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 системного программирования

Разработка эффективного метода реализации рендеринга

КУРСОВАЯ РАБОТА
по дисциплине «Программная инженерия»
ЮУрГУ – 09.03.04.2025.308-1582.КР

Нормоконтролер,
преподаватель кафедры СП
_____П.А. Манатин
«___»_____ 2025 г.

Научный руководитель:
преподаватель кафедры СП
_____П.А. Манатин

Автор работы:
студент группы КЭ-343
_____Г.О. Кузьмин

Работа защищена
с оценкой: _____
«___»_____ 2025 г.

Челябинск, 2025 г.

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 системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
15.02.2025 г.

ЗАДАНИЕ
на выполнение курсовой работы
по дисциплине «Программная инженерия»
студенту группы КЭ-343
Кузьмину Глебу Олеговичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы**

Разработка эффективного метода реализации рендеринга.

2. **Срок сдачи студентом законченной работы:** 31.05.2025 г.
3. **Исходные данные к работе**

3.1. Ермоленко, А. В., Осипов, К. С. Параллельное программирование в контактных задачах со свободной границей // Вестник Сыктывкарского университета. Серия 1. Математика. Механика. Информатика. — 2017. — № 23. — С. 85–91.

3.2. Berzal, F. Structured parallel programming by Michael McCool, James Reinders & Arch Robison // SIGSOFT Software Engineering Notes. — 2013. — Vol. 38, No. 2. — P. 35–39.

3.3. Документация OpenMP [Электронный ресурс]. — URL: https://homepages.math.uic.edu/jan/mcs572f16/mcs572notes/lec09.html (дата обращения: 18.06.2025).

4. **Перечень подлежащих разработке вопросов**

4.1. Анализ предметной области.

4.2. Проектирование и разработка приложения.

4.3. Проведение тестирования приложения.

5. Дата выдачи задания: 15.02.2025 г.

Научный руководитель,
преподаватель кафедры СП П.А. Манатин

Задание принял к исполнению Г.О. Кузьмин

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В данный момент потребность в вычислительной мощности со стороны программного обеспечения к ресурсам системы, на которой она выполняется, очень высока. Причин для этой проблемы много, но основной является нынешняя особенность архитектуры ЭВМ. Современные настольные компьютеры содержат многоядерные процессоры, что позволяет системе выделять процессам отдельные ядра (или потоки) для выполнения множества задач параллельно, а не по очереди. Однако в случае нехватки производительности отдельного ядра (потока), выполняемый процесс будет «тормозить», при том, что утилизация ресурса процессора не полная. Многие разработчики рассчитывают на то, что ресурсов одного логического ядра вполне хватит, однако это не всегда так.

Для решения этой конкретной проблемы используются методы параллельного программирования, позволяющие использовать больше ресурсов для выполнения задачи.

Актуальность

Актуальность работы заключается в создании приложения с использованием новейших методов оптимизации и параллелизации приложения. Результат работы может стать показательным для разработчиков.

Постановка задачи

Целью курсовой работы является Разработка эффективного метода реализации рендеринга. Для достижения поставленной цели необходимо решить следующие задачи:

1) Провести анализ предметной области и аналогичных решений;
2) Сформулировать функциональные и нефункциональные требования к программному продукту;
3) Реализовать программу для симуляции физической модели.
4) Оптимизировать вычисления доступными методами.
5) Выполнить тестирование.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 34 страницы, объем списка литературы – 13 источников.

В первой главе проводится анализ предметной области. Рассматриваются основы параллельного программирования, его преимущества и ключевые метрики оценки эффективности, такие как ускорение и эффективность распараллеливания. Выполняется сравнительный анализ аналогов, включая библиотеки OpenMP, OpenCL и системы частиц в Unity и Houdini FX.

Вторая глава посвящена проектированию программного продукта. Формулируются функциональные и нефункциональные требования, описываются варианты использования программы, представленные в виде диаграммы. Разрабатывается архитектура системы, включающая процессы обработки модели, визуализации и управления, с акцентом на модульность и оптимизацию.

В третьей главе описывается программная реализация. Приводятся используемые инструменты, структура данных для частиц и алгоритмы физической симуляции. Рассматриваются механизмы оптимизации, включая пространственную сетку и параллельные вычисления, а также реализация пользовательского взаимодействия.

Четвертая глава посвящена тестированию приложения. Описываются методики оценки производительности, включая замеры FPS, времени обработки физики и рендеринга, а также сравнение однопоточной и многопоточной версий программы.

В приложениях содержатся спецификация вариантов использования (Приложение А) и скриншот разработанного приложения (Приложение Б).

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Предметная область проекта

Целью данной работы является разработка программы, направленной на симуляцию физической модели и изучение эффективности применения методов параллельного программирования к ней.

Параллельное программирование – это подход к разработке программ, в котором задачи выполняются на нескольких процессорах или ядрах процессора. Вместо того, чтобы выполнять код последовательно, подход позволяет распределить вычислительную нагрузку между несколькими вычислительными ресурсами, такими как процессоры, для более эффективного выполнения. Параллельное программирование позволяет кратно уменьшить время, затраченное на выполнение поставленной задачи.

Для исследования эффективности применения подхода обычно используются два показателя: эффективность распараллеливания и ускорение распараллеливания. Первый показывает, насколько хорошо применен подход, второй – как количество выделенных ресурсов влияет на эффективность.

