МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**ВИЗУАЛИЗАЦИЯ ПОВЕДЕНИЯ ЧАСТИЦ С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ OpenGL**

Работу выполнил С.В. Мереуца

(подпись, дата) (инициалы, фамилия)

Факультет компьютерных технологий и прикладной математики 3 курс

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Научный руководитель

доц., канд.ф-м.н., Е.В. Кособуцкая

(подпись, дата) (инициалы, фамилия)

Нормоконтролер

доц., канд.ф-м.н., Е.В. Кособуцкая

(подпись, дата) (инициалы, фамилия)

Краснодар 2017

Содержание

[ВВЕДЕНИЕ 2](#_Toc501529881)

[1.Средства OpenGL 4](#_Toc501529882)

[1.1 Вершинные шейдеры 6](#_Toc501529883)

[1.2 Фрагментные шейдеры 6](#_Toc501529884)

[1.3 Геометрические шейдеры 6](#_Toc501529885)

[1.4 Шейдеры тесселяции 7](#_Toc501529886)

[1.5 Вычислительные шейдеры 7](#_Toc501529887)

[1.6 Клонирование объектов 8](#_Toc501529888)

[2. Моделирование физических процессов 8](#_Toc501529889)

[2.1 Явный метод Эйлера 10](#_Toc501529890)

[2.2 Метод Рунге-Кутты четвёртого порядка 11](#_Toc501529891)

[2.3 Параллельные вычислительные системы 12](#_Toc501529892)

[3. Разработка визуальной модели системы частиц 14](#_Toc501529893)

[3.1 Постановка задачи «Система частиц, демонстрирующих силу притяжения» 14](#_Toc501529894)

[3.2 Постановка задачи «Система частиц, образующих полотно» 14](#_Toc501529895)

[3.3 Использованные средства 15](#_Toc501529896)

[3.4 Работа программы 15](#_Toc501529897)

[3.5 Сравнение моделей 17](#_Toc501529898)

[3.5.1 Система частиц, демонстрирующих силу притяжения 17](#_Toc501529899)

[3.5.2 Система частиц, образующих полотно 20](#_Toc501529900)

[ЗАКЛЮЧЕНИЕ 22](#_Toc501529901)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc501529902)

# ВВЕДЕНИЕ

Визуализация – это процесс получения изображения по модели с помощью компьютерной программы.

В данном случае модель – это описание любых объектов или явлений на строго определённом языке или в виде структуры данных. Такое описание может содержать геометрические данные, положение точки наблюдателя, информацию об освещении, степени наличия какого-то вещества, напряжённость физического поля и прочее.

Синонимом визуализации является рендеринг. Часто в компьютерной графике под рендерингом понимают создание плоской картинки - цифрового растрового изображения - по разработанной трехмерной сцене.

Компьютерная модель - компьютерная программа, работающая на отдельном компьютере, реализующая представление объекта или системы в форме, отличной от реальной, но приближенной к алгоритмическому описанию, включающей и набор данных, характеризующих свойства системы и динамику их изменения со временем.

OpenGL (Open Graphics Library) – спецификация, определяющая, независимый от языка программирования, программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

OpenGL включает более 300 функций для рисования сложных трёхмерных сцен из простых примитивов. Используется при создании САПР, виртуальной реальности, компьютерных игр и визуализации в научных исследованиях, что и будет продемонстрировано в курсовой работе.

Целью курсовой работы является изучение возможностей средств OpenGL для создания компьютерных моделей.

# 1.Средства OpenGL

Программируемое графическое аппаратное обеспечение существует почти столько же времени, сколько и обычное. Единственный способ гарантировать поддержу современных API – внести программируемость

C наступлением XXI века некоторые фундаментальные принципы разработки графической аппаратуры изменились. Разработчики требовали все новых и новых возможностей, чтобы создавать захватывающие эффекты. В результате аппаратура стала более программируемой, чем когда-либо ранее.

Одновременно с совершенствованием графической аппаратуры совершенствовались графические API. Первоначально разработчикам были доступны подобные ассемблеру языки для обработки графики, однако со временем появились удобные и надежные языки высокого уровня.

Сегодня уходят в прошлое старые графические API и вместе с ними фиксированная функциональность. GPU стали универсальными процессорами для параллельной обработки чисел с плавающей запятой и могут быть использованы для решения огромного числа задач, даже не имеющих прямого отношения к графике.

