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.

Разработка и реализация математической модели почвы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.488 ПЗ ВКР

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

Автор работы,
студент группы КЭ-405
_____И. А. Лысиков
«___»_____2024 г.

Нормоконтролер,
ст. преподаватель. каф. ЭВМ
_____С. В. Сяськов
«___»_____2024 г.

Челябинск-2024

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

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

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Лысикову Ивану Александровичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы**: «Разработка и реализация математической модели почвы» утверждена приказом по университету от 22.04.2024 г. № 764-13/12

2. **Срок сдачи студентом законченной работы:** 1 июня 2024 г.

3. **Исходные данные к работе:**

В качестве реальной модели используется почва, состоящая из песка и супеси с параметрами:

а) плотность 1800 кг/м^3;

б) коэффициент Пуассона 0,33;

в) модуль Юнга 70 Мпа;

г) коэффициент поверхностного трения 0,91.

4. **Перечень подлежащих разработке вопросов:**

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

2. Выбор средств реализации проекта.
3. Проектирование и реализация модели.
4. Проведение тестирования поведения почвы и сбор результатов.

5. Дата выдачи задания: 01 декабря 2023 г.

Руководитель работы_____/*Ю. Г. Плаксина*/

Студент _____/*И. А. Лысиков*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы	01.03.2024	
Выбор средств реализации проекта	03.03.2024	
Проектирование и реализация модели	24.03.2024	
Проведение тестирования поведения почвы и сбор результатов	07.04.2024	
Подготовка презентации и доклада	15.05.2024	

Руководитель работы_____/*Ю. Г. Плаксина*/

Студент _____/*И. А. Лысиков*/

АННОТАЦИЯ

И. А. Лысиков. Разработка и реализация математической модели почвы. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 49 с., 15 ил., библиогр. список – 27 наим.

В рамках выпускной квалификационной работы осуществляется разработка и реализация математической модели почвы. Проводится анализ существующих проектов для понимания работы с различными методами моделирования. Определяются требования, предъявляемые к разрабатываемому продукту. Проводится сравнение сред разработки с целью выбора наиболее подходящего игрового движка, который соответствует поставленным требованиям. Разрабатывается проект, включая доработку математической модели, и реализуется на платформе Unity.

ОГЛАВЛЕНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения, обозначения и сокращения:

Игровой движок (от англ. game engine) – специализированное программное обеспечение, созданное для разработки и запуска компьютерных игр. Он предоставляет инструменты и функциональность для реализации различных аспектов игры, таких как графика и физика.

Тензор (от лат. tensus, «напряжённый») – применяемый в математике и физике математический объект линейной алгебры, заданный на векторном пространстве конечной размерности.

Terrain – это компонент, который позволяет создавать и редактировать трехмерную местность на основе высотной карты. Terrain представляет собой плоскую поверхность, покрытую высотной картой, которая состоит из трехмерных вершин и граней, которые образуют поверхность.

K-мерное дерево (от англ. k-dimensional tree) – это структура данных, используемая для эффективного поиска точек в k-мерном пространстве. k-мерное дерево разбивает пространство на подпространства с помощью гиперплоскостей и позволяет быстро и эффективно искать ближайшие точки к заданному запросу.

ВВЕДЕНИЕ

Создание современного симулятора тяжёлой строительной техники требует высокой реалистичности поведения почвы.

Для этого необходимо создать визуальное представление процесса разбиения почвы, которое будет максимально приближено к реальности. Внешний вид почвы должен изменяться в зависимости от приложенной нагрузки.

Метод конечных элементов (МКЭ) и метод дискретных элементов (МДЭ) – это два численных метода, которые применяются для моделирования сложных физических систем.

В данной выпускной квалификационной работе рассматривается преобразование МКЭ к МДЭ на примере физической модели почвы. Почва будет представлена в виде области, которая разбита на конечное количество элементов. Каждый элемент будет иметь форму квадрата. Эти элементы образуют сетку, зная координаты и высоту узлов которой, будет произведен расчёт деформации поверхности.

Метод дискретных элементов позволяет учитывать динамические взаимодействия между элементами. В данной работе он используется для отделения и переноса массы почвы в результате сгребания или копания.

Рассмотрим принципы обоих методов, а также их применение в моделировании поведения почвы. Это позволит нам лучше понять, как выбрать наиболее подходящий метод для конкретных задач и какие преобразования необходимы для перехода от одного метода к другому.

Цель работы – разработка и реализация математической модели почвы.

Для достижения поставленной цели необходимо решить следующий ряд задач:

– провести аналитический обзор существующих аналогов разрабатываемого проекта;

– выполнить анализ и выбор средств реализации проекта согласно

поставленным требованиям;

– разработать программный код и реализовать проект;

– провести моделирование поведения почвы.

Актуальность работы: создание визуального представления процесса разбиения почвы, которое будет максимально приближено к реальности и способно изменяться в зависимости от приложенной нагрузки. Это позволит обеспечить высокую степень реалистичности при моделировании поведения почвы в различных условиях.

1. АНАЛИТИЧЕСКИЙ ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ, НОРМАТИВНОЙ, МЕТОДИЧЕСКОЙ ЛИТЕРАТУРЫ И СУЩЕСТВУЮЩИХ АНАЛОГОВ

1.1. Аналитический обзор научно-технической, нормативной и методической литературы

Численное моделирование является неотъемлемой частью современного развития техники и науки. Для различных областей техники используются различные численные методы:

1. Метод конечных элементов.
2. Метод дискретных элементов.
3. Метод конечных разностей.
4. Метод границы элементов.
5. Метод конечных объемов.

Метод конечных элементов [1] – основан на разбиении сложной геометрической области на конечное число простых подобластей, так называемых конечных элементов. Затем, используя аппроксимацию и методы численного интегрирования, система уравнений приводится к матричному виду, которую можно решить для определения деформаций и напряжений в теле. Пример использования МКЭ представлен на рисунке 1.

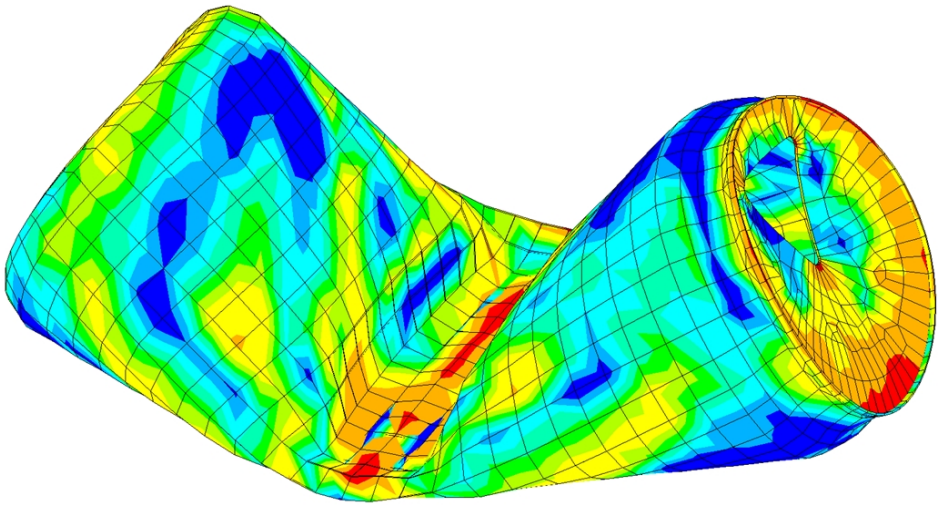

Рисунок 1 – Пример работы с МКЭ

Метод дискретных элементов [2] – численный метод, применяемый для моделирования и анализа поведения материалов и структур в механике твердого тела. Он основан на разделении твердого тела на набор дискретных элементов и моделировании их взаимодействия. Пример использования МДЭ представлен на рисунке 2.

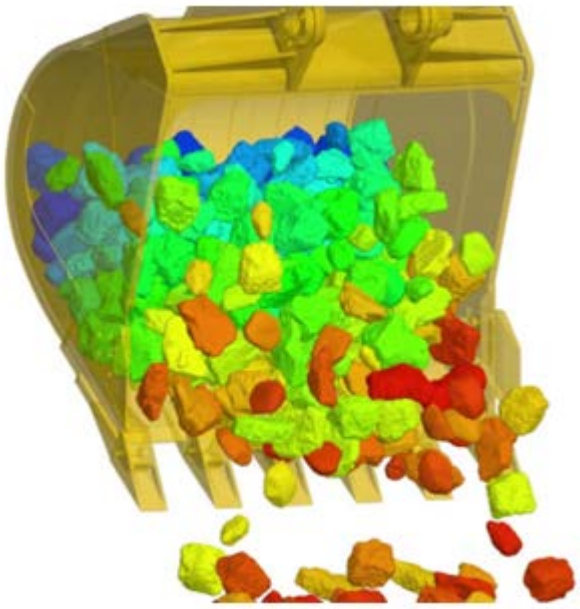

Рисунок 2 – Пример работы с МДЭ

Метод конечных разностей (МКР) [3]: В этом методе область задачи разбивается на сетку, а дифференциальные уравнения, описывающие твердое тело, аппроксимируются разностными выражениями на этой сетке. Затем система разностных уравнений решается и производится обратная интерполяция результатов на внутренние точки сетки для получения деформаций, напряжений и других величин.

Метод границы элементов (МГЭ) [4]: Данный метод комбинирует подходы МКЭ и МКР. Задача разбивается на внутренние конечные элементы, подобно МКЭ, а на границе тела используются конечные разности для аппроксимации граничных условий.

Метод конечных объемов (МКО) [5]: В этом методе расчетная область разбивается на объемные ячейки, и интегро-дифференциальные уравнения механики твердого тела аппроксимируются по законам сохранения в каждой

ячейке. Таким образом, значения деформаций, напряжений и других величин считаются средними по объему каждой ячейки.

В механике твердого тела ведущими методами являются метод конечных элементов и метод дискретных элементов. МКЭ строго выведен из теории сплошных сред и используется для описания деформируемых сплошных тел, в то время как МДЭ описывает материалы в виде частиц, обычно моделируемые идеально твердые частицы и их взаимодействия, определяемые по фиктивным наложениям этих твердых частиц.

Часто инженерную задачу можно смоделировать, используя только один из вышеупомянутых методов. Стальная балка может быть смоделирована с помощью МКЭ, небольшая сборка частиц гравия – с помощью МДЭ. Одним из возможных подходов смоделировать удар стального стержня о гравий было бы разделить проблему на две области. Стальная часть, смоделированная с помощью МКЭ, и гравийная часть, смоделированная с помощью МДЭ и далее соответствующим образом связать их.