1.2. Сравнительный анализ аналогов

Работы по теме параллельного программирования в основном связаны с разделением основную задачу программы на несколько подзадач, которые выполняются параллельно друг другу, что в итоге приводит к более быстрому получению результатов.

Большинство проектов с открытым исходным кодом используют общедоступные библиотеки для реализации параллельности в программном продукте, например OpenMP или OpenMPI из-за их простой интеграции в проект (OpenMP поддерживается в компиляторе GCC «из коробки»). В более серьезных проектах используются такие библиотеки как OpenCL или thread для получения доступа к аппаратным ресурсам на низком уровне.

В отдельных случаях разработчик проекта может применить GPGPU вычисления, что подразумевает использование ресурсов графического чипа

и видеопамяти для решения задачи. Реализация такого подхода очень трудоемка, так как фактически в персональном настольном компьютере GPU является отдельным устройством и не имеет доступа к памяти программы, а программа не может напрямую взаимодействовать с ресурсами графического чипа.

В основном расчет физической модели используется в компьютерной графике для видео игр и кино, например для просчета объемного тумана или создания динамически изменяющихся объектов окружения. Существует несколько реализаций аналогичных проектов с использованием параллельного программирования, например встроенный в программный пакет Unity [12], предназначенный для разработки видео игр, модуль «particle system». Этот модуль реализован достаточно эффективно, чтобы работать в реальном времени и при этом не перегружать систему. Точных расчетов эффективности реализации «particle system» получить не представляется возможным, так как Unity автоматически определяет, сколько ресурсов выделять как для отдельного кадра, так и для уникальной конфигурации системы пользователя.

Второй пример применения представлен в профильном программном пакете «houdini fx», предназначенном для создания реалистичных объемных или двумерных компьютерных эффектов. В «houdini fx» используются OpenCL и OpenMP с точной настройкой процесса просчета, программный пакет предоставляет возможность использования нескольких процессоров, видеокарт и специализированных ускорителей.

Основным методом оценки успеха в распараллеливании программы являются две метрики: ускорение и эффективность ускорения. Ускорение вычисляется делением времени выполнения непараллельной программы на время выполнения параллельной программы. Под эффективностью распараллеливания понимается отношение получаемого ускорения к числу задействованных процессов.

В результате обзора литературы был выявлен набор инструментальных средств для проведения исследования эффективности распараллеливания задач расчета и отображения симуляции физической модели.

Однопоточный вариант проекта будет реализован на языке программирования C++, с использованием системы сборки Cmake и компилятором GCC.

Наиболее подходящим вариантом для реализации проекта с поддержкой параллельных вычислений является библиотека OpenMP для распараллеливания на CPU.

Вывод по первой главе

2. ПРОЕКТИРОВАНИЕ

2.1. Требования к программному продукту

Для реализации приложения были выведены следующие требования:

Функциональные:

1. Симуляция двумерной физической модели.
2. Поддержка многопоточной симуляции с использованием библиотеки OpenMP.
3. Режим работы без пользователя для сбора информации о модели.
4. Запись результатов выполнения теста производительности программы в файл CSV формата.
5. Поддержка взаимодействия пользователя и симуляции.

Нефункциональные:

- Отображение текущей производительности.

Эти требования описывают минимальный функционал программы.

2.2. Варианты использования программы

Исходя из поставленных требований была создана диаграмма использования приложения, она представлена на рисунке 1.