Графический конвейер - это некоторое программно-аппаратное средство, которое преобразует описанные в виртуальном мире объекты в матрицу ячеек видеопамяти растрового дисплея. Каждое состояние конвейера получает данные из предыдущего состояния, обрабатывает их и передает следующему. Стандартный графический конвейер обрабатывает множество вершин, геометрических примитивов и фрагментов конвейерным способом. В OpenGL состояния конвейера реализованы в виде шейдеров.

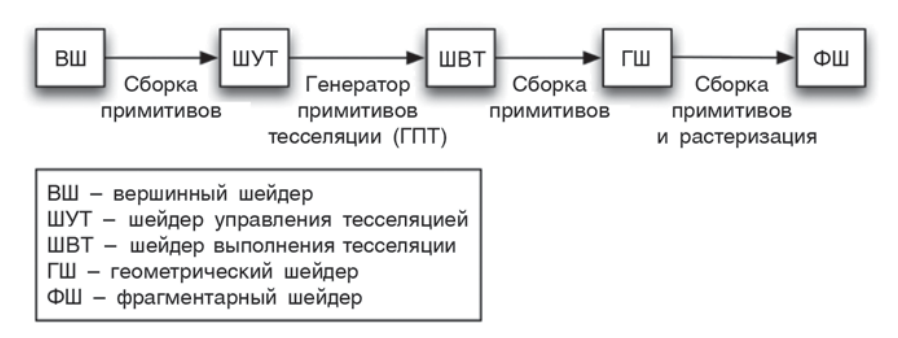


Рисунок 1 - Графический конвейер

В настоящее время язык GLSL является фундаментальной и неотъемлемой частью OpenGL API. Любая программа, написанная с привлечением OpenGL API, внутри использует одну или более программ на языке GLSL. Такие «мини-программы» часто называют шейдерными программами. Обычно шейдерная программа состоит из нескольких компонентов, называемых шейдерами. Каждый шейдер выполняется в рамках отдельного этапа в общем конвейере OpenGL. Каждый шейдер выполняется на GPU и, как можно заключить из названия, реализует алгоритм, так или иначе связанный с эффектами освещения и затенения в изображении. Однако шейдеры способны на большее, чем просто воспроизводить эффекты освещения и затенения. С их помощью можно также воспроизводить анимацию, выполнять тесселяцию и даже производить универсальные математические вычисления.

В зависимости от назначения, шейдеры делят на несколько типов:

* вершинные шейдеры;
* шейдеры тесселяции;
* геометрические шейдеры;
* фрагментные шейдеры;
* вычислительные шейдеры.

## 1.1 Вершинные шейдеры

Вершинный шейдер вызывается один раз для каждой вершины. Его главная задача – обработать данные, ассоциированные с вершиной, и передать их (и, возможно, другую информацию) на следующий этап конвейера. Обычно это координаты в пространстве, нормаль, компоненты цвета и текстурные координаты. Примерами обработки данных, связанных с вершинами могут служить: умножение вершин и нормалей на матрицу проекции и моделирования, установка цветов вершин, установка материалов освещения. Так же информация может передаваться дальше по конвейеру через выходные переменные. Это необходимо, например, для передачи данных из буфера, который недоступен из шейдеров, следующих за вершинным. Обязательной работой для вершинного шейдера является запись позиции вершины, в встроенную переменную gl\_Position.

## 1.2 Фрагментные шейдеры

Как и вершинный шейдер, фрагментный шейдер работает со входными данными, которые находятся на конвейере, и обрабатывает для передачи на следующие этапы. Фрагментный шейдер работает с растеризованными объектами, потому, в зависимости от примитива, который был растеризован, фрагментный шейдер может выполняться во много больше раз чем вершинный шейдер. Обработка может включать такие стадии, как получение данных из текстуры, просчет освещения, просчет смешивания. Обязательной работой для фрагментного шейдера является запись цвета фрагмента во встроенную переменную gl\_FragColor.

## 1.3 Геометрические шейдеры

Геометрический шейдер так же работает с данными, полученными из предыдущих стадий конвейера. Он выполняется для каждого примитива и имеет доступ к информации о всех вершинах, принадлежащих этому примитиву. Однако, в отличие от вершинного и фрагментного шейдера, геометрический шейдер не ограничивается модификацией поступающих данных. На этой стадии так же могут создаваться новые примитивы, которые так же поступят на конвейер.