В статье [6] описывается объединение двух бесплатных программ с открытым исходным кодом. Одна из них основана на методе конечных элементов (МКЭ) и представлена библиотекой OOFEM, а другая – на методе дискретных элементов (МДЭ) и реализована в библиотеке YADE. Обе программы имеют ядро, написанное на C++, что обеспечивает эффективное выполнение трудоёмких процедур.

Эта реализация служит только для тестирования методов и функционала, поэтому не все функции пока доступны в общедоступных версиях OOFEM и YADE. На рисунке 3 представлена иллюстрация контактных сил частиц сферической формы. На рисунке 4 показано взаимодействие поверхностей, представленных МКЭ и МДЭ.

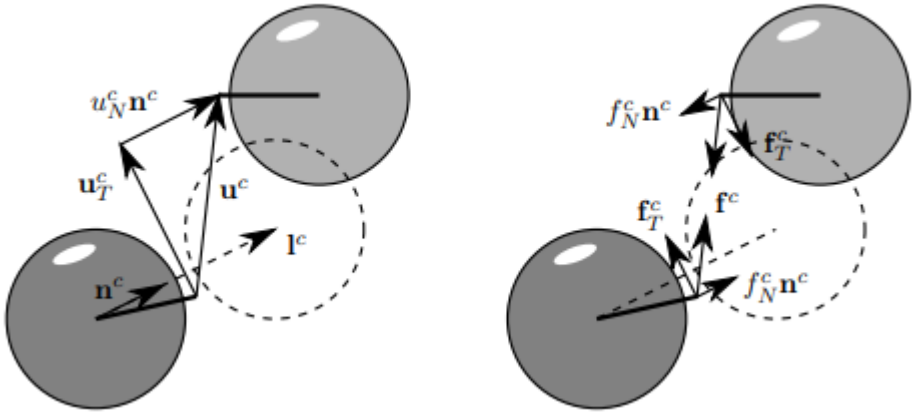

Рисунок 3 – Упрощенная иллюстрация контактных сил [6]

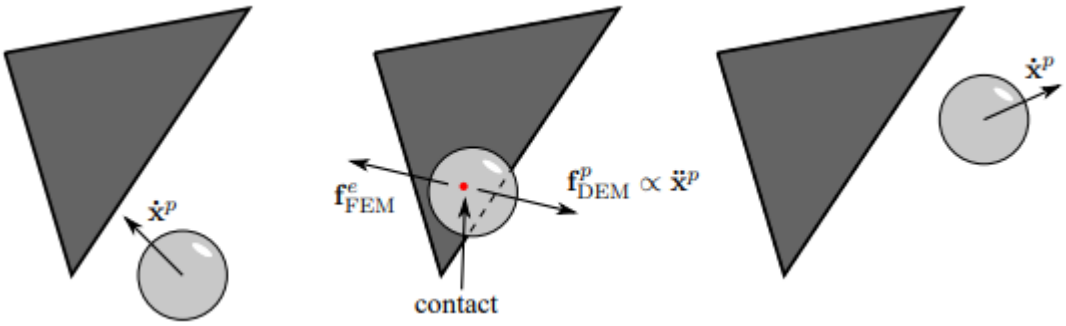

Рисунок 4 – Взаимодействие МКЭ/МДЭ поверхностей [6]

В статье [7] представлен общий обзор комбинированного использования МКЭ и МДЭ, который считается современным методом механического анализа каменных конструкций.

В процессе моделирования каждый каменный блок представляется в виде модели конечного элемента для описания его деформируемости. Статья подробно рассматривает основные этапы комбинированного метода, включая обнаружение контактов, контактное взаимодействие, алгоритмы разрушения и фрагментации, расчёт деформаций и интегрирование по времени уравнения движения.

На рисунке 5 показано использование комбинированного подхода для расчёта нагрузок арочного моста. Этот подход особенно полезен при моделировании перехода от конечных к дискретным процессам, которые могут привести к обрушению конструкции.

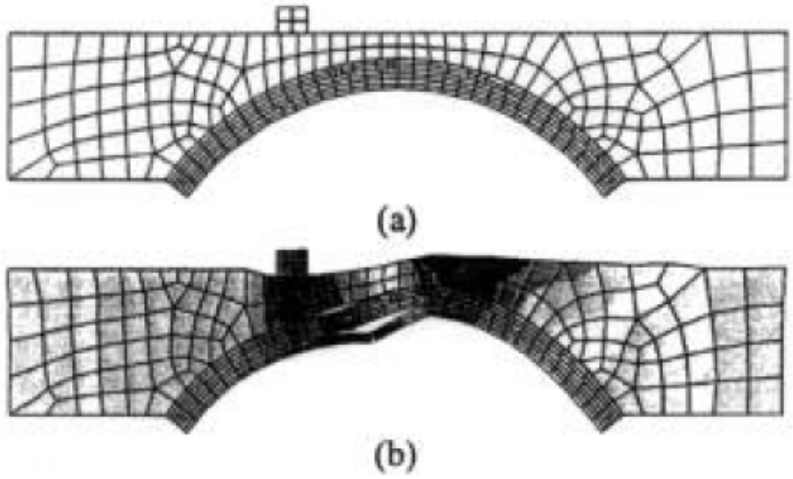

Рисунок 5 – Деформация арочного моста перед разрушением

В исследовании [8] описывается контактная модель, основанной на напряжении, отсутствующей в классическом МДЭ. Чтобы решить эту проблему, обобщается классический закон взаимодействия силы и смещения путем использования тензора напряжений частицы, чтобы сделать все контакты зависимыми от всех других контактов частицы и, таким образом, учесть множество контактов, одновременно действующих на одну частицу. Проведено сравнение результатов моделирования одноосного ограниченного сжатия с использованием новой многоконтактной модели с классической формулировкой МДЭ и существующей моделью на основе деформации. Расчет сил с учетом множества контактов представлено на рисунке 6.

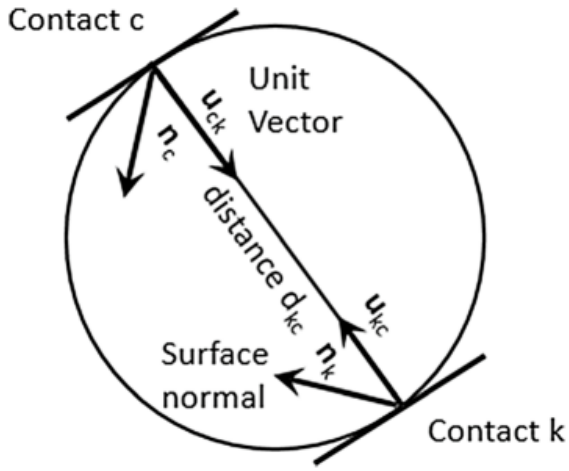

Рисунок 6 – Контакты одновременно действующих на одну частицу

Моделирование деформации грунта в режиме реального времени приведено в статье [9]. Осознавая сильные и слабые стороны подходов, основанных на частицах и сетке, была предложена гибридная модель, которая сочетает в себе оба метода.

Подход продиктован встречаемостью почв в природе: в равновесии, не нарушаемом внешними силами, почвы остаются в статичном состоянии, не проявляя никакого движения, в то время как, подвергаясь достаточно высокому напряжению, они проявляют высокую пластичность. Подход, основанный на использовании частиц без сетки, позволяет почве видоизменяться свободно и несвязанно. Это даёт возможность моделировать высокодинамичные явления поведения почвы. С другой стороны, представление грунта в статическом состоянии с помощью поля высот позволяет быстро обнаруживать столкновения и легко визуализировать модель.

Смоделированный грунт рассматривается как неоднородный, учитывая степень уплотнения грунта в каждом положении. Подходе соединяет поле высот для представления почвы в ее статическом состоянии и с основанным на МДЭ моделированием почвы в ее рыхлом динамическом состоянии, представлением частицами почвы. Почвенная сетка показана на рисунке 7.

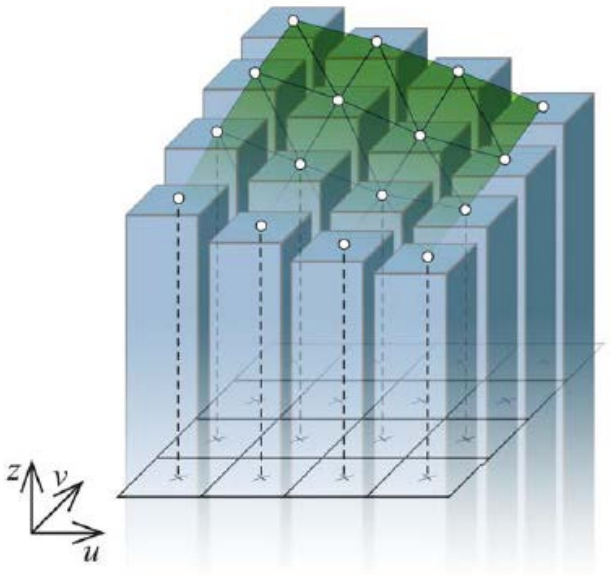

Рисунок 7 – Модель представления почвенной сетки

Непрерывный массив почвы аппроксимируется в виде массива прямоугольных столбцов почвы, каждый из которых представлен значением высоты. Эти данные хранятся в двумерной сетчатой структуре.

Основная концепция представленной модели заключается в том, чтобы адаптивно обеспечить деформируемость большой площади, покрытой почвой. Это достигается путём замены частей почвенной сетки частицами почвы. Благодаря этому почва может свободно перемещаться в областях, которые подвергаются деформации.

Когда частицы почвы осядут и достигнут равновесия, они будут объединены обратно в почвенную сетку с сохранением объёма. Такой подход значительно уменьшает количество частиц, необходимых для моделирования, что позволяет использовать метод в режиме реального времени.

В работе [10] представлены новые адаптивные алгоритмы выборки для моделирования жидкости на основе частиц. Введено новое условие выборки, которое основано на геометрическом размере локального объекта. Это позволяет сосредоточить вычислительные ресурсы в сложных областях, одновременно уменьшая количество частиц глубоко внутри жидкости или вблизи её поверхности.

Дополнительный прирост производительности достигается за счёт изменения плотности отбора проб в зависимости от визуальной значимости. Также предложено новое определение поверхности жидкости, основанное на приблизительном соотношении частиц к расстоянию до поверхности. Эти данные переносятся вместе с частицами и соответствующим образом обновляются.

Полученный метод реконструкции поверхности имеет ряд преимуществ перед существующими методами, включая стабильность при повторной выборке частиц и пригодность для представления гладких поверхностей.

В статье [11] описаны два основных подхода к моделированию дискретных элементов, называемых гладким (smooth) и негладким (nonsmooth). Разница заключается в том, устраняется ли вязкоупругая природа контактов во времени или нет.

