — Создание и инициализация связи между балансирной балкой и рамой

```
m_link_lock = chrono_types::make_shared<ChLinkLockLock>();
m_link_lock->SetNameString("Beam_Joint");
m_link_lock->Initialize(carrier, m_balancing_beam_body,
ChCoordsys<>(link_pos, link_to_abs.GetRot() *
Q_from_AngX(CH_C_PI_2)));
chassis->GetSystem()->AddLink(m_link_lock);
```

Листинг Б2 - Создание и инициализация связи для тележек

```
m_rev_left = chrono_types::make_shared<ChLinkLockLock>();
m_rev_left->SetNameString("Trolley_left");
m_rev_left->Initialize(carrier, m_trolley_body,
ChCoordsys<>(pos_left, rev_left_to_abs.GetRot() *
Q_from_AngX(CH_C_PI_2)));
chassis->GetSystem()->AddLink(m_rev_left);
m_rev_right = chrono_types::make_shared<ChLinkLockLock>();
m_rev_right->SetNameString("Trolley_right");
m_rev_right->Initialize(carrier, m_trolley_body,
ChCoordsys<>(pos_right, rev_right_to_abs.GetRot() *
Q_from_AngX(CH_C_PI_2)));
chassis->GetSystem()->AddLink(m_rev_right);
m_lock_left = chrono_types::make_shared<ChLinkLockLock>();
m_lock_left->SetNameString("Trolley_left");
m_lock_left->Initialize(chassis->GetBody(), m_trolley_body,
ChCoordsys<>(lock_pos_left, lock_left_to_abs.GetRot() *
Q_from_AngX(CH_C_PI_2)));
```

```
m_lock_right = chrono_types::make_shared<ChLinkLockLock>();
m_lock_right->SetNameString("Trolley_right");
m_lock_right->Initialize(chassis->GetBody(), m_trolley_body,
ChCoordsys<>(lock_pos_right, lock_right_to_abs.GetRot() *
Q_from_AngX(CH_C_PI_2)));
```

Листинг Б3 — Создание и инициализация связей отвала

```
m_rev_left =
chrono_types::make_shared<ChLinkLockRevolute>();
m_rev_left->SetNameString("Blade_revolute_left");
m_rev_left->Initialize(carrier, m_blade_body,
ChCoordsys<>(pos_left, rev_left_to_abs.GetRot() *
Q_from_AngX(CH_C_PI_2)));
chassis->GetSystem()->AddLink(m_rev_left);

m_rev_right =
chrono_types::make_shared<ChLinkLockRevolute>();
m_rev_right->SetNameString("Blade_revolute_right");
m_rev_right->Initialize(carrier, m_blade_body,
ChCoordsys<>(pos_right, rev_right_to_abs.GetRot() *
Q_from_AngX(CH_C_PI_2)));
m_rev_right->GetForce_X().SetActive(true);
chassis->GetSystem()->AddLink(m_rev_right);

m_lock_left = chrono_types::make_shared<ChLinkLockLock>();
m_lock_left->SetNameString("Blade_lock_left");
m_lock_left->Initialize(chassis->GetBody(), m_blade_body,
ChCoordsys<>(pos_lock_left, lock_left_to_abs.GetRot() *
```

```
Q_from_AngX(CH_C_PI_2)));

chassis->GetSystem()->AddLink(m_lock_left);


m_lock_right = chrono_types::make_shared<ChLinkLockLock>();

m_lock_right->SetNameString("Blade_lock_right");

m_lock_right->Initialize(chassis->GetBody(), m_blade_body,

ChCoordsys<>(pos_lock_right,  lock_right_to_abs.GetRot()  *
Q_from_AngX(CH_C_PI_2)));

     chassis->GetSystem()->AddLink(m_lock_right);
```

Листинг Б4 — Отображение информации о двигателе

```
auto powertrain = m_vehicle->GetPowertrain();

if  (powertrain)  {double  engine_rpm  =  powertrain-
>GetMotorSpeed()

* 60 / CH_C_2PI; sprintf(msg, "Eng.speed(RPM): %+.2f",

engine_rpm); renderLinGauge(std::string(msg),  engine_rpm  /
7000,

false, m_HUD_x, m_HUD_y + 50, 170, 15);