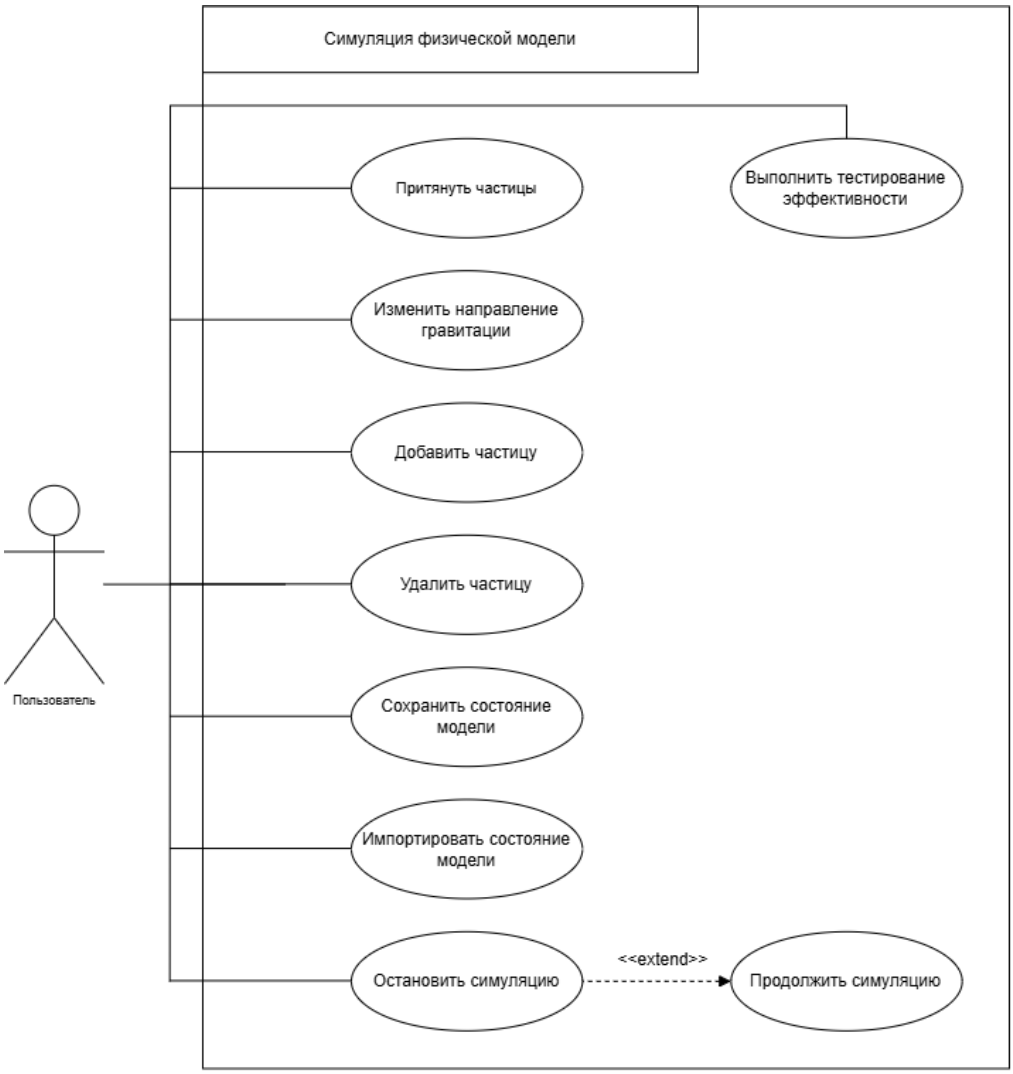

Рисунок 1 – Диаграмма вариантов использования

Пользователь является основным актером, который будет пользоваться системой, однако во время тестирования производительности пользователь не может взаимодействовать с ней.

Спецификация вариантов использования находится в приложении А.

Краткое описание всех вариантов использования приложения:

Притянуть частицы: Пользователь создает локальную временную точку притяжения.

Изменить направление гравитации: Пользователь меняет направление гравитации.

Добавить частицу: Пользователь добавляет новую частицу на место курсора.

Удалить частицу: Пользователь убирает существующую частицу на месте курсора из симуляции.

Сохранить состояние модели: Пользователь сохраняет состояние модели в файл на устройстве.

Импортировать состояние модели: Пользователь импортирует сохраненную модель с устройства.

Остановить симуляцию: Пользователь приостанавливает просчет физики в системе.

Продолжить симуляцию: Пользователь возобновляет просчет физики в системе.

Выполнить тестирование эффективности: Система выполняет симуляцию по заранее определенным параметрам.

Сохранить результаты в файл: Система сохраняет результаты тестирования в файл.

2.3. Архитектура программы

Система симуляции и рендеринга физической модели представляет собой программное решение, предназначенное для моделирования поведения частиц в реальном времени с учетом внешних взаимодействий. Архтектура системы разработана в соответсвии с требованием к высокой скорости выполнения.

Основу системы состовляют три отдельных процесса:

1. **Процес обработки модели**, отвечающий за расчет взаимодействия между частицами, влияние гравитации и внешние силы. Для оптимизации вычислений в данном процессе реализованы два механизма: пространственная сетка (требуется для пропуска расчета коллизий частиц, находящихся на большом удалении друг

от друга) и параллельные вычисления физики с использованием библиотеки «OpenMP».

2. **Процесс визуализации**, предназначенный для отрисовки частиц и вывода служебной информации (такой как текущее количество частиц и время выполнения одного цикла). Визуализация модели реализована с учетом минимизации накладных расходов, что важно в рамках проекта.
3. **Процесс управления и взаимодействия**, включающий обработку пользовательского ввода (мышь или клавиатура).

Все три процесса выполняются обособленно, что позволяет выделить время выполнения на самый важный – процесс обработки модели, в то время как процессы визуализации и взаимодействия можно выполнять реже, тем самым выделить больше вычислительного времени на более важный процесс просчета физики.

Для хранения и управления состоянием системы реализованы механизмы сериализации данных, позволяющие сохранять и загружать текущее состояние физической модели.

Архитектура системы демонстрирует высокую степень модульности, что упрощает дальнейшее расширение функциональности (например интеграция дополнительных свойств или правил модели, таких как трение).

Важным аспектом архитектуры является ее адаптивность к аппаратным ресурсам. Использование кроссплатформенной библиотеки «OpenMP» позволяет генерировать исполняемый файл для всех популярных в текущее время операционных систем и архитектур.

Диаграмма классов проекта представлена на рисунке 2.