Негладкий подход рассматривает столкновения и фрикционные переходы "прилипание-проскальзывание" как мгновенные события, при которых скорость может изменяться скачкообразно во времени в соответствии с заданным законом контакта. Это позволяет выполнять интеграцию с большим временным шагом и потенциально требует значительно меньших вычислительных затрат.

Сделан вывод, что негладкий подход наиболее благоприятен для плотных квазистатических систем с небольшим количеством частиц и высоким отношением жесткости материала к массе элемента, в то время как гладкий подход становится все более благоприятным с увеличением числа частиц, кинетической энергии и уменьшением жесткости материала

В исследовании [12] представлена модель взаимодействия вращающегося ножа с почвой, разработанная на основе метода дискретных элементов. С помощью этой модели были определены факторы, влияющие на энергопотребление и качество работы вращающегося ножа.

В процессе моделирования был учтён закон взаимодействия вращающейся лопасти с почвенным покровом. Точность имитационной модели была подтверждена полевыми испытаниями.

В статье [13] представлен новый вычислительный метод для прогнозирования возникновения и развития разрушения в сплошной среде простым способом, сочетающим метод конечных элементов и метод дискретных элементов. Начало разрушения в определенной точке определяется простой моделью повреждения. Как только трещина обнаруживается на стороне элемента в сетке МКЭ, в узлах, разделяющих одну сторону, генерируются дискретные элементы, и считается, что простой механизм следует за эволюцией трещины. Сочетание МДЭ с простыми

трехузловыми линейными треугольными элементами правильно фиксирует начало разрушения и его развитие, как показано в нескольких примерах применения в двух и трех измерениях.

В статье [14] описана инновационная модель фрикционного контакта, которая позволяет эффективно моделировать взаимодействие между частицами неправильной формы. В отличие от неё, традиционная модель силового контакта ближнего действия формирует поле контактных сил таким образом, что между частицами остаётся большой зазор. Это приводит к более мягкой жёсткости контакта и неточному моделированию плотно упакованных гранулированных материалов.

Разработанная модель контакта с учетом прикосновения инициирует расчет контактной силы при соприкосновении контактирующих тел, что не оставляет зазора между контактирующими телами и обеспечивает более жесткий контакт точным и эффективным способом. Он моделирует поведение фрикционного контакта и демпфирование между контактирующими телами неправильной формы.

Приводятся различные примеры для проверки модели контакта с учетом прикосновения с помощью теоретических решений, таких как теория контакта Герца, теория удара Герца, закон Кулона и формулировка демпфирования.

В статье [15] представлена реализация деформируемого грунта в реальном времени, основанную на модели контакта с почвой, разработанной в Немецком аэрокосмическом центре, которую, в свою очередь, можно рассматривать как обобщение полуэмпирических моделей почвы Беккера-Вонга и Янози-Ханамото. взаимодействие с произвольными трехмерными формами и произвольными пятнами контакта. Описана общая реализацию и особенность модели, эффективные базовые структуры данных, текущие аспекты многоядерного распараллеливания и ее свойства масштабируемости для одновременного моделирования нескольких транспортных средств на деформируемой местности.

1.2. Обзор аналогов разрабатываемой модели

Обзор аналога будет осуществлён по следующему ряду критериев:

1. Отечественная разработка.
2. Открытый исходный код.
3. Кроссплатформенность.
4. Возможность использования как сторонней библиотеки.

AGX Dynamics [16] многофункциональный физический движок для симуляторов в промышленных приложениях, подготовленный как для использования вместе с инструментами САПР, так и с игровыми движками, такими как Unity. Ядро движка представляется собой библиотеку, написанную на языке C++. Данное решение обеспечивает высокую точность, стабильность и скорость для приложений реального времени. В таблице 1 приведено сравнение с существующим аналогом.

Таблица 1 – Сравнительный анализ существующего аналога

	Отечественная разработка	Возможность использовать на территории РФ	Кроссплатформенность	Возможность использования как сторонней библиотеки
Собственная разработка	+	+	-	-
Algoryx (AGX Dynamics)	-	-	+	+

1.3. Вывод

Задача разработки и реализации математической модели почвы является частью большой системы, представляющей собой виртуальный полигон для обкатки новых образцов техники и обучения персонала.

Для решения задачи моделирования поведения почвы при контакте с колесной или гусеничной техникой будет использована гибридная модель, которая сочетает в себе МДЭ для моделирования поведения сыпучей породы и МКЭ для моделирования сплошной почвенной сетки.

2. АНАЛИЗ И ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ ПРОЕКТА СОГЛАСНО ПОСТАВЛЕННЫМ ТРЕБОВАНИЯМ

2.1. Функциональные требования

Определим следующие функциональные требования:

- при сдвиговой деформации почвенная сетка преобразуется в частицы сферической формы;
- для отдельных частиц рассчитывается сила их взаимодействия;
- как только частицы достигли равновесия они объединяются с почвенной сеткой.

2.2. Нефункциональные требования

Определим следующие нефункциональные требования:

- платформа: Unity 2023.2.6f1;
- операционная система: Windows 10 и новее.

2.3. Выбор средств реализации

В обычном методе дискретных элементов нормальная контактная сила является функцией величины перекрытия и скорости его изменения.

Одной из распространенных функций является нелинейная модель Герца, которая следует из теории линейных вязкоупругих материалов [17]. Силы, возникающие в результате контакта между двумя телами, представлены на рисунке 8.

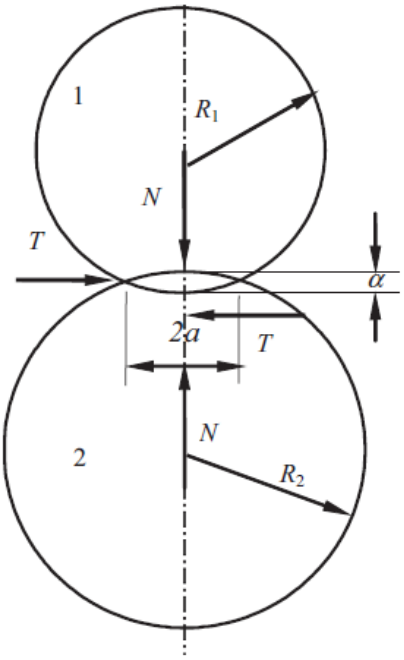

Рисунок 8 – Иллюстрация контакта между двумя телами

Модель нормальной контактной силы включает нормальную отталкивающую силу ($f_{el}{}^{n}$) и нормальную диссипативную силу ($f_{visc}{}^{n}$) [18]. Расчет нормальной контактной силы представлен в формуле (1):

$$\mathrm{f}_n = f_{el_n} + f_{visc_n} = \frac{4}{3} * E^* * \sqrt{r_{ij}} * \delta^{\frac{3}{2}}{}_n + \eta_n * \sqrt{r_{ij} * \delta_n} * \delta_n{}^*, \qquad (1)$$

где $r_{ij} = \frac{r_i r_j}{r_i + r_j}$ – действительный радиус, $\delta_n = \left(r_i + r_j\right) - \left(x_i - x_j\right) * n > 0$ – взаимное перекрытие сфер, E^*- действительный модуль Юнга, $\frac{1}{E^*} = \frac{1-\nu_i}{2G_i} + \frac{1-\nu_j}{2G_j}$, где ν представляет коэффициент Пуассона частицы, E – модуль Юнга, а $G = \frac{E}{2(1+\nu)}$ – модуль сдвига соответственно [8].

В действительности столкновения частиц неупругие и часть энергии при взаимодействии рассеивается. В рассматриваемой модели величина рассеивания связана с относительной скоростью частиц в нормальном направлении ($\delta_n{}^* = v_{rel_n} = -\left(v_i - v_j\right) * n$) и величины η_n – постоянной вязкоупругого демпфирования для нормальной контактной вязкости, которая является собственным параметром материала сферы.

Трение моделируется как движение пружины в тангенциальном направлении. Тангенциальное растяжение пружины вычисляется путем интегрирования скорости скольжения, и сила проецируется на конус трения в соответствии с законом Кулона [19]. Аналогично, сопротивление качению моделируется как крутящий момент, противодействующий относительному движению качения, также ограниченный законом, подобным Кулоновскому. Расчет контактных сил представлен в формулах (2) – (3):

$$\mathrm{f}_t = proj_{\mu|f_n|}\left(\int k_t * u_t * dt\right), \tag{2}$$

$$\tau_r = proj_{\mu_r|\mathrm{f}_n|}\left(\int k_r * w * dt\right), \tag{3}$$

где u и w представляют собой относительную тангенциальную скорость и относительную угловую скорость в точке контакта. Коэффициент жесткости при трении, k_t и жесткость сопротивления качению, k_r, не связаны с фундаментальными параметрами материала и должны определяться путем настройки из экспериментов.

Другим популярным способом расчета контактных сил является модель Беккера [20] как одна из наиболее широко используемых моделей для прогнозирования поведения однородного грунта при сжатии. Формула (4) обеспечивает взаимосвязь между давлением и вертикальной деформацией грунта в виде:

$$p = \left(\frac{k_c}{b} + k_\gamma\right) * y^n, \tag{4}$$

где p – давление в пятне контакта, y – просадка колеса, k_c – эмпирический коэффициент, представляющий когезионный эффект грунта, k_γ – эмпирический коэффициент, отражающий жесткость грунта. Параметр n – показатель степени, выражающий эффект упрочнения, который увеличивается при уплотнении почвы.

Параметр b представляет собой длину самой короткой стороны прямоугольного пятна контакта. Это связано с тем, что первоначальная

теория Беккера предполагает, что цилиндрическая шина катится по ровной поверхности.

Однако величина b оценивается путём совмещения всех связанных контактных пятен. В результате, используя алгоритм затопления и аппроксимацию, получаем результирующее пятно контакта. Пятно контакта рассчитывается по формуле (5):

$$b \approx \frac{2 * A}{L}, \tag{5}$$

где A – площадь такого пятна контакта, а L – его периметр.

Для тангенциальной величины силы, которая представляет собой модель отклика на горизонтальный сдвиг, была выбрана модель Яноши – Ханамото [21]. Эта модель лежит в основе большинства современных моделей динамики поведения почвы и представляет собой аналитическое уравнение касательного напряжения и сдвигового смещения, которое представлено в формуле (6).

$$\tau = (c + p * \tan(\varphi)) * \left(1 - e^{-\frac{j}{k}}\right), \tag{6}$$

где τ- напряжение сдвига, j – смещение грунта при сдвиговой деформации, k – постоянный модуль сдвиговой деформации, c – когезионное напряжение, φ – внутренний угол трения грунта, p – давление в пятне контакта, описанное формулой (4).