## 1.4 Шейдеры тесселяции

Процессор тесселяции контролируется двумя шейдерами: шейдером управления тесселяцией и шейдером выполнения тесселяциии. На этой стадии происходит работа с примитивами, называемыми «заплатка». «Заплатки» - это наборы вершин и различных атрибутов. Шейдеры тесселяции, после получения такого набора, поразбивают его на набор точек, линий или треугольников и выводят данные на выполнение оставшимся этапам конвейера [1].

## 1.5 Вычислительные шейдеры

Впервые вычислительные шейдеры появились в версии OpenGL 4.3. Вычислительный шейдер представляет этап в шейдерной программе, в ходе которого можно выполнять произвольные вычисления. Он дает возможность использовать возможности GPU и присущий ему параллелизм для решения любых вычислительных задач, которые прежде можно было реализовать только на CPU. Вычислительные шейдеры с особым успехом можно использовать для задач, не связанных непосредственно с отображением графики, таких как симуляция физических процессов.

В настоящее время уже существуют библиотеки, обеспечивающие возможность выполнения универсальных вычислений на GPU, такие как OpenCL и CUDA, однако они никак не связаны с OpenGL. Вычислительные шейдеры, напротив, интегрированы непосредственно в OpenGL и, как следствие, лучше подходят для организации вычислений, имеющих отношение к отображению графики.

Вычислительный шейдер не является традиционным этапом выполнения шейдерной программы, как фрагментный или вершинный шейдер, то есть он не является частью графического конвейера. Он не вызывается в ответ на команды отображения. Фактически, когда вычислительный шейдер компонуется с вершинным, фрагментным или другими шейдерами, он остается инертным в отношении команд рисования [2].

## 1.6 Клонирование объектов

Отображение клонированием – это удобный и эффективный прием рисования нескольких копий одного и того же объекта. Моделирование поведения частиц, представленных сложными объектами, требует специальных средств. Вычисления производятся по отношению к объектам в целом вне зависимости от их формы. Визуализация объектов средствами OpenGL подразумевает задание атрибутов для каждой вершины. Отображение клонированием позволяет разделить объект на две составляющие, минимально связанные между собой. Форма, цвет и другие визуальные параметры частицы, и, например, позиция, скорость и ускорение не связаны между собой. Благодаря такому приему, можно работать с разными составляющими, не опасаясь появления ошибок.

# 2. Моделирование физических процессов

Для изучения объектов или процессов, протекающих в окружающем нас мире, широко используются методы математического моделирования. Математические модели являются отличным средством познания окружающего мира. Все естественные и общественные науки, использующие математический аппарат, по сути, занимаются математическим моделированием: заменяют объект исследования его математической моделью и затем изучают последнюю.

Физика – наука, в которой математическое моделирование является чрезвычайно важным методом исследования. Существует много причин, но из всех можно выделить две наиболее ярких: нелинейность многих физических процессов и необходимость исследования совместного движения многих тел.

Компьютерные модели стали обычным инструментом математического моделирования и применяются в физике, астрофизике, механике, химии, биологии, экономике, социологии, метеорологии, других науках и прикладных задачах в различных областях радиоэлектроники, машиностроения, автомобилестроения и прочие. Компьютерные модели используются для получения новых знаний об объекте или для приближенной оценки поведения систем, слишком сложных для аналитического исследования.

Компьютерное моделирование является одним из эффективных методов изучения сложных систем. Компьютерные модели проще и удобнее исследовать в силу их возможности проводить т. н. вычислительные эксперименты, в тех случаях, когда реальные эксперименты затруднены из-за финансовых или физических препятствий, или могут дать непредсказуемый результат. Логичность и формализованность компьютерных моделей позволяет определить основные факторы, определяющие свойства изучаемого объекта-оригинала (или целого класса объектов), в частности, исследовать отклик моделируемой физической системы на изменения её параметров и начальных условий.

Различают аналитическое и имитационное моделирование. При аналитическом моделировании изучаются математические (абстрактные) модели реального объекта в виде алгебраических, дифференциальных и других уравнений, а также предусматривающих осуществление однозначной вычислительной процедуры, приводящей к их точному решению. При имитационном моделировании исследуются математические модели в виде алгоритма, воспроизводящего функционирование исследуемой системы путём последовательного выполнения большого количества элементарных операций.