double engine_torque = powertrain->GetMotorTorque();

sprintf(msg, "Eng.torque(Nm): %+.2f", engine_torque);

renderLinGauge(std::string(msg), engine_torque / 600, false,
m_HUD_x, m_HUD_y + 70, 170, 15);
```

ПРИЛОЖЕНИЕ В

Исходный код функции main()

Листинг В1 – функция main ()

```
int main(int argc, char* argv[]) {
    if (use_track_bushings || use_suspension_bushings) {
        if (contact_method == ChContactMethod::NSC) {
            cout << "The NSC iterative solvers cannot be
used if bushings are present." << endl;
            return 1;
        }
    }

    if (shoe_type == TrackShoeType::DOUBLE_PIN &&
shoe_topology == DoublePinTrackShoeType::TWO_CONNECTORS) {
        if (!use_track_bushings) {
            cout << "Double-pin two-connector track shoes
must use bushings." << endl;
            return 1;
        }
    }

collision::ChCollisionSystemType collsys_type =
collision::ChCollisionSystemType::BULLET;
    CollisionType chassis_collision_type =
CollisionType::NONE;

    D12 d12;
    d12.SetContactMethod(contact_method);
    d12.SetCollisionSystemType(collsys_type);
    d12.SetTrackShoeType(shoe_type);
    d12.SetDoublePinTrackShoeType(shoe_topology);
    d12.SetTrackBushings(use_track_bushings);
    d12.SetSuspensionBushings(use_suspension_bushings);
    d12.SetTrackStiffness(use_track_RSDA);
    d12.SetDrivelineType(driveline_type);
    d12.SetBrakeType(brake_type);
    d12.SetPowertrainType(powertrain_type);
    d12.SetChassisCollisionType(chassis_collision_type);

    d12.SetChassisFixed(fix_chassis);
    d12.CreateTrack(create_track);


    d12.SetInitPosition(ChCoordsys<>(initLoc, initRot));
    d12.Initialize();

    auto& vehicle = d12.GetVehicle();

    VisualizationType track_vis =
```

```
(shoe_type == TrackShoeType::SINGLE_PIN) ?
VisualizationType::MESH : VisualizationType::PRIMITIVES;


d12.SetChassisVisualizationType(VisualizationType::MESH);

d12.SetSprocketVisualizationType(VisualizationType::MESH);
    d12.SetBladeVisualizationType(VisualizationType::MESH);

d12.SetBalancingBeamVisualizationType(VisualizationType::MES
H);

d12.SetTrolleyVisualizationType(VisualizationType::MESH);
    d12.SetIdlerWheelVisualizationType(track_vis);
    d12.SetRollerVisualizationType(track_vis);
    d12.SetTrackShoeVisualizationType(track_vis);

  // Create the 'deformable terrain' object
        vehicle::SCMTerrain mterrain(d12.GetSystem());

        mterrain.SetPlane(ChCoordsys<>(ChVector<>(-35.7, -2,
-0.1), Q_from_AngX(0)));

        // Initialize the geometry of the soil

        // Use either a regular grid:
        double length = 6;
        double width = 6;

mterrain.Initialize(vehicle::GetDataFile("terrain/height_map
s/bump64.bmp"), width, length, 0, 1, mesh_resolution);

         if (enable_bulldozing) {
            mterrain.EnableBulldozing(true);  // inflate
soil at the border of the rut
            mterrain.SetBulldozingParameters(
                55,   // angle of friction for erosion of
displaced material at the border of the rut
                1,    // displaced material vs downward
pressed material.