Для расчёта контактных сил была выбрана нелинейная модель Герца, поскольку она специально разрабатывалась для определения сил, возникающих при контакте между двумя телами. В дополнение к нормальной контактной силе, описываемой формулой Беккера, модель Герца также учитывает нормальную диссипативную силу.

2.4. Выбор среды разработки

Проанализируем наиболее популярные физические движки:

1. Unity.

2. Unreal Engine 5.

3. Project Chrono.

Unity [22] – кроссплатформенная среда разработки компьютерных игр, созданная американской компанией Unity Technologies.

Достоинства:

– Unity предоставляет интуитивный интерфейс и хорошую документацию, что облегчает начало работы;
– Unity предоставляет гибкий редактор для создания миров, анимации и игровых сцен;
– объекты в Unity содержат наборы компонентов, с которыми взаимодействуют скрипты.

Недостатки:

– графические возможности Unity не всегда на уровне Unreal Engine;
– некоторые аспекты оптимизации могут быть сложными для новичков.

Unreal Engine 5 [23] – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games.

Достоинства:

– Unreal Engine 5 предлагает потрясающие графические возможности, включая реалистичное освещение и детализацию;
– Unreal Engine имеет мощную систему физики.

Недостатки:

– Unreal Engine может быть сложным для новичков из-за большого количества функций;
– для работы с Unreal Engine 5 требуется мощное оборудование.

Project Chrono [24] — это физический движок с открытым исходным кодом, предназначенный для моделирования и симуляции.

Достоинства:

– предоставляет инфраструктуру для моделирования различных физических систем;

– Project Chrono реализован на C++ и может быть встроен в различные проекты;
– существует версия PyChrono, которая позволяет использовать Chrono с помощью языка Python;
– Project Chrono может использоваться для больших симуляций, таких как гранулярные потоки, взаимодействие транспортных средств с грунтом и взаимодействие жидкости с твердыми телами;
– Project Chrono разработан как промежуточное ПО, которое можно настраивать и встраивать в другое программное обеспечение.

Недостатки:

– Project Chrono имеет меньшее сообщество и меньше ресурсов по сравнению с Unity и Unreal Engine;
– имеет ограниченные графические возможности;
– длительное время выполнения кода по сравнению с популярными игровыми движками.

Вывод по разделу

Сравнив сильные и слабые стороны физических движков, был сделан вывод, что Unity является лучшим выбором для решения поставленной задачи: наличие подробной технической документации и гибкий редактор сцен упростит разработку модели.

Для расчёта контактных сил была выбрана нелинейная модель Герца, которая наиболее точно соответствует цели - определению контактных усилий.

3. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ МОДЕЛИ

3.1. Доработка математической модели

Модель Герца в стандартном виде не подходит для моделирования с большим шагом выполнения программы, равным 0,02 секунды. Из-за длительного промежутка времени, затрачиваемого на моделирование, накапливается ошибка, поскольку основной характеристикой при расчёте сил соударяющихся объектов является их взаимное перекрытие.

Приведем выбранную модель Герца к виду стандартного неявного интегрирования. Это позволит рассматривать удары и переходы от трения к проскальзыванию как мгновенные события, которые делают скорости непостоянными во времени.

Преимущество такого подхода заключается в том, что она позволяет интегрировать с гораздо большим шагом моделирования, чем модель, использующая явный метод дискретных элементов, и, следовательно, потенциально быстрее.

Рассмотрим точечную частицу единичной массы, потенциал которой $U = \frac{1}{2*\varepsilon} * x^2$, где $\varepsilon -$ условие регуляризации [25]. Уравнение скорости и положения частицы представлено в формулах (7) – (8):

$$u_{i+1} = u_i + h * \frac{dU}{dx} = u_i - h * \varepsilon^{-1} * x_i. \quad (7)$$

$$x_{i+1} = x_i + h * u_{i+1}, \quad (8)$$

где h - временной шаг выполнения программы.

Приведем формулу (7) к виду стандартного неявного интегрирования [26]. Результат представлен в формуле (9):

$$\left[1 + \frac{h^2}{4 * \varepsilon}\right] * u_{i+1} = \left[1 - \frac{h^2}{4 * \varepsilon}\right] * u_i - \frac{h}{\varepsilon} * x_i. \quad (9)$$

Переменная ε, находясь в знаменателе, будет вызывать ошибку в пределе $\varepsilon \to 0$. Введем вспомогательную переменную $\lambda = \varepsilon^{-1} * x$. После введения новой переменной получаем формулу (10):

$$\begin{bmatrix} 1 & -1 \\ 1 & \frac{4*\varepsilon}{h^2} \end{bmatrix} * \begin{bmatrix} u_{i+1} \\ h*\lambda \end{bmatrix} = \begin{bmatrix} u_i + h*\lambda \\ -\frac{4}{h} * x_i - u_i \end{bmatrix}. \quad (10)$$

Чтобы рассмотреть общую механическую систему введем матрицу g, содержащую величину перекрытия между контактирующими телами. N_p – количество объектов и N_c – количество контактов. Тогда размерность матрицы M, хранящую информацию о массе объектов будет $\dim(M) = 6 * N_p \times 6 * N_c$, а размерность матрицы $\dim(g) = N_c \times 1$. Потенциал для общей механической системы $U = \frac{1}{2*\varepsilon} * g^T * g$, где $g(x)$ это функция от x, такая, что матрица Якоби $G = \frac{dg}{dx}$.

Уравнение скорости для общей механической системы представлено в формуле (11):

$$\left[M + \frac{h^2}{4*\varepsilon} * G^T * G\right] * u_{i+1} = \left[M - \frac{h^2}{4*\varepsilon} * G^T * G\right] * u_i - \frac{h}{\varepsilon} * G^T * g_i. \quad (11)$$

После введения новой переменной $\lambda = \varepsilon^{-1} * x$ формула (12) имеет вид:

$$\begin{bmatrix} M & -G^T \\ G & \frac{4*\varepsilon}{h^2} \end{bmatrix} * \begin{bmatrix} u_{i+1} \\ \lambda \end{bmatrix} = \begin{bmatrix} u_i + h*f_S \\ -\frac{4}{h} * g_i - G_i * u_i \end{bmatrix}, \quad (12)$$

где f_S – внешние силы, действующие на систему.

Используя формулу (12), мы можем найти скорость объекта после контакта, в зависимости от приложенных внешних сил, а значит и обновить его положение на сцене. Контактная сила будет равна $f = \frac{G*\lambda}{h}$.

Чтобы применить модель Герца, мы представим условие регуляризации через нормальную силу, которая содержится в формуле (1). Тогда $\varepsilon = \frac{10}{6} * E^* * \sqrt{r_{ij}}$, где r_{ij} – действительный радиус, E^*-действительный модуль Юнга [25]. Исходный код функции расчета контактных усилий представлена в листинге А1 приложения А.

3.2. Обнаружение контактов

Контакт двух сферических объектов происходит, когда величина взаимного перекрытия $g \geq 0$. Иллюстрация контакта между двумя телами представлена на рисунке 9.

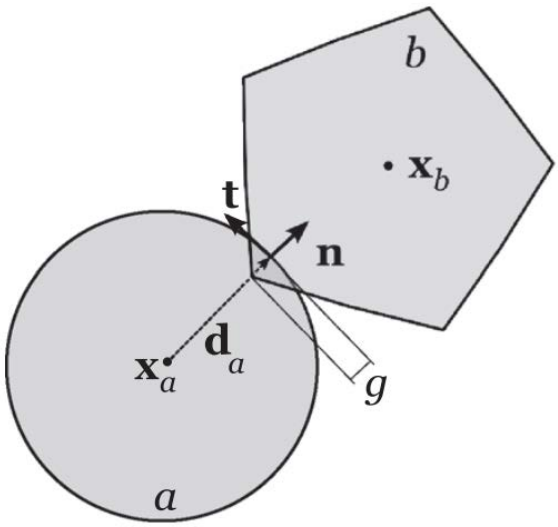

Рисунок 9 – Иллюстрация контакта между двумя телами

Расчет величины взаимного перекрытия представлен в формуле (13):

$$g_a = \overrightarrow{n_a} * \left(\overrightarrow{x_a} + \overrightarrow{d_a} - \overrightarrow{x_b} - \overrightarrow{d_b}\right), \quad (13)$$

где $\overrightarrow{n_a}$ – вектор нормали для объекта a, $\overrightarrow{d_a}$ – позиция точки контакта на поверхности объекта a, $\overrightarrow{x_a}$ – позиция объекта a.

Для каждого цикла выполнения программы необходимо, зная положения объектов, считать их взаимное перекрытие. Это очень трудоёмкая операция, поэтому для обнаружения потенциальных контактов используется k-мерное дерево [27].

K-мерное дерево разбивает пространство на подпространства с помощью гиперплоскостей и позволяет быстро и эффективно искать ближайшие точки к заданному запросу.

Для начала нужно построить k-мерное дерево. Зададим размер массива, в котором будут храниться отслеживаемые объекты, и максимальное количество объектов в подпространстве. Если количество объектов в подпространстве превышает максимально допустимое количество, оно

повторно делится на ещё меньшее пространство, сужая тем самым диапазон поиска возможных контактов. Исходный код функции инициализации k-мерного дерева представлена в листинге А2 приложения А.

Имея информацию о местоположении всех отслеживаемых объектов на сцене и зная радиус сфер, создадим очередь для поиска соседних объектов. Каждый объект будет помещаться в очередь вместе со своими координатами и радиусом, а объектами-соседями будут считаться все те, которые находятся в пределах этого радиуса.

Из найденных объектов-контактов формируются пары, которые затем добавляются в список. Чтобы избежать дублирования пар, перед добавлением новой пары проверяется её наличие в списке. Исходный код функции поиска контактных пар и их записи в список приведён в листинге А3 приложения А.

3.3. Архитектура проекта

Для создания проекта мы будем использовать гибридный метод моделирования почвы, который объединяет метод конечных элементов, представленный в виде сетки, и метод неявной дискретизации с элементами, представленный частицами. Это позволит нам использовать преимущества каждого из методов.

Частицы создаются в том случае, если почвенная сетка оказывается в зоне деформации. Если ранее созданные объекты покидают эту зону, они объединяются с почвенной сеткой, тем самым увеличивая её высоту. Блок-схема процесса моделирования представлена на рисунке. 10.