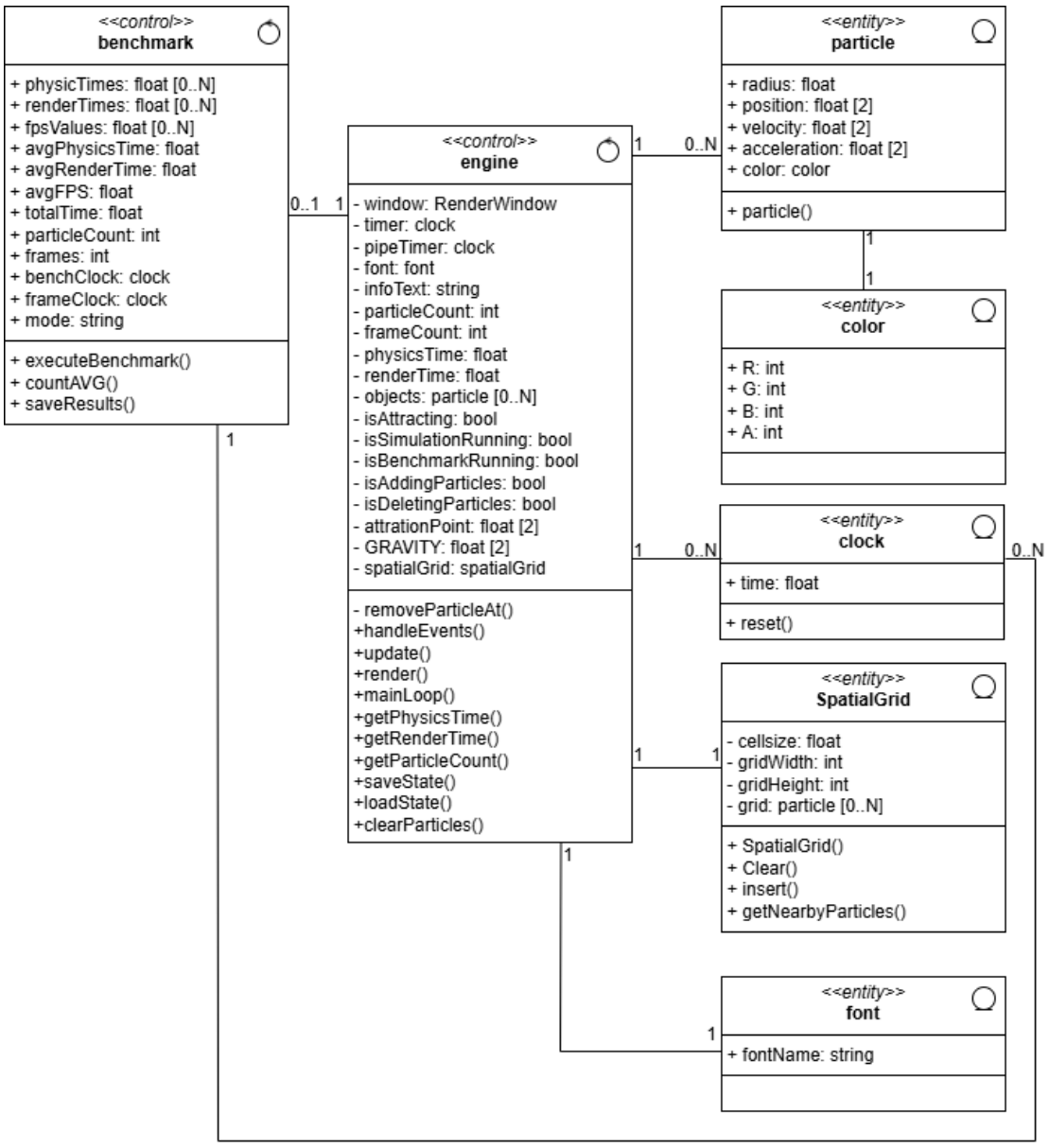

Рисунок 2 – Диаграмма классов

Структурная организация системы симуляции частиц основана на принципах модульности и разделения ответственности между компонентами. Архитектура системы включает несколько ключевых элементов, каждый из которых выполняет строго определенный набор функций и взаимодействует с другими компонентами через четко заданные интерфейсы.

Центральным элементом системы выступает ядро физического моделирования, реализующее основные алгоритмы расчета физических взаимодействий. Данный компонент отвечает за вычисление траекторий движения частиц, обработку гравитационных воздействий и обнаружение коллизий

между объектами. Для обеспечения высокой производительности в ядре реализованы два взаимодополняющих механизма оптимизации: пространственное разбиение с использованием пространственной сетки и параллельные вычисления на основе библиотеки «OpenMP». Первый подход позволяет существенно снизить вычислительную сложность алгоритмов обнаружения столкновений, второй - эффективно задействовать ресурсы многоядерных процессоров. Оба механизма работают согласованно, обеспечивая стабильную производительность даже при большом количестве моделируемых частиц.

Подсистема визуализации, построенная на базе библиотеки SFML, обеспечивает графическое представление текущего состояния модели. Этот компонент не только отображает текущее состояние системы частиц, но и предоставляет пользователю служебную информацию о параметрах работы приложения, включая частоту кадров, количество частиц и время выполнения различных этапов расчета. В процессе визуализации учитывается работа с тысячами объектов.

Модуль управления состоянием выполняет функции координатора работы системы. В его обязанности входит создание и удаление частиц, сохранение и восстановление состояний системы, а также обработка пользовательского ввода.

Все компоненты системы разработаны с учетом требований к расширяемости и модифицируемости. Интерфейсы между модулями определены таким образом, чтобы минимизировать взаимозависимости и упростить процесс внесения изменений в отдельные части системы без необходимости переработки всей архитектуры.

Система использует оптимизированную бинарную модель хранения данных, где состояние частиц сохраняется в виде последовательных снимков (snapshots), каждый из которых содержит заголовок с метаданными (версия формата, временная метка, количество частиц) и массив структур фиксированного размера (40 байт на частицу), описывающих координаты, скорость,

ускорение, радиус и цвет частиц. Модель обеспечивает целостность данных через контрольные суммы и проверку согласованности, поддерживая два режима работы: оперативное хранение в памяти в формате, оптимизированном для расчетов, и долговременное хранение в сжатых бинарных файлах, демонстрируя линейную масштабируемость до 1 миллиона частиц.

Диаграмма типа «сущность-связь» описывает взаимодействие данными сущностей внутри системы симуляции частиц, она приведена на рисунке 3.