                5,    // number of erosion refinements per
timestep
                10);  // number of concentric vertex
selections subject to erosion
        }

        if (enable_moving_patch) {
            //mterrain.AddMovingPatch(mrigidbody,
ChVector<>(0, 0, 0), ChVector<>(0.5, 2 * tire_rad, 2 *
tire_rad));
        }
```

```
mterrain.SetPlotType(vehicle::SCMTerrain::PLOT_PRESSURE, 0,
30000.2);

        mterrain.SetMeshWireframe(true);


    std::shared_ptr<ChVehicleVisualSystem> vis;
    std::shared_ptr<ChDriver> driver;

    switch (vis_type) {
        case ChVisualSystem::Type::IRRLICHT: {
#ifdef CHRONO_IRRLICHT
            // Create the vehicle Irrlicht interface
            auto vis_irr =
chrono_types::make_shared<ChTrackedVehicleVisualSystemIrrlic
ht>();
            vis_irr->SetWindowTitle("D12 Vehicle Demo");
            vis_irr->SetChaseCamera(ChVector<>(0, 2, 1),
7.0, 0.5);

////vis_irr->SetChaseCameraPosition(vehicle.GetPos() +
ChVector<>(0, 2, 0));
            vis_irr->SetChaseCameraMultipliers(1e-4, 10);
            vis_irr->Initialize();
            vis_irr->AddLightDirectional();
            vis_irr->AddSkyBox();

vis_irr->EnableContactDrawing(ContactsDrawMode::CONTACT_DIST
ANCES);
            vis_irr->AddLightWithShadow(ChVector<>(2.5, 7.0,
0.0), ChVector<>(0, 0, 0), 50, 4, 25, 130, 512,

ChColor(0.8f, 0.8f, 0.8f));
            vis_irr->EnableShadows();
            //vis_irr->AddLogo();

            vis = vis_irr;

            if (driver_mode == DriverMode::KEYBOARD) {
                auto irr_driver =
chrono_types::make_shared<ChInteractiveDriverIRR>(*vis_irr);
                double steering_time = 0.5;  // time to go
from 0 to +1 (or from 0 to -1)
                double throttle_time = 1.0;  // time to go
from 0 to +1
                double braking_time = 0.3;   // time to go
from 0 to +1

irr_driver->SetSteeringDelta(render_step_size /
steering_time);
```

```
irr_driver->SetThrottleDelta(render_step_size /
throttle_time);
                irr_driver->SetBrakingDelta(render_step_size
/ braking_time);
                irr_driver->SetGains(2, 5, 5);
                driver = irr_driver;
            }

#endif
            break;
        }
        default:
        case ChVisualSystem::Type::VSG: {
#ifdef CHRONO_VSG
            // Create the vehicle VSG interface
            auto vis_vsg =
chrono_types::make_shared<ChTrackedVehicleVisualSystemVSG>()
;
            vis_vsg->SetWindowTitle("D12 Vehicle Demo");
            vis_vsg->SetChaseCamera(ChVector<>(0, 0, 0),
7.0, 0.5);
            vis_vsg->AttachVehicle(&d12.GetVehicle());
            ////vis_vsg->ShowAllCoGs(0.3);
            vis_vsg->Initialize();

            vis = vis_vsg;

            if (driver_mode == DriverMode::KEYBOARD) {
                auto vsg_driver =
chrono_types::make_shared<ChInteractiveDriverVSG>(*vis_vsg);
                double steering_time = 0.5;  // time to go
from 0 to +1 (or from 0 to -1)
                double throttle_time = 1.0;  // time to go
from 0 to +1
                double braking_time = 0.3;   // time to go
from 0 to +1

vsg_driver->SetSteeringDelta(render_step_size /
steering_time);

vsg_driver->SetThrottleDelta(render_step_size /
throttle_time);
                vsg_driver->SetBrakingDelta(render_step_size
/ braking_time);
                vsg_driver->SetGains(2, 5, 5);
                driver = vsg_driver;
            }

#endif
            break;
        }
```

```
}

    switch (driver_mode) {
        case DriverMode::DATAFILE: {
            auto data_driver =
chrono_types::make_shared<ChDataDriver>(vehicle,
vehicle::GetDataFile(driver_file));
            driver = data_driver;
            break;
        }
        case DriverMode::PATH: {


     auto endLoc = initLoc +
initRot.Rotate(ChVector<>(terrainLength, 0, 0));
            auto path =
chrono::vehicle::StraightLinePath(initLoc, endLoc, 50);
            auto path_driver =
std::make_shared<ChPathFollowerDriver>(vehicle, path,
"my_path", target_speed);

path_driver->GetSteeringController().SetLookAheadDistance(5.