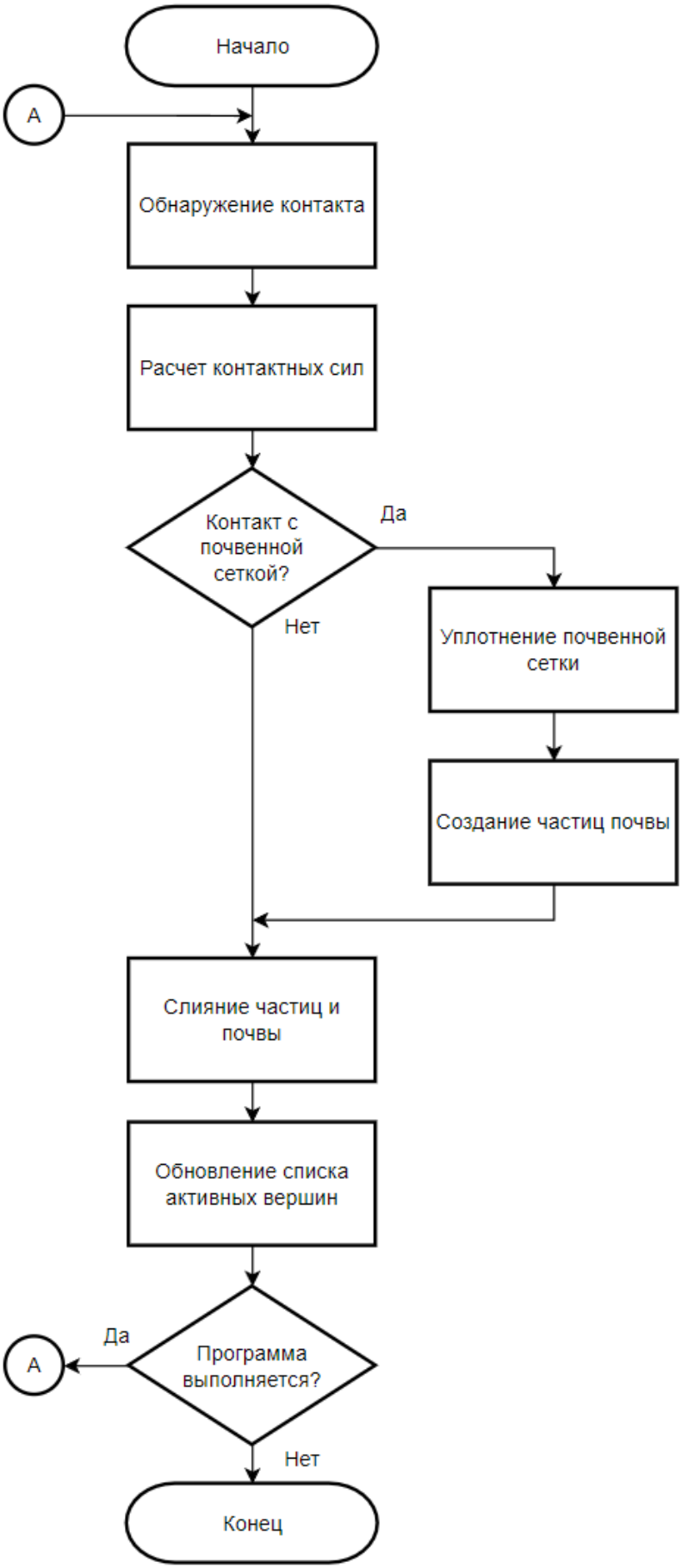

Рисунок 10 – Блок-схема процесса моделирования

Когда частица почвы, которая соприкоснулась с почвенной сеткой, достигает состояния равновесия, то есть значения её угловой и линейной скорости становятся меньше определённого порогового значения, её движение уже не оказывает существенного влияния на динамику почвы. В этом случае мы удаляем эту частицу из симуляции и заменяем её статическим объёмом в почвенной сетке, увеличивая высоту соответствующей толщи почвы.

Такой подход значительно снижает общее количество частиц в моделировании, не ограничивая при этом динамическое поведение почвы. Это, в свою очередь, вносит значительный вклад в возможности метода для работы в режиме реального времени.

Диаграмма классов, представленная на рисунке 11, отображает структуру проекта, показывая названия классов, их поля и методы. Также она демонстрирует связи между классами.

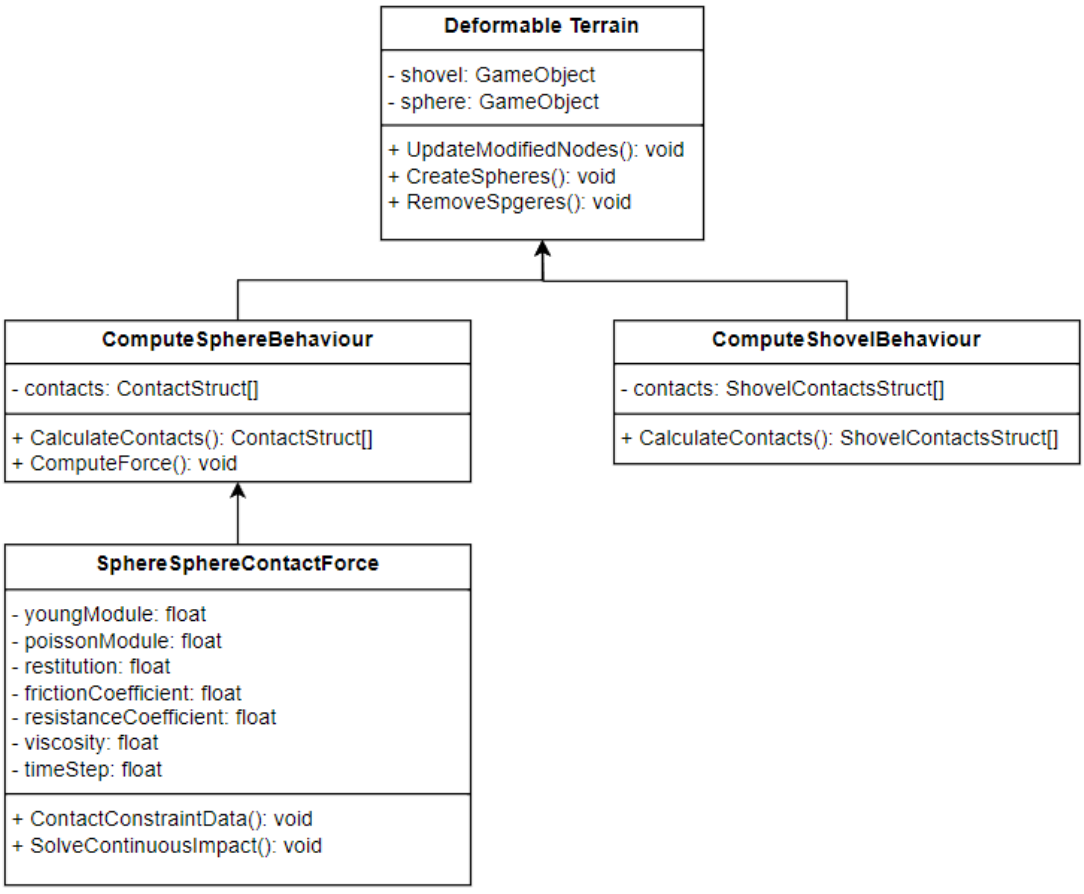

Рисунок 11 – Диаграмма классов

3.4. Реализация на платформе Unity

При создании нового проекта в Unity необходимо добавим в него компонент Terrain, представленный через МКЭ. Этот компонент будет отвечать за создание и управление трёхмерной местностью (ландшафтом) в процессе моделирования.

Поверхность, созданная с помощью Terrain, представляет собой сетку, где каждая ячейка сетки записывается в массив и имеет свой индексы, а также значение высоты. Изменяя высоту ячеек, мы можем менять высоту ландшафта. Это позволяет нам создавать разнообразные ландшафты, такие как горы, долины или равнины. Результат представлен на рисунке 12.

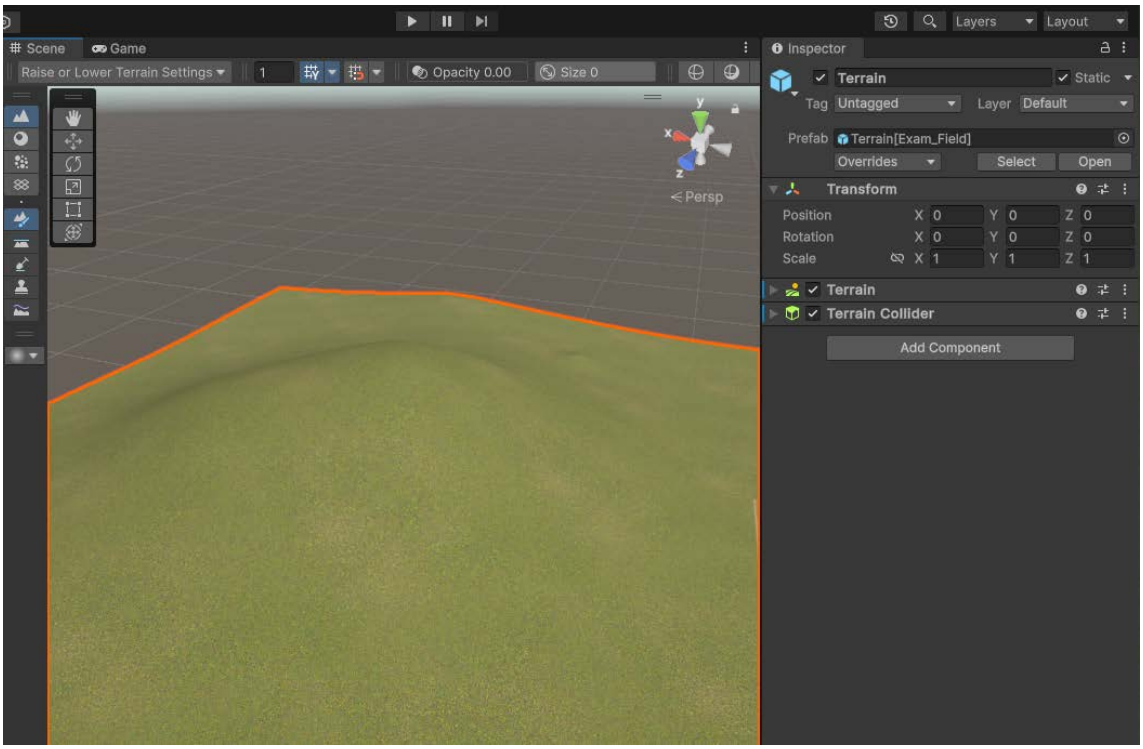

Рисунок 12 – Компонент terrain

Для того чтобы оказывать внешнее воздействие на поверхность, созданную с помощью компонента Terrain, добавим модель гусеничной техники, имеющую отвал. Отвал будет использоваться для взаимодействия с ландшафтом и изменения его формы. Модель гусеничной техники представлена на рисунке 13.