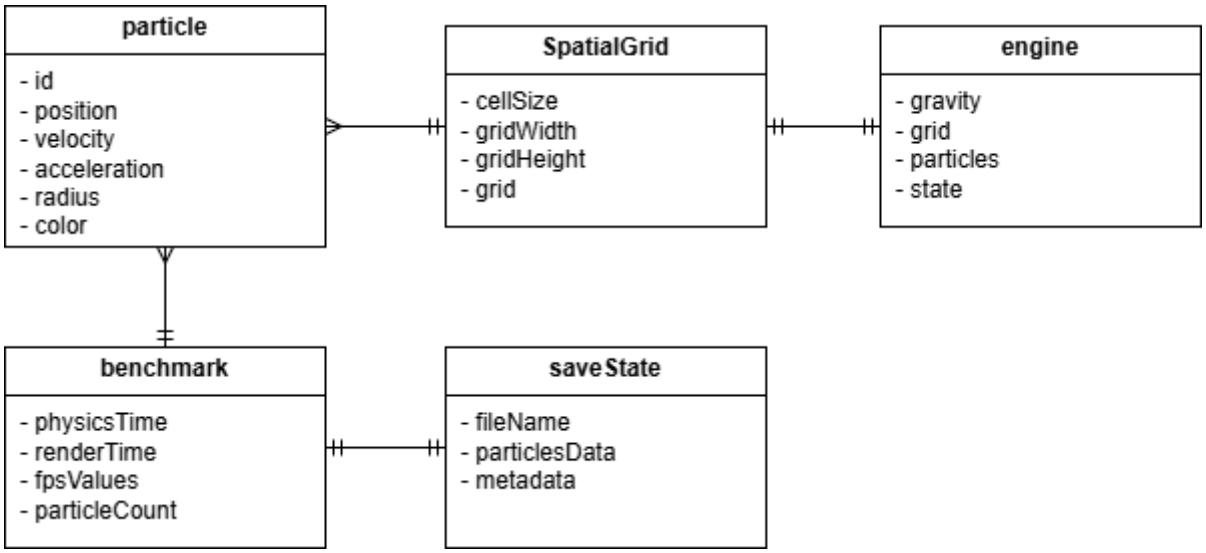

Рисунок 3 – Диаграмма «сущность-связь»

3. РЕАЛИЗАЦИЯ

3.1. Программная реализация

Программа была реализована с использованием ранее рассмотренных инструментов:

- GCC – Компилятор для компиляции исходного C++ кода в исполняемый файл.
- CMake – Система сборки проектов, удобна в использовании и интеграции.
- SFML – C++ библиотека с открытым исходным кодом, предназначенная для создания окна в операционной системе windows, в рамках проекта также используется для отрисовки объектов на экране.
- OpenMP – Встроенная в компилятор GCC библиотека, используется для распараллеливания задач.

Этот набор инструментов является наиболее подходящим для реализации проекта, так как язык C++ предоставляет непосредственный доступ к памяти приложения, это необходимо для последующей оптимизации.

Изначальная реализация программы не имеет поддержки многопоточной обработки физической модели, а также не использует SIMD инструкции, поддерживаемые современными процессорами.

Основой системы является структура «Particle», определенная в файле particle.hpp, которая описывает свойства частицы: позиция (sf::Vector2f) задает координаты, скорость определяет движение, ускорение изменяется под действием сил, радиус используется для столкновений и рендеринга, а цвет (sf::Color) обеспечивает визуальное различие. Частицы создаются динамически через функцию «createParticle» и хранятся в векторе «std::vector» в классе «Engine», с максимальным количеством 1000, заданным параметром «g_maximumParticles» в файле «parameters.hpp». Цвет частиц генерируется на основе номера кадра, что улучшает визуальную идентификацию.

Физическая симуляция, реализованная в файле «updateOpenMP.cpp», управляет движением и взаимодействием частиц. Функция «applyGravity» добавляет к ускорению каждой частицы вектор гравитации (GRAVITY), направление и величина которого могут изменяться пользователем с помощью клавиш стрелок. По умолчанию гравитация направлена вниз, обеспечивая естественное падение частиц, но пользователь может моделировать различные сценарии, такие как невесомость или боковые силы. Функция «checkCollisions» обрабатывает столкновения между частицами, используя класс «SpatialGrid» для оптимизации вычислений. «SpatialGrid» делит пространство на ячейки, ограничивая проверку столкновений соседними частицами, что снижает вычислительную сложность с $O(n^2)$ до $O(n)$, особенно важно при максимальном количестве частиц. При обнаружении столкновения вычисляется импульс на основе относительной скорости и нормали столкновения, а позиции частиц корректируются для устранения перекрытия.

Параллельная обработка с помощью OpenMP, реализованная через директиву «#pragma omp parallel for», ускоряет этот процесс. Функция «applyAttraction» активирует силу притяжения при нажатии средней кнопки мыши, притягивая частицы в заданном радиусе к курсору с силой, уменьшающейся с расстоянием, создавая эффект, подобный гравитационному притяжению. Функция «checkBorders» ограничивает движение частиц рамками окна, инвертируя скорость при достижении границы и уменьшая ее за счет коэффициента затухания, что обеспечивает реалистичный отскок.

Класс «Benchmark» измеряет производительность системы, записывая время обработки физики и рендеринга за 1000 кадров. Средние значения FPS, времени физики и рендеринга сохраняются в CSV-файл для последующего анализа. Класс позволяет сравнить производительность при использовании OpenMP (параллельная обработка) и без него, демонстрируя эффективность параллелизации.