0);

path_driver->GetSteeringController().SetGains(0.5, 0, 0);
            path_driver->GetSpeedController().SetGains(0.6,
0.3, 0);
            driver = path_driver;
            break;
        }
        default:
            break;
    }
    driver->Initialize();

    if (vehicle.GetNumTrackShoes(LEFT) > 0)
        std::cout << "Track shoe type: " <<
vehicle.GetTrackShoe(LEFT, 0)->GetTemplateName() <<
std::endl;
    std::cout << "Driveline type:  " <<
vehicle.GetDriveline()->GetTemplateName() << std::endl;
    std::cout << "Powertrain type: " <<
d12.GetPowertrain()->GetTemplateName() << std::endl;
    std::cout << "Vehicle mass: " << vehicle.GetMass() <<
std::endl;

    vis->AttachVehicle(&d12.GetVehicle());


    if
(!filesystem::create_directory(filesystem::path(out_dir))) {
```

```
std::cout << "Error creating directory " << out_dir <<
std::endl;
        return 1;
    }

    if (povray_output) {
        if
(!filesystem::create_directory(filesystem::path(pov_dir))) {
            std::cout << "Error creating directory " <<
pov_dir << std::endl;
            return 1;
        }

    }


 if (img_output) {
        if
(!filesystem::create_directory(filesystem::path(img_dir))) {
            std::cout << "Error creating directory " <<
img_dir << std::endl;
            return 1;
        }
    }

    double step_size = 1e-3;
    switch (contact_method) {
        case ChContactMethod::NSC:
            std::cout << "Use NSC" << std::endl;
            step_size = step_size_NSC;
            break;
        case ChContactMethod::SMC:
            std::cout << "Use SMC" << std::endl;
            step_size = step_size_SMC;
            break;
    }

    SetChronoSolver(*d12.GetSystem(), slvr_type,
intgr_type);

d12.GetSystem()->GetSolver()->SetVerbose(verbose_solver);

d12.GetSystem()->GetTimestepper()->SetVerbose(verbose_integr
ator);

    std::cout << "SOLVER TYPE:     " <<
(int)d12.GetSystem()->GetSolver()->GetType() << std::endl;
    std::cout << "INTEGRATOR TYPE: " <<
(int)d12.GetSystem()->GetTimestepper()->GetType() <<
std::endl;
```

```
BodyStates shoe_states_left(vehicle.GetNumTrackShoes(LEFT));
    BodyStates
shoe_states_right(vehicle.GetNumTrackShoes(RIGHT));
    TerrainForces
shoe_forces_left(vehicle.GetNumTrackShoes(LEFT));
    TerrainForces
shoe_forces_right(vehicle.GetNumTrackShoes(RIGHT));

    int render_steps = (int)std::ceil(render_step_size /
step_size);

    int step_number = 0;
    int render_frame = 0;

    auto trolley_link = vehicle.GetTrolleyLinkRight();
    trolley_link->GetForce_X().SetActive(true);

    while (vis->Run()) {

if (dbg_output) {
            auto track_L = vehicle.GetTrackAssembly(LEFT);
            auto track_R = vehicle.GetTrackAssembly(RIGHT);

            auto force_beam =
d12.GetBalancingBeam()->GetLinkLock()->Get_react_force();
            auto sum_beam_f = force_beam.x() +
force_beam.y() + force_beam.z();

            auto force_tr_r =
d12.GetTrolley()->GetRevoluteRight()->Get_react_force();
            auto sum_r = force_tr_r.x() + force_tr_r.y() +
force_tr_r.z();

            auto force_tr_l =
d12.GetTrolley()->GetRevoluteLeft()->Get_react_force();
            auto sum_l = force_tr_l.x() + force_tr_l.y() +
force_tr_l.z();

            cout << "      trolley force left:    " <<
force_tr_r <<  endl;
            cout << "      trolley force right:    " <<
force_tr_l <<  endl;
        }


        if (step_number % render_steps == 0) {

            vis->BeginScene();
            vis->Render();
            vis->EndScene();
```

```
if (povray_output) {
                char filename[100];
                sprintf(filename, "%s/data_%03d.dat",
pov_dir.c_str(), render_frame + 1);

utils::WriteVisualizationAssets(d12.GetSystem(), filename);
            }
            if (img_output && step_number > 200) {
                char filename[100];
                sprintf(filename, "%s/img_%03d.jpg",
img_dir.c_str(), render_frame + 1);
                vis->WriteImageToFile(filename);
            }
            render_frame++;
        }

        // Collect output data from modules
        DriverInputs driver_inputs = driver->GetInputs();
        vehicle.GetTrackShoeStates(LEFT, shoe_states_left);


      vehicle.GetTrackShoeStates(RIGHT, shoe_states_right);

        // Update modules (process inputs from other
modules)
        double time = vehicle.GetChTime();
        driver->Synchronize(time);
        mterrain.Synchronize(time);
        d12.Synchronize(time, driver_inputs,
shoe_forces_left, shoe_forces_right);
        vis->Synchronize(time, driver_inputs);

        // Advance simulation for one timestep for all
modules
        driver->Advance(step_size);
        mterrain.Advance(step_size);
        d12.Advance(step_size);
        vis->Advance(step_size);

        if (ReportTrackFailure(vehicle, 0.1)) {

            ReportConstraintViolation(*d12.GetSystem());
            break;
        }

        // Report if the chassis experienced a collision
        if
(vehicle.IsPartInContact(TrackedCollisionFlag::CHASSIS)) {
            std::cout << time << "  chassis contact" <<
std::endl;
        }
```

```
        step_number++;
    }

    vehicle.WriteContacts(out_dir + "/D12_contacts.out");

    return 0;
}
```