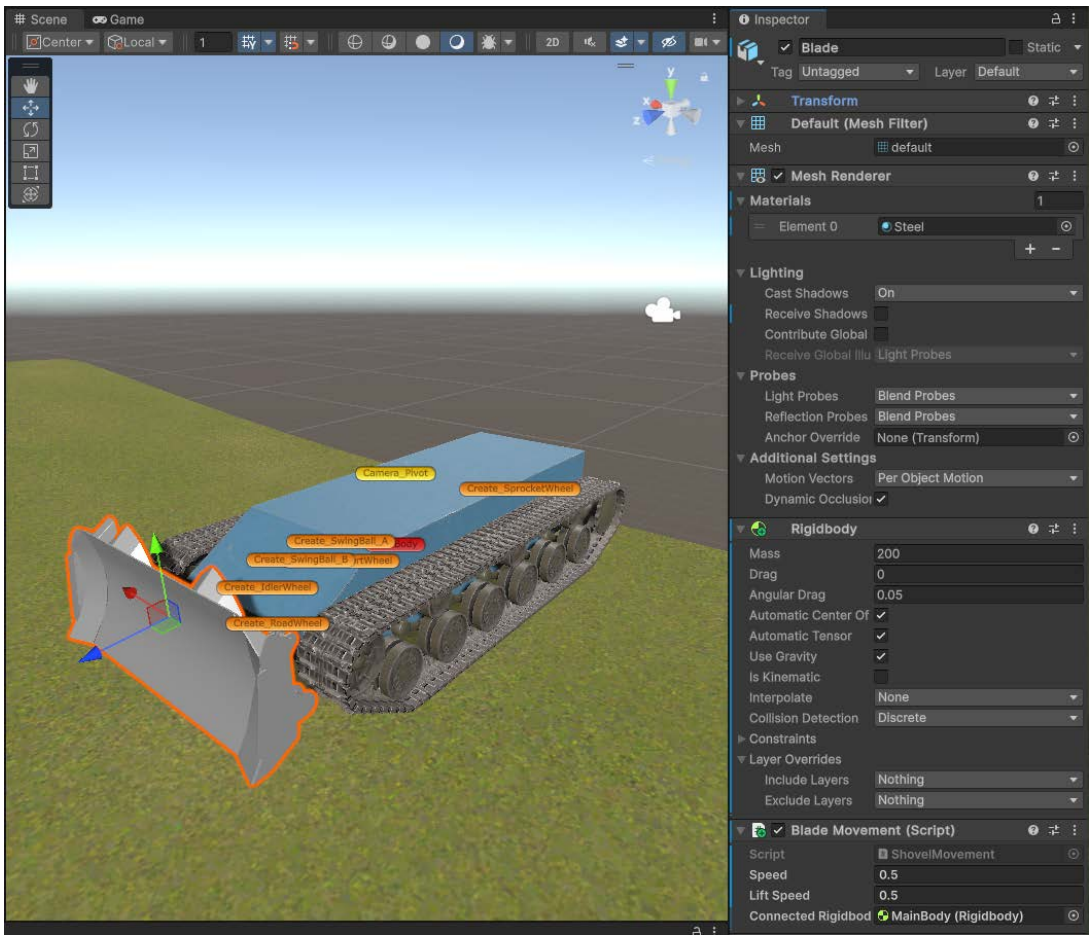

Рисунок 13 – Модель гусеничной техники

Для поднимания и опускания отвала создадим скрипт и добавим его к модели отвала. Эти функции будут вызываться при нажатии соответствующих кнопок на клавиатуре или при использовании других методов ввода. Исходный код функций подъёма и опускания отвала представлен в листинге А4 приложения А. Эти функции используют свойства компонента Transform для изменения положения отвала по оси Y.

После создания скрипта добавим его к компоненту отвала в инспекторе Unity. Затем станет возможным использование кнопок на клавиатуре для управления движением отвала вверх и вниз. Таким образом, создана модель гусеничной техники с возможностью взаимодействия с поверхностью Terrain. Модель гусеничной техники с опущенным отвалом представлена на рисунке 14.

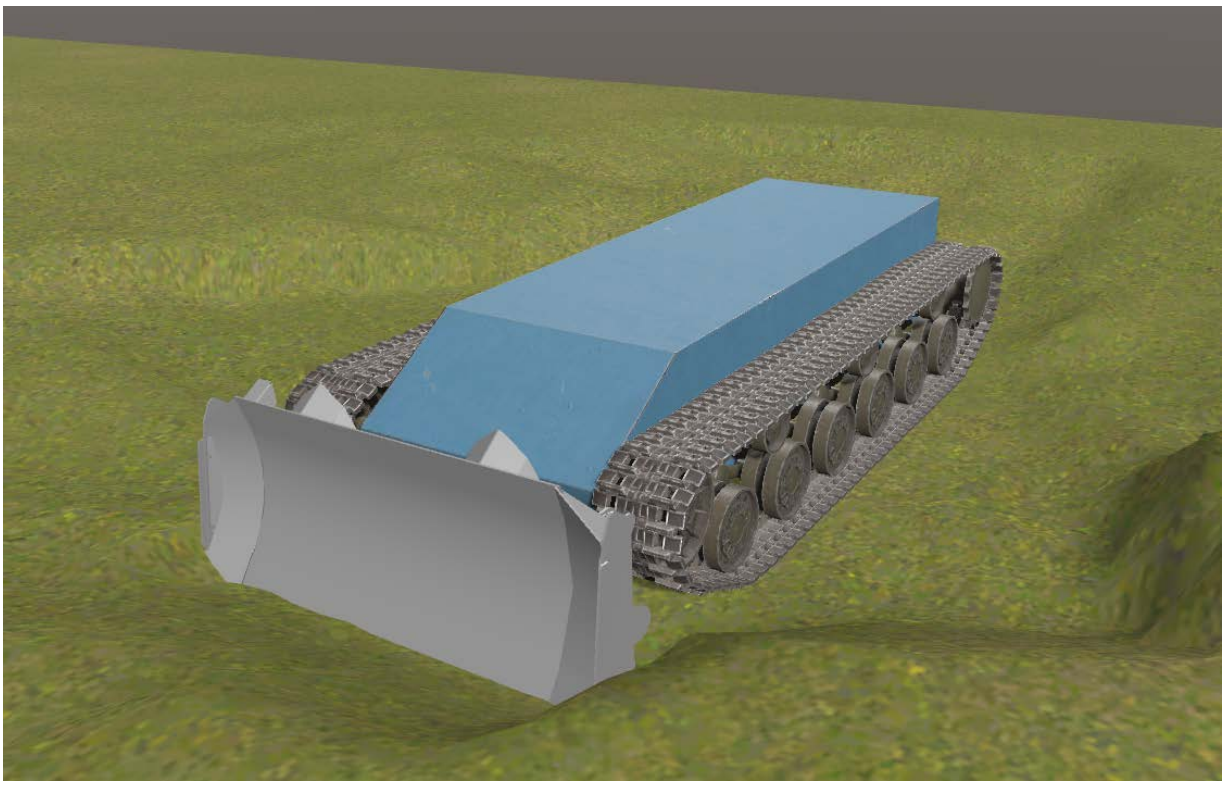

Рисунок 14 – Модель гусеничной техники с опущенным отвалом

В процессе взаимодействия отвала гусеничной техники с поверхностью, созданной с помощью компонента Terrain, происходит уплотнение почвенной сетки. Это приводит к изменению рельефа местности и созданию новых форм ландшафта.

При этом в зоне деформации почвенная сетка заменяется отдельными частицами сферической формы. Эти частицы представляют собой небольшие трёхмерные объекты, которые имитируют различные типы поверхностей, такие как камни, песок или грязь. Процесс создания частиц почвы представлен на рисунке 15.

Расчет контактных сил для отдельных частиц описывается формулой (12). Эта формула позволяет рассчитать силу взаимодействия между частицами, которая после приложения к объекту задаёт ему реалистичное поведение на сцене.

Для управления процессом создания и размещения частиц используются функции, представленные в листинге А5-А6 приложения А. Они будут определять количество, размер и расположение частиц в

соответствии с параметрами деформации. Это позволит точно настроить внешний вид и свойства ландшафта в зависимости от конкретных требований проекта.

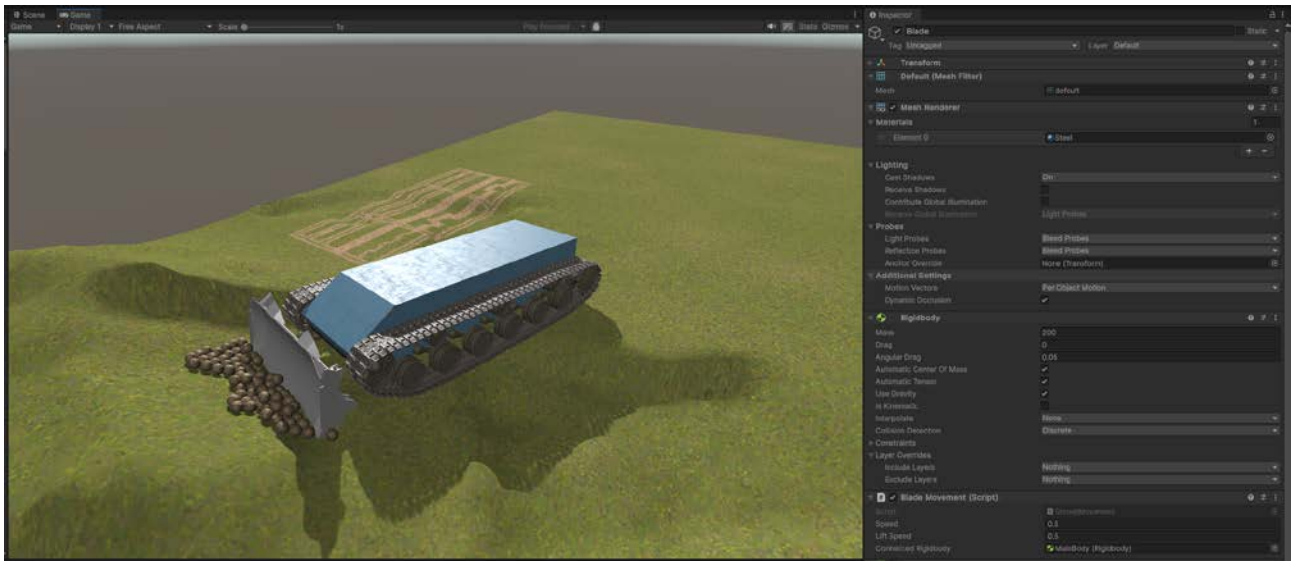

Рисунок 15 – Создание частиц почвы

Если частица почвы, которая столкнулась с почвенной сеткой, достигает состояния покоя, то есть её угловая и линейная скорости становятся меньше определённого порога, её движение больше не влияет на динамику почвы. В этом случае эта частица исключается из симуляции и заменяется её статическим объёмом в почвенной сетке, увеличивая высоту соответствующего слоя почвы.