Функция «render» в классе «Engine» отображает частицы как круги, используя их радиус и цвет. Экран очищается перед каждым кадром, после чего рисуются все частицы. В верхнем левом углу окна отображаются метрики производительности: текущий FPS, количество частиц, время обработки физики и рендеринга. Это позволяет пользователю отслеживать производительность системы в реальном времени.

Поддержка пользовательского ввода выполнена с использованием API windows, что делает приложение эксклюзивным для платформы для этой платформы. Скриншот приложения приведен на рисунке 4.

Класс «Engine» управляет визуализацией и пользовательским вводом. Функция «render» отображает частицы как круги, используя их радиус и цвет, очищая экран перед каждым кадром. В верхнем левом углу окна отображаются метрики производительности: FPS, количество частиц, время физики и рендеринга, что позволяет отслеживать производительность в реальном времени. Функция «handleEvents» обрабатывает события клавиатуры и мыши: пробел приостанавливает симуляцию, клавиши S и L сохраняют и загружают состояние частиц, стрелки изменяют гравитацию, клавиша B запускает бенчмарк через класс «Benchmark», левый клик добавляет частицы, правый клик удаляет, а средний клик активирует притяжение. Функции «saveState» и «loadState» обеспечивают сохранение и загрузку состояния частиц в бинарный файл.

Все настройки и переменные, влияющие на взаимодействие частиц с симуляцией, а также размер окна вынесены в отдельный файл «parameters.hpp», фрагмент которого приведен в листинге 1.

Листинг 1 – Фрагмент файла исходного кода «parameters.hpp»

```
const int g_windowWidth = 800;
const int g_windowHeight = 600;
const int g_fpsLimit = 1200;
const int g_maximumParticles = 1000;
const float g_particleRadius = 5.f;
const int g_collisionCheckCount = 15;
const float g_particleCollisionStrength = 1.55f;
const float g_damp = 0.75f;
```

```
const float g_attractionForce = 0.3f;
const float g_attractionRadius = 200.f;
const int g_benchmarkFrames = 1000;
const std::string g_benchResultFileName = "results.csv";
const std::string g_saveFileName = "particles_state.bin";
```

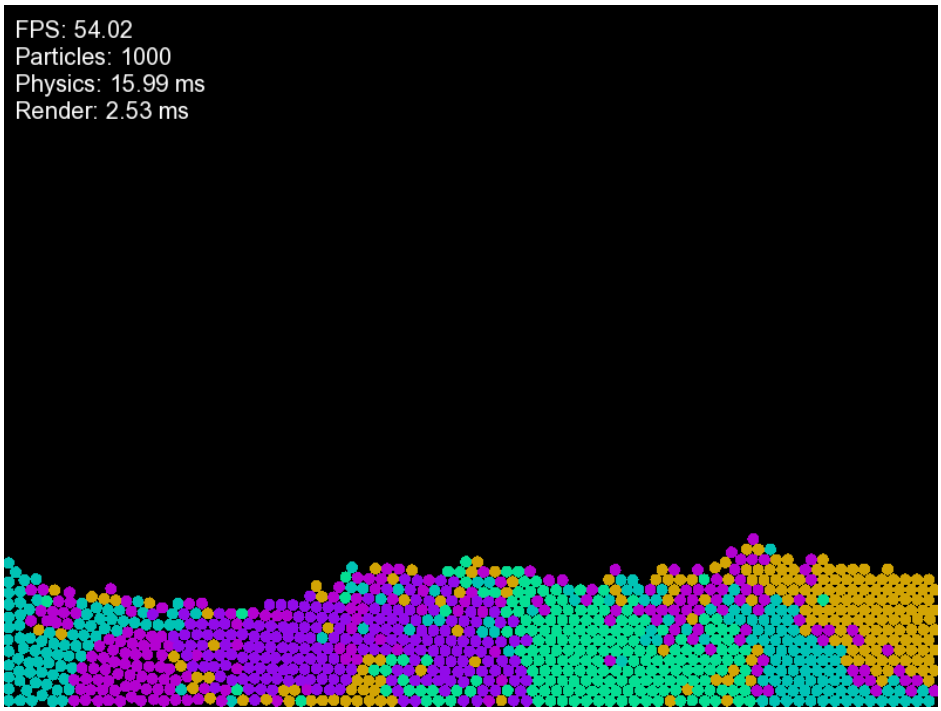

Рисунок 4 – скриншот программы

3.2. Оптимизация

Цель оптимизации программы заключается в повышении эффективности рендеринга физической модели, что позволяет увеличить количество вычислений, выполняемых за фиксированный промежуток времени, и улучшить общую производительность системы. Оптимизация проводилась с использованием нескольких подходов, направленных на снижение вычислительной сложности и эффективное использование аппаратных ресурсов.

Пространственное разбиение

Одним из ключевых методов оптимизации является использование пространственной сетки (spatial grid), реализованной в классе SpatialGrid. Этот подход позволяет значительно сократить количество проверок столкновений между частицами. Вместо проверки всех пар частиц, что имеет вы-

числительную сложность O(n^2), где n — количество частиц, пространственная сетка делит область симуляции на ячейки размером 1.5 радиуса частицы. Каждая частица помещается в соответствующую ячейку на основе ее координат, что позволяет ограничивать проверки столкновений только соседними ячейками. Это снижает сложность до O(n · k), где k — среднее количество частиц в соседних ячейках, что значительно меньше n при большом числе частиц.