Так, например, некоторые процессы в радиотехнике, кинетика химических реакций, динамика биологических популяций, движение космических объектов, модели экономического развития исследуются с помощью уравнений, в которых кроме независимых переменных и неизвестных функций этих переменных, содержатся производные неизвестных функций. Таким образом появляется необходимость в решении дифференциальных уравнений. Для этого были созданы специальные численные методы.

## 2.1 Явный метод Эйлера

Метод Эйлера – простейший численный метод решения систем обыкновенных дифференциальных уравнений. Метод Эйлера является явным, одношаговым методом первого порядка точности, основанном на аппроксимации интегральной кривой кусочно-линейной функцией, так называемой ломаной Эйлера. Метод Эйлера являлся исторически первым методом численного решения задачи Коши. О. Коши использовал этот метод для доказательства существования решения задачи Коши.

Для дифференциального уравнения первого порядка

(1)

с начальным условием

(2)

явный метод Эйлера выглядит так

. (3)

где h – величина шага.

Ввиду невысокой точности и вычислительной неустойчивости для практического нахождения решений задачи Коши метод Эйлера применяется редко.

## 2.2 Метод Рунге-Кутты четвёртого порядка

Метод Рунге-Кутты - большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем. Первые методы данного класса были предложены около 1900 года немецкими математиками К. Рунге и М. В. Куттой.

Метод Рунге-Кутты четвертого порядка используют для расчета стандартных моделей достаточно часто, так как при небольшом объеме вычислений он обладает точностью метода.

Этот метод на каждом шаге требует четырех вычислений правой части дифференциального уравнения, но так как он четвертого порядка, то там, где метод Эйлера для достижения определенной точности требует 10000 вычислений правой части дифференциального уравнения, метод Рунге-Кутты четвертого порядка - только 40, что ведет к уменьшению вычислительной погрешности.

При пяти вычислениях правой части дифференциального уравнения не существует явного метода Рунге-Кутты пятого порядка, этот факт называется барьером Бутчера. Для построения метода пятого порядка нужно, как минимум, шесть вычислений. [3] Методы седьмого порядка должны иметь по меньшей мере девять стадий, а методы восьмого порядка – не менее 11 стадий. Для методов девятого и более высоких порядков (не имеющих, впрочем, большой практической значимости) неизвестно, сколько стадий необходимо для достижения соответствующего порядка точности. [4]

Для дифференциального уравнения первого порядка (1) с начальным условием (2) Метод Рунге-Кутты четвёртого порядка выглядит так

, (4)

где

,

,

,

.

## 2.3 Параллельные вычислительные системы

С появлением новых, более сложных задач, появилась необходимость в новых способах обработки данных. Таким образом новые центральные процессоры стали включать в себя несколько ядер, что повысило производительность вычислительных систем.

Параллельные вычислительные системы - это компьютерные, а также программные системы, реализующие тем или иным способом параллельную обработку данных на многих вычислительных узлах.

Идея распараллеливания вычислений основана на том, что большинство задач может быть разделено на набор меньших задач, которые могут быть решены одновременно, что определенно оказывается полезным в компьютерном моделировании, так как часто бывает необходимо моделировать различные стороны огромного количества объектов. Ярким примером может служить моделирование молекулярной динамики.

GPGPU (англ. General-purpose computing for graphics processing units, неспециализированные вычисления на графических процессорах) – техника использования графического процессора видеокарты, который обычно имеет дело с вычислениями только для компьютерной графики, чтобы выполнять расчёты в приложениях для общих вычислений, которые обычно проводит центральный процессор. Это стало возможным благодаря добавлению программируемых шейдерных блоков и более высокой арифметической точности растровых конвейеров, что позволяет разработчикам ПО использовать потоковые процессоры для неграфических данных.