4. ПРОВЕДЕНИЕ ТЕСТИРОВАНИЯ И СБОР РЕЗУЛЬТАТОВ

Для проверки работоспособности проекта, созданного в предыдущей части, необходимо провести тестирование. Тестирование можно разделить на две основные категории: функциональное и нефункциональное.

Функциональное тестирование отвечает за проверку основных функций программы, таких как взаимодействие отвала с Terrain, создание и управление трёхмерной местностью (ландшафтом), а также добавление и удаление сферических объектов.

Нефункциональное тестирование сосредоточено на таких аспектах, как безопасность, производительность и удобство использования. Созданный проект соответствует всем заявленным нефункциональным требованиям, таким как операционная система или среда разработки.

Так как это первая версия модели, её целью является тестирование и отладка различных подходов для создания максимально реалистичной модели почвы. Поэтому для тестирования модели будем использовать метод функционального тестирования.

Этот тип тестирования направлен на проверку функций приложения путём выполнения тестовых случаев и сопоставления ожидаемого результата с фактическим. Это позволит выявить возможные ошибки и недочёты в работе программы и внести необходимые исправления. Результаты тестирования представлены в таблице 1.

Таблица 1 – Тестирование приложения на соответствие функциональнымтребованиям

Назначение	**Ожидаемый результат**	**Полученный результат**	**Итог**
Преобразование почвенной сетки в частицы сферической формы при сдвиговой деформации	При деформации почвы, в месте изменения высоты вершины сетки, должен быть создан гранулированный материал, замещающий разрушенную почву.	Модель успешно преобразует почвенную сетку в частицы.	Тест пройден

Продолжение таблицы 1

Расчёт силы взаимодействия между частицами	Расчётная сила взаимодействия после приложения к объекту должна задавать ему реалистичное поведение на сцене.	Объект двигается и реагирует на внешние воздействия в соответствии с физическими законами и параметрами, заданными в модели.	Тест пройден
Объединение частиц с почвенной сеткой после достижения равновесия	После достижения равновесия частицы должны объединиться с почвенной сеткой, образуя единую структуру.	Частицы объединились с почвенной сеткой, образуя единую структуру.	Тест пройден

Эти результаты подтверждают успешное функциональное тестирование модели и её способность точно рассчитывать и имитировать различные аспекты взаимодействия отвала с почвой. Модель может быть использована для создания реалистичных ландшафтов, моделирования процессов деформации почвы и других задач, связанных с взаимодействием объектов с поверхностью.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была разработана и реализована математическая модель почвы на платформе Unity.

В процессе исследования был проведён анализ литературы, в результате которого для моделирования поведения сыпучей породы была выбрана модель МДЭ, а для моделирования сплошной почвенной сетки – МКЭ. На основе анализа были определены функциональные и нефункциональные требования к модели.

Для расчёта контактных усилий объектов была выбрана математическая модель Герца. С её применением была спроектирована модель, приведённая к виду стандартного неявного интегрирования.

Функциональное тестирование показало успешное выполнение всех поставленных требований. Результатом моделирования стала математическая модель почвы, реализованная на платформе Unity, которая позволяет выполнить визуальное представление процесса разбиения почвы, максимально приближенное к реальности.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Матвеев, А.Д. Метод многосеточных конечных элементов / А.Д. Матвеев // Вестник КрасГАУ. – 2018. – №2 (137). – С. 90–103.

2. Чепеленкова, В. Д. Применение метода дискретных элементов для оценки прочностных свойств упругих сред / В.Д. Чепеленкова, В. В. Лисица // Интерэкспо Гео-Сибирь. - 2022. - №2.- 210 с.

3. Харкевич, Л.В. Метод конечных разностей / Л.В. Харкевич // Вестник магистратуры. – 2017. – №7 (70). – С. 5–8.

4. Сибиряков, Е.Б. Использование метода граничных элементов для моделирования отражения от шероховатых границ // Интерэкспо Гео-Сибирь. – 2016. – №1. – 6 с.

5. Ковеня, В. М. Применение алгоритмов расщепления в методе конечных объемов / В.М. Ковеня // ЖВТ. – 2001. – №2. – С. 84–103.

6. Stransky, J. Open source fem-dem coupling / J. Stransky, M. Jirasek // 18th International Conference engineering mechanics. – 2012. – P. 1237–1251.

7. Hazay, M. Introduction to the Combined Finite-Discrete Element Method / M. Hazay // Budapest University of Technology and Economics, Hungary. – 2004. – 20 p.

8. Giannis, K. Stress based multi-contact model for discrete-element simulations / K. Giannis, C. Schilde, J. H. Finke, A. Kwade, M. A. Celigueta, K. Taghizadeh, S. Luding // Springer Nature. – 2021 – 14 p.

9. Holz, Daniel Soil Deformation Models for Real-Time Simulation: A Hybrid Approach / Daniel Holz, Thomas Beer, Torsten Kuhlen // Workshop on Virtual Reality Interaction and Physical Simulation. – 2021 – 11 p.

10. Adams, Bart Adaptively Sampled Particle Fluids / Bart Adams, Mark Pauly, Richard Keiser, Leonidas J. Guibas // Workshop on Virtual Reality Interaction and Physical Simulation. – 2006 – 7 p.

11. Servin, M. Examining the smooth and nonsmooth discrete element approaches to granular matter / Martin Servin, Claude Lacoursière, Da Wang,

Kenneth Bodin // International Journal for Numerical Methods in Engineering. – 2014 – 26 p.

12. Zhang, J. Simulation Analysis and Experiments for Blade-Soil-Straw Interaction under Deep Ploughing Based on the Discrete Element Method / Jin Zhang, Min Xia, Wei Chen, Dong Yuan, Chongyou Wu, Jiping Zhu // Nanjing Institute of Agricultural Mechanization, Ministry of Agriculture and Rural Affair/ – 2023 – 20 p.

13. Zarate, F. A simple FEM-DEM technique for fracture prediction in materials and structures / F. Zarate, E.Onate // Computational Particle Mechanics. – 2002. – 17 p.

14. Mohajerani, S. “Touch-aware” contact model for peridynamics modeling of granular systems / Soheil Mohajerani, Gang Wang // International Journal for Numerical Methods in Engineering. – 2022. – 29 p.

15. Serban, R. REAL-TIME SIMULATION OF GROUND VEHICLES ON DEFORMABLE TERRAIN / Radu Serban, Jay Taves, Zhenhao Zhou // International Design Engineering Technical Conferences. – 2022. – 10 p.

16. AGX-Dynamics [Электронный ресурс]: – URL: https://www.algoryx.se/agx-dynamics/ (дата обращения: 01.03.2024)

17. Johnson, K. Contact Mechanics / K. L. Johnson // Cambridge University Press. – 1987. – 232 p.

18. Боровин, Г.К. Обобщенная модель удара Герца - Ханта – Кроссли / Боровин Г.К., Лапшин В.В. // Вестник МГТУ им. Н. Э. Баумана. Сер. Естественные науки. – 2018. – №6 (81). – P. 18–30.

19. Mindlin, R. D. Elastic spheres in contact under varying oblique forces / R. D. Mindlin, H. Deresiewicz // Journal of Applied Mathematics. – 1953. – P. 327–344.

20. Bekker, MG Introduction to Terrain-Vehicle Systems / Bekker MG // University of Michigan Press. – 1969. – 864 p.

21. Janosi, Z. The analytical determination of drawbar pull as a function of slip for tracked vehicles in deformable soils / Z. Janosi, B. Hanamoto //

Proceedings of the 1st International Conference on Mechanics of Soil–vehicle Systems. – 1961. – 20 p.

22. Unity [Электронный ресурс]: – URL: https://unity.com/ (дата обращения: 01.03.2024)

23. Unreal Engine 5 [Электронный ресурс]: – URL: https://www.unrealengine.com/ (дата обращения: 01.03.2024)

24. Project Chrono [Электронный ресурс]: – URL: https://projectchrono.org/ (дата обращения: 01.03.2024)

25. Renzo, D. An improved integral non-linear model for the contact of particles in distinct element simulations / Di Renzo, A., Di Maio, F.P. // Chemical Engineering Science. – 2005. – P. 1303 – 1312.

26. Bornemann, F. A. Homogenization of Hamiltonian systems with a strong constraining potential / F.Bornemann, C. Schiitte // Courant Institute of Mathematical Sciences. — Berlin, Germany. – 1996. – P. 57–77.