Параллельные вычисления

Для дальнейшего повышения производительности применена библиотека OpenMP, которая обеспечивает многопоточную обработку задач. В частности, в файле updateOpenMP.cpp реализованы параллельные циклы (#pragma omp parallel for) для следующих операций: Применение гравитации к частицам; проверка столкновений; применение притяжения частиц к курсору.

Использование быстрого приближения квадратного корня

Для оптимизации вычислений расстояний между частицами применена функция быстрого приближения квадратного корня (fastSqrt) из файла «qSqrt.cpp». Эта функция, основанная на алгоритме Quake III для ускорения расчета освещения сцены, использует побитовые операции для быстрого вычисления обратного квадратного корня с последующей итерацией Ньютона для повышения точности. Это позволило сократить время, затрачиваемое на каждое вычисление расстояния между частицами.

Использование SIMD инструкций

SMID (single instruction multiple data) инструкции позволяют наиболее эффективно использовать ресурсы процессора, а точнее его вычислительных блоков, передавая сразу несколько переменных для выполнения над ними одной операции, это существенно влияет на количество вычислений за такт работы процессора.

Результаты оптимизации

Эти улучшения позволили программе эффективно работать в реальном времени даже при большом количестве частиц, обеспечивая высокую производительность рендеринга и симуляции физической модели.

Итоговый прирост производительности составляет около 4-х раз. В зависимости от конфигурации компьютера, на котором выполняется программа, результаты могут разниться, так как требуется быстрый доступ к памяти, пропускная способность памяти и чистая вычислительная мощность меньше влияют на скорость рендеринга.

4. ТЕСТИРОВАНИЕ

Тестирование системы проводилось для проверки функциональности, производительности и удобства пользовательского интерфейса, охватывая основные аспекты, включая создание частиц, физические взаимодействия, управление и производительность. Тестовые случаи подтвердили корректность работы системы: запуск симуляции приводит к созданию частиц до достижения лимита (1000); нажатие клавиш стрелок изменяет направление гравитации, и частицы начинают двигаться в новом направлении; при столкновении частиц они отскакивают с учетом физики без перекрытия; при достижении границ окна частицы отскакивают с уменьшением скорости за счет затухания; нажатие средней кнопки мыши притягивает частицы к курсору в заданном радиусе; клавиша Space приостанавливает и возобновляет симуляцию; клавиши S и L корректно сохраняют и загружают состояние частиц; левый и правый клики мыши добавляют и удаляют частицы в позиции курсора; клавиша B запускает бенчмарк, который выполняется 1000 кадров и сохраняет результаты в CSV-файл; метрики производительности (FPS, количество частиц, время физики и рендеринга) корректно отображаются в верхнем левом углу окна.

Тестирование приложения представлено в таблице 1.

Таблица 1 – Тестирование приложения.

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Создание частиц	1. Запустить программу. 2. Наблюдать автоматическое создание частиц.	Частицы создаются до тех пор, пока их не будет 1000.	Да
2	Изменение гравитации	1. Запустить программу. 2. Нажать одну из четырех стрелок на клавиатуре.	Частицы начинают двигаться в другом направлении гравитации.	Да
3	Столкновение частиц	1. Запустить программу. 2. Дать частицам столкнуться.	Частицы отскакивают друг от друга, не перекрывают друг друга.	Да

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
4	Столкновение с границами	1. Запустить программу. 2. Дать частицам столкнуться с границей.	Частицы отталкиваются от стенок окна.	Да
5	Сила притяжения	1. Запустить программу. 2. Дождаться появления нескольких частиц. 3. Зажать среднюю кнопку мыши. 4. Переместить курсор.	Частицы притягиваются к курсору.	Да
6	Пауза симуляции	1. Запустить программу. 2. Нажать клавишу «space» на клавиатуре.	Симуляция останавливается.	Да
7	Сохранение и загрузка состояния	1. Запустить программу. 2. Нажать кнопку «S» на клавиатуре. 3. Нажать кнопку «L» на клавиатуре.	Система записывает состояние модели в файл при нажатии «S» и загружает при нажатии «L».	Да
8	Добавление частицы	1. Запустить программу. 2. Нажать левую кнопку мыши.	Система создает новую частицу на месте курсора.	Да
9	Тестирование производительности	1. Запустить программу. 2. Нажать кнопку «B» на клавиатуре.	Система отчищает симуляцию, отрисовывает тысячу кадров и сохраняет результаты теста в файл.	Да
10	Отображение метрик производительности	1. Запустить программу. 2. Наблюдать левый верхний угол окна.	Отображаются корректные значения производительности.	Да

Все тестовые сценарии использования приложения прошли проверку, что говорит о верной программной реализации приложения.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана программа для симуляции двумерной физической модели с использованием методов параллельного программирования и оптимизации вычислений. Основной целью работы являлась разработка эффективного метода реализации рендеринга, что было успешно достигнуто путем решения поставленных задач.

1) Проведен анализ предметной области.
2) Спроектирована архитектура приложения.
3) Разработана программная реализация, соответствующая требованиям.
4) Проведена оптимизация процессов приложения.
5) Выполнено тестирование.

Таким образом, выполненная курсовая работа демонстрирует успешное применение методов параллельного программирования и оптимизации для реализации эффективного рендеринга физической модели, а также создает основу для дальнейших исследований и разработок в области высокопроизводительных вычислений.