Обычно параллельные вычисления требуют координации действий. Для облегчения распараллеливания на GPU существуют различные программные средства:

1) CUDA SDK. CUDA - это архитектура параллельных вычислений от NVIDIA, позволяющая существенно увеличить вычислительную производительность благодаря использованию GPU. На сегодняшний день продажи CUDA процессоров достигли миллионов, а разработчики программного обеспечения, ученые и исследователи широко используют CUDA в различных областях, включая обработку видео и изображений, вычислительную биологию и химию, моделирование динамики жидкостей, восстановление изображений, полученных путем компьютерной томографии, сейсмический анализ, трассировку лучей и многое другое;

2) OpenCL. OpenCL (от англ. Open Computing Language – открытый язык вычислений) - фреймворк для написания компьютерных программ, связанных с параллельными вычислениями на различных графических и центральных процессорах.

Цель OpenCL состоит в том, чтобы дополнить открытые отраслевые стандарты для трёхмерной компьютерной графики и звука OpenGL и OpenAL возможностями GPU для высокопроизводительных вычислений. OpenCL разрабатывается и поддерживается некоммерческим консорциумом Khronos Group, в который входят много крупных компаний, включая AMD, Apple, ARM, Intel, Nvidia, Sony Computer Entertainment и другие.

3) Вычислительные шейдеры OpenGL. Вычислительные шейдеры стали поддерживаться в версии OpenGL 4.3. Их можно использовать для осуществления ресурсозатратных вычислений. Удобство проявляется в том, что вычисления и визуализация осуществляются с минимальным количеством посредников, не говоря уже о совместимости данных.

# 3. Разработка визуальной модели системы частиц

## 3.1 Постановка задачи «Система частиц, демонстрирующих силу притяжения»

Необходимо создать модель, состоящую из аттрактора и системы из N частиц, подчиняющихся закону всемирного тяготения, а также визуализировать ее для наглядного отображения. Поведение частиц задается дифференциальным уравнением. Произвести сравнение поведения системы для двух вычислительных методов: метода Эйлера и метода Рунге-Кутты. Вычисления, связанные с этими методами, должны быть распараллелены и перенесены на GPU.

## 3.2 Постановка задачи «Система частиц, образующих полотно»

Необходимо создать модель, состоящую из системы частиц и имитирующую поведение полотна ткани, а также визуализировать ее для наглядного отображения. Вычисления выполняются путем решения дифференциальных уравнений. Произвести сравнение поведения системы для двух вычислительных методов: метода Эйлера и метода Рунге-Кутты. Вычисления, связанные с этими методами должны быть распараллелены, и перенесены на GPU.

## 3.3 Использованные средства

Вычисления и визуализация реализованы средствами OpenGL. До недавних OpenGL пор был удобным и гибким решением для создания двухмерной и трехмерной графики в приложениях, но с появлением вычислительных шейдеров, появилась возможность производить вычисления на GPU. Использование видеокарты подразумевает предварительную загрузку данных в ее память, что является относительно долгим процессом. Использование OpenGL как для вычислений, так и визуализации позволяет повысить производительность, потому что оба процесса используют одни и те же данные. Так же такая реализация позволяет не беспокоиться о совместимости данных.

OpenGL является распространенным средством, поэтому для большого количества языков программирования существуют свои реализации. В данной реализации используется язык С++. Этот язык программирования отлично подходит для программирования с использованием OpenGL. Этот язык довольно быстр, и способ представления данных в нем довольно прост и хорошо подходит, например, для занесения данных в буфер, где нужно передавать последовательность байтов, а не массив определенного типа данных.

## 3.4 Работа программы

В начале происходит инициализация окна, в котором будет визуализирована модель. Далее происходит компиляция всех шейдеров и их объединение в шейдерные программы. После этого инициализация массивов позиций и скорости частиц. Процессы инициализации и компилируемые шейдеры определяют модель, которая будет изображена.

В цикле отрисовки происходят вызовы шейдерных программ. Первая шейдерная программа выполняет вычисления новых значений позиции и скорости для каждой частицы. Вызов сопровождается заданием рабочих групп, количество которых зависит от количества частиц. Вторая шейдерная программа состоит из вычислительного шейдера и выполняет рендеринг модели, то есть располагает вершины согласно полученным значениям позиции, объединяет вершины в примитивы и добавляет освещение, если это необходимо.

Вычислительные шейдеры отличаются для каждой модели:

1)В модели демонстрирующей воздействие силы притяжения задается аттрактор. Это единственный объект, который оказывает какое-либо воздействие на частицы. Сила воздействия определяется формулой:

, (5)

где ri – вектор между i-м аттрактором и частицей;

Gi – мощность i-го аттрактора.