27. Hemant, M. K. Range Searching using Kd Tree / Hemant M. Kakde // Springer-Verlag, USA. – 2005. – 12 p.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Листинг А1 - Функция расчета контактных сил

```
public void CalculateContactForce(ContactStruct contact)
{
    rn = contact.rhs[0] - normal_epsilon *
contact.lambda[0];

    rn -= Vector3.Dot(contact.G_n.normal[0],
contact.body_1.velocity);
    rn -= Vector3.Dot(contact.G_n.normal[1],
contact.body_1.angularVelocity);
    rn -= Vector3.Dot(contact.G_n.normal[2],
contact.body_2.velocity);
    rn -= Vector3.Dot(contact.G_n.normal[3],
contact.body_2.angularVelocity);

    newLambdaN = rn / contact.D[0] + contact.lambda[0];
    oldLambdaN = contact.lambda[0];

    contact.lambda[0] = (float)Math.Max(newLambdaN, 0);

    dln = contact.lambda[0] - oldLambdaN;

    contact.body_1.velocity += contact.GMInv.normal[0] *
dln;
    contact.body_1.angularVelocity +=
contact.GMInv.normal[1] * dln;
    contact.body_2.velocity += contact.GMInv.normal[2] *
dln;
    contact.body_2.angularVelocity +=
contact.GMInv.normal[3] * dln;

    // тангенциальный компонент
    ru = 0 - tan_epsilon * contact.lambda[1];
    ru -= Vector3.Dot(contact.G_n.tang_u[0],
contact.body_1.velocity);
    ru -= Vector3.Dot(contact.G_n.tang_u[1],
contact.body_1.angularVelocity);
    ru -= Vector3.Dot(contact.G_n.tang_u[2],
contact.body_2.velocity);
    ru -= Vector3.Dot(contact.G_n.tang_u[3],
contact.body_2.angularVelocity);

    rv = 0 - tan_epsilon * contact.lambda[2];
    rv -= Vector3.Dot(contact.G_n.tang_v[0],
contact.body_1.velocity);
         rv -= Vector3.Dot(contact.G_n.tang_v[1],
contact.body_1.angularVelocity);
```

```
    rv -= Vector3.Dot(contact.G_n.tang_v[2],
contact.body_2.velocity);
    rv -= Vector3.Dot(contact.G_n.tang_v[3],
contact.body_2.angularVelocity);

    oldLambdaU = contact.lambda[1];
    oldLambdaV = contact.lambda[2];

    newLambdaU = ru / contact.D[1] + contact.lambda[1];
    newLambdaV = rv / contact.D[2] + contact.lambda[2];

    contacts[i].lambda[1] = newLambdaU;
    contacts[i].lambda[2] = newLambdaV;

    double modG_n_Lambda_n =
contact.G_n.normal[0].magnitude * contact.lambda[0];

    if (contact.lambda[0] > 0)
    {
        double f = Math.Sqrt(Math.Pow(newLambdaU, 2) +
Math.Pow(newLambdaV, 2));

        if (f != 0 && f >= friction_coefficient *
modG_n_Lambda_n)
        {
            contact.lambda[1] =
                (float)(newLambdaU *
Math.Min(friction_coefficient * modG_n_Lambda_n / f, 1));
            contact.lambda[2] =
                (float)(newLambdaV *
Math.Min(friction_coefficient * modG_n_Lambda_n / f, 1));
        }

    }
    else
    {
        contact.lambda[1] = 0;
        contact.lambda[2] = 0;
    }

    dlu = contact.lambda[1] - oldLambdaU;
    dlv = contact.lambda[2] - oldLambdaV;

    contact.body_1.velocity += contact.GMInv.tang_u[0] *
dlu + contact.GMInv.tang_v[0] * dlv;
    contact.body_1.angularVelocity +=
contact.GMInv.tang_u[1] * dlu + contact.GMInv.tang_v[1] *
dlv;
    contact.body_2.velocity += contact.GMInv.tang_u[2] *
dlu + contact.GMInv.tang_v[2] * dlv;
```

```
    contact.body_2.angularVelocity +=
contact.GMInv.tang_u[3] * dlu + contact.GMInv.tang_v[3] *
dlv;
    ////////////////////////////////////////////////
    ///// трение качения
    ////////////////////////////////////////////////
    rrx = 0 - tan_epsilon * contact.lambdaR[0];
    rrx -= Vector3.Dot(contact.G_n.rotation_x[1],
contact.body_1.angularVelocity);
    rrx -= Vector3.Dot(contact.G_n.rotation_x[3],
contact.body_2.angularVelocity);

    rry = 0 - tan_epsilon * contact.lambdaR[1];
    rry -= Vector3.Dot(contact.G_n.rotation_y[1],
contact.body_1.angularVelocity);
    rry -= Vector3.Dot(contact.G_n.rotation_y[3],
contact.body_2.angularVelocity);

    rrz = 0 - tan_epsilon * contact.lambdaR[2];
    rrz -= Vector3.Dot(contact.G_n.rotation_z[1],
contact.body_1.angularVelocity);
    rrz -= Vector3.Dot(contact.G_n.rotation_z[3],
contact.body_2.angularVelocity);

    oldLambdaX = contact.lambdaR[0];
    oldLambdaY = contact.lambdaR[1];
    oldLambdaZ = contact.lambdaR[2];

    contact.lambdaR[0] += rrx / contact.Dr[0];
    contact.lambdaR[1] += rry / contact.Dr[1];
    contact.lambdaR[2] += rrz / contact.Dr[2];


    if (contact.lambda[0] > 0)
    {
        double f = Math.Sqrt(Math.Pow(contact.lambdaR[0], 2)
+ Math.Pow(contact.lambdaR[1], 2) +
Math.Pow(contact.lambdaR[2], 2));

        if (f != 0 && f >= resistance_coefficient * realR *
modG_n_Lambda_n)
        {
            contact.lambdaR[0] =
                (float)(contact.lambdaR[0] *
Math.Min(resistance_coefficient * realR * modG_n_Lambda_n /
f, 1));

            contact.lambdaR[1] =
```

```
                (float)(contact.lambdaR[1] *
Math.Min(resistance_coefficient * realR * modG_n_Lambda_n /
f, 1));

            contact.lambdaR[2] = 0;
        }
    }
    else
    {
        contact.lambdaR[0] = 0;
        contact.lambdaR[1] = 0;
        contact.lambdaR[2] = 0;
    }


    //contacts[i].lambdaR[2] = Math.Max(newLambdaZ, 0);

    dlx = contact.lambdaR[0] - oldLambdaX;
    dly = contact.lambdaR[1] - oldLambdaY;
    dlz = contact.lambdaR[2] - oldLambdaZ;

    contact.body_1.angularVelocity +=
contact.GMInv.rotation_x[1] * dlx +

contact.GMInv.rotation_y[1] * dly +

contact.GMInv.rotation_y[1] * dlz;

    contact.body_2.angularVelocity +=
contact.GMInv.rotation_x[3] * dlx +

contact.GMInv.rotation_y[3] * dly +

contact.GMInv.rotation_y[3] * dlz;

}
```

Листинг А2 - Функция инициализации k-мерного дерева

```
public void Init()
{
    Vector3[] pointCloud = new Vector3[100];
    for (int i = 0; i < pointCloud.Length; i++)
        pointCloud[i] = Random.insideUnitSphere;
    tree = new KDTree(pointCloud, 12);
}
```

Листинг А3 - Функция поиска контактных пар и их записи в список

```
public void CalculateImpact(GameObject[] granules, float
granularRadius, int maxPointsPerLeaf)
```

```
{
    Vector3[] pointCloud = new Vector3[granules.Length];
    allCollisions.Clear();

    //обновляем позиции объектов на сцене
    for (int i = 0; i < granules.Length; i++)
    {
        pointCloud[i] = granules[i].transform.position;
    }
    //строим KDTree
    tree.Build(pointCloud, maxPointsPerLeaf);

    //информацию о коллизонных парах храним в словаре
    //важно чтобы пары не повторялись
    List<int> local_results = new List<int>();
    for (int i = 0; i < pointCloud.Length; i++)
    {
        Vector3 pointPosition = pointCloud[i];
        local_results.Clear();

        //очередь для поиска соседей
        query.Radius(tree, pointPosition, 2 *
granularRadius, local_results);

        //обнаружены коллизии
        if (local_results.Count > 0)
        {
            //считаем все актуальные коллизии
            List<int> stayCollisions = new List<int>();
            foreach (var local_idx in local_results)
            {
                if (allCollisions.ContainsKey(local_idx))
                {
                    //добавляем в список
                    if (!allCollisions[local_idx].Any(idx =>
 idx == i))
                    {
                        stayCollisions.Add(local_idx);
                    }
                }
                else
                {
                    stayCollisions.Add(local_idx);
                }
            }
            if (stayCollisions.Count > 0)
            {
                allCollisions[i] = stayCollisions;
            }
        }
```

```
    }
}
```

Листинг А4 - Функция поднимания и опускания отвала

```
void Update()
{
    float forwardInput = Input.GetAxis("Vertical");

    // Вычисляем вектор направления движения на основе
текущего угла поворота
    Vector3 movementDirection = Quaternion.Euler(0,
transform.eulerAngles.y, 0) * Vector3.forward;

    // Вычисляем направление движения
    Vector3 movement = movementDirection * forwardInput *
speed * Time.deltaTime;

    // Получаем ввод с клавиатуры
    float liftInput = 0f;
    if (Input.GetKey(KeyCode.Q))
    {
        if (fixedJointComponent != null)
        {
            Destroy(fixedJointComponent);
            Debug.Log("FixedJoint removed.");
        }
        liftInput += liftSpeed * Time.deltaTime;
        // Перемещаем объект по оси Y
        movement.y = liftInput;
        transform.Translate(movement, Space.World);
    }
    if (Input.GetKey(KeyCode.E))
    {
        liftInput -= liftSpeed * Time.deltaTime;

        if (fixedJointComponent != null)
        {
            Destroy(fixedJointComponent);
            Debug.Log("FixedJoint removed.");
        }
        liftInput += liftSpeed * Time.deltaTime;
        // Перемещаем объект по оси Y
        movement.y = liftInput;
        transform.Translate(movement, Space.World);
    }
    if (fixedJointComponent == null) {
        // Добавление компонента FixedJoint к игровому
объекту
        fixedJointComponent =
gameObject.AddComponent<FixedJoint>();
        fixedJointComponent.connectedBody =
```

```
connectedRigidbody;
        }
    }
```

Листинг А5 - Функция создания сфер

```
    void CreateSpheres() {

        float diameter = 2 * granularRadius;
        foreach (var ij in modifiedNodes) {
            Vector2 gridPosition2D = ij.Key;
            Vector2 point = Grid2World(gridPosition2D);
            VoxelRecord vr = voxels[gridPosition2D];
            float heightDifference = vr.level_initial -
vr.level;
            List<Vector3> sphereCenters =
    ClosePackingSpheres(granularRadius, cellSize, cellSize,
    heightDifference);
            float soilVolume = 2 * cellSize * heightDifference;
            float numNewParticals = soilVolume / granularVolume;
            int numNewParticlesFloor =
    (int)Math.Floor(numNewParticals);

            foreach (Vector3 center in sphereCenters) {
                Vector3 lowerLeftCorner = new Vector3(point.x –
    cellSize / 2, Get(gridPosition2D), point.y - cellSize / 2);
                Vector3 position = lowerLeftCorner + center;
                GameObject body =
    PoolManager.GetGameObjectFromPool(spherePrefab);
                body.transform.position = position;
                granularMaterial.Add(body);
            }
        }
    }
```

Листинг А6 - Функция размещения сфер

```
    List<Vector3> ClosePackingSpheres(float radius, float
length,
    float width, float height)
    {
        List<Vector3> sphereCenters = new List<Vector3>();
        float distance = (float)(2f * radius);
    // Расстояние между центрами сфер, чтобы они не пересекались

        float xSpacing = distance;
        float ySpacing = distance;
        float zSpacing = distance;
```

```
    float xOffset = radius;
    float yOffset = radius;
    float zOffset = radius;

    float x = xOffset;
    float y = yOffset;
    float z = zOffset;

    while (z <= width)
    {
        while (x <= length)
        {
            while (y <= height)
            {
                Vector3 point = new Vector3(x, y, z);
                sphereCenters.Add(point);
                y += ySpacing;
            }
            x += xSpacing;
            y = yOffset;

        }
        z += zSpacing;
        x = xOffset;
    }

    return sphereCenters;
}
```