Для реализации имитации необходимо вычислить силу, воздействующую на каждую частицу, и изменить позицию этой частицы, интегрируя уравнения движения Ньютона.

2) В модели имитирующей поведение полотна, частицы взаимодействуют друг с другом. Распространенным способом имитации полотна ткани является связь точек «пружинами». Таким образом силы воздействия на частицы – это сила упругости от других восьми ближайших вершин. Сила воздействия определяется по формуле:

(6)

где K – жесткость пружины, R – длина пружины в состоянии покоя, r – вектор между соседней и данной частицами.

Здесь так же требуется интегрирование уравнения движения Ньютона для каждой частицы.

## 3.5 Сравнение моделей

### 3.5.1 Система частиц, демонстрирующих силу притяжения

При запуске программы частицы, представленные треугольниками, расположены в соответствии с начальной инициализацией позиций. Начальная скорость равна нулю. Затем начинает действовать сила аттрактора, и частицы начинают движение к нему.

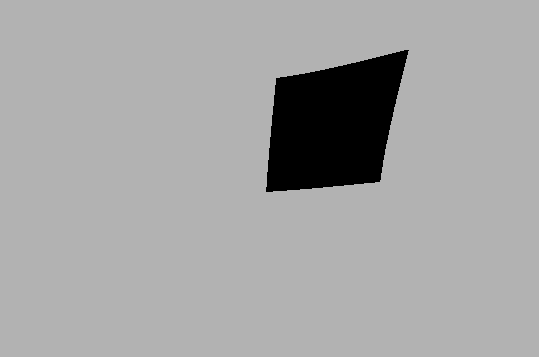


Рисунок 2 - Частицы до достижения аттрактора

Так как начальная скорость равна нулю, то частицы проходят напрямую через центр точки гравитации и потому «квадрат» частиц разрывает. Результаты зависят от метода.

1) Метод Эйлера

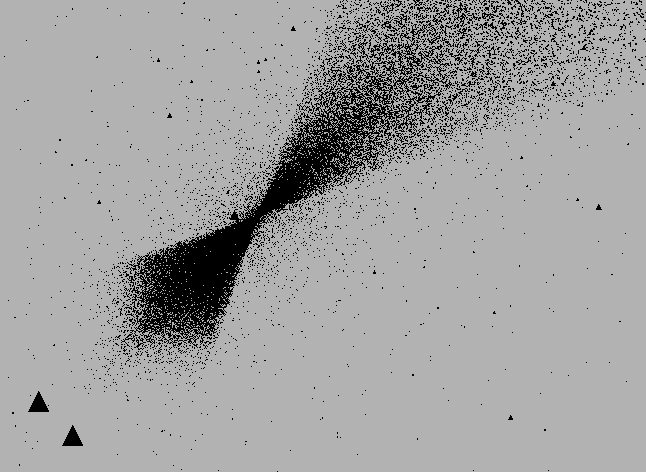


Рисунок 3 - Последствия прохождения через аттрактор

Как можно заметить, основные места расположений частиц – это два конуса направленные к аттрактору. Также большое количество частиц движется беспорядочно. Метод Эйлера довольно неточный.

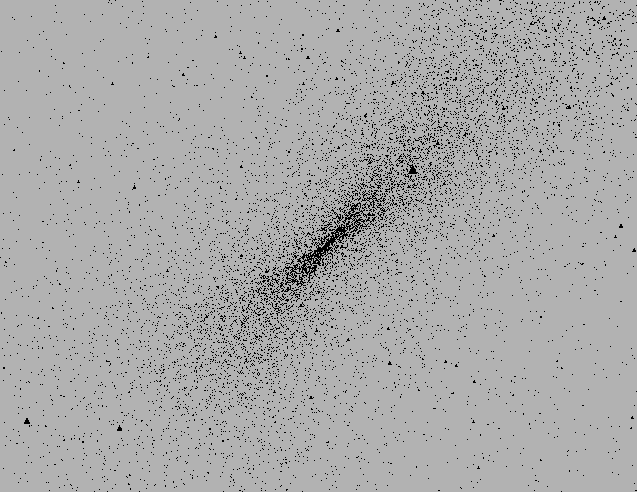


Рисунок 4 - Метод Эйлера спустя время

Через некоторое время конусы вовсе исчезают, а частицы движутся настолько беспорядочно насколько это возможно. Это связано с тем, что метод Эйлера хуже всего справляется там, где ускорение часто и сильно меняется, а прохождение через аттрактор, согласно формуле (5), заметно влияет на силу воздействия.

2) Метод Рунге-Кутты



Рисунок 5 - Метод Рунге-Кутты

Легко заметить, что в данном случае частицы движутся гораздо менее хаотично, чем в методе Эйлера. Частицы движутся так как должны двигаться и делают это стабильно. Даже через большой промежуток времени частицы не выбиваются из своих траекторий. Метод Ранге-Кутты более точен и рациональнее использовать в моделировании именно его. Метод Эйлера довольно прост, но, с точки зрения компьютера, разность в сложности вычислений пренебрежительно мала.

### 3.5.2 Система частиц, образующих полотно

Для наглядности, модели была добавлена сила тяжести и демпфирующая сила. Некоторое время оба метода справляются одинаково хорошо.



Рисунок 6 - Имитация полотна

1. Метод Эйлера

При использовании метода Эйлера модель некоторое время сохраняет свою форму, но через некоторое время полностью ее теряет.

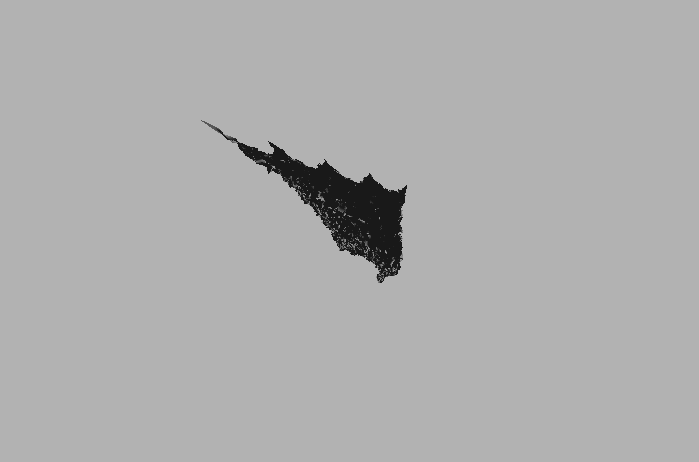


Рисунок 7 – Метод Эйлера

2) Метод Рунге-Кутты

При использовании метода Рунге-Кутты можно наблюдать отсутствие ошибок. После долгого времени работы модель сохраняет форму полотна.

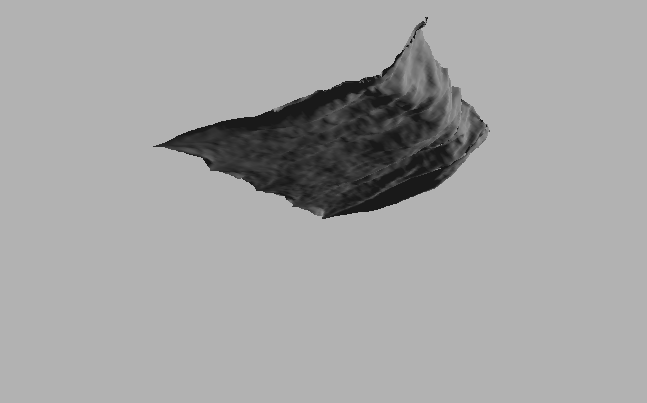


Рисунок 8 – Метод Рунге-Кутты

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были изучены общие сведения о способах моделирования и визуализации, а также созданы модели систем из физических тел с использованием средств OpenGL на языке программирования C++.

Проведены исследования эффективности двух численных методов: Эйлера и Рунге-Кутты, – для моделирования физических явлений. Было выявлено, что метод Рунге-Кутты четвертого порядка по всем параметрам превосходит метод Эйлера. Как вычислительная сложность, так и точность позволяют использовать именно метод Рунге-Кутты.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Вольф, Д. OpenGL 4. Язык шейдеров. Книга рецептов. – М.: ДМК Пресс, 2015.

2 Hearn D. Computer Graphics with Open GL. – Pearson Education, 2015.

3 Пименов, В. Г. Численные методы. Екатеринбург: Издательство Уральского университета, 2014

4 Butcher J. C. Numerical Methods for Ordinary Differential Equations. – New York: John Wiley & Sons, 2008.