ЛИТЕРАТУРА

1. Ермоленко, А. В., Осипов, К. С. Параллельное программирование в контактных задачах со свободной границей // Вестник Сыктывкарского университета. Серия 1. Математика. Механика. Информатика. — 2017. — № 23. — С. 85–91.

2. Berzal, F. Structured parallel programming by Michael McCool, James Reinders & Arch Robison // SIGSOFT Software Engineering Notes. — 2013. — Vol. 38, No. 2. — P. 35–39.

3. Документация OpenMP [Электронный ресурс]. — URL: https://homepages.math.uic.edu/ jan/mcs572f16/mcs572notes/lec09.html (дата обращения: 18.06.2025).

4. OpenMP Architecture Review Board. OpenMP Application Program Interface, Version 5.0 [Электронный ресурс]. — URL: https://www.openmp.org/specifications/ (дата обращения: 18.06.2025).

5. SFML Documentation [Электронный ресурс]. — URL: https://www.sfml-dev.org/documentation/2.5.1/ (дата обращения: 18.06.2025).

6. CMake Documentation [Электронный ресурс]. — URL: https://cmake.org/documentation/ (дата обращения: 18.06.2025).

7. GCC, the GNU Compiler Collection [Электронный ресурс]. — URL: https://gcc.gnu.org/onlinedocs/ (дата обращения: 18.06.2025).

8. Chapman, B., Jost, G., Pas, R. Using OpenMP: Portable Shared Memory Parallel Programming. — MIT Press, 2007. — 353 p

9. McCool, M., Reinders, J., Robison, A. Structured Parallel Programming: Patterns for Efficient Computation. — Morgan Kaufmann, 2012. — 432 p.

10. Reeves, W. T. Particle Systems — A Technique for Modeling a Class of Fuzzy Objects // ACM Transactions on Graphics. — 1983. — Vol. 2, No. 2. — P. 91–108.

11. Khronos Group. OpenCL Specification [Электронный ресурс]. — URL: https://www.khronos.org/registry/OpenCL/ (дата обращения: 18.06.2025).

12. Unity Documentation: Particle System [Электронный ресурс]. — URL: https://docs.unity3d.com/Manual/PartSysReference.html (дата обращения: 18.06.2025).

13. SideFX Houdini Documentation [Электронный ресурс]. — URL: https://www.sidefx.com/docs/houdini/ (дата обращения: 18.06.2025).

ПРИЛОЖЕНИЯ

Приложение А. Спецификация вариантов использования

Спецификация вариантов использования (ВИ) приложения приведена в таблицах 1–4.

Таблица 1 – Спецификация ВИ «Выполнить тестирование эффективности»

UseCase: Выполнить тестирование эффективности
ID: 1
Аннотация: Пользователь дает команду системе на выполнение тестирования производительности.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Пользователь нажимает на кнопку «B». 2. Система сбрасывает текущее состояние модели. 3. Система выполняет тестирование производительности. 4. Система записывает результаты тестирования в файл на устройстве пользователя.
Постусловия: Началось тестирование эффективности.
Альтернативные потоки: Нет.

Таблица 2 – Спецификация ВИ «Притянуть частицы»

UseCase: Притянуть частицы
ID: 2
Аннотация: Пользователь создает локальную точку притяжения.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Пользователь перемещает курсор на место, куда хочет установить точку притяжения. 2. Пользователь зажимает среднюю кнопку мыши. 3. Система создает точку притяжения на месте курсора. 4. Пользователь отпускает среднюю кнопку мыши. 5. Система удаляет точку притяжения.
Постусловия: Точка притяжения была создана и удалена.
Альтернативные потоки: Нет.

Продолжение приложения А

Таблица 3 – Спецификация ВИ «Изменить направление гравитации»

UseCase: Изменить направление гравитации
ID: 3
Аннотация: Пользователь изменяет направление гравитации.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Пользователь нажимает стрелку на клавиатуре. 2. Система изменяет направление гравитации в соответствии с направлением нажатой стрелки.
Постусловия: Направление гравитации изменено.
Альтернативные потоки: Нет.

Таблица 4 – Спецификация ВИ «Добавить частицу»

UseCase: Добавить частицу
ID: 4
Аннотация: Пользователь добавляет новую частицу в систему.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Пользователь перемещает курсор желаемое место появления новой частицы. 2. Пользователь нажимает левую кнопку мыши. 3. Система создает новую частицу на месте курсора.
Постусловия: Создана новая частица.
Альтернативные потоки: Нет.

Таблица 5 – Спецификация ВИ «Удалить частицу»

UseCase: Удалить частицу
ID: 5
Аннотация: Пользователь удаляет частицу из системы.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Пользователь перемещает курсор на место существующей точки. 2. Пользователь нажимает правую кнопку мыши. 3. Система перемещает точку притяжения на новое место.
Постусловия: Точка притяжения перемещена.
Альтернативные потоки: Нет.

Таблица 6 – Спецификация ВИ «Сохранить состояние модели»

UseCase: Сохранить состояние модели
ID: 6
Аннотация: Пользователь дает команду системе о сохранении состояния модели.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Пользователь нажимает кнопку «S» на клавиатуре. 2. Система сохраняет состояние в файл.
Постусловия: Файл состояния системы сохранен на устройстве пользователя.
Альтернативные потоки: Нет.

Таблица 7 – Спецификация ВИ «Импортировать состояние модели»

UseCase: Импортировать состояние модели
ID: 7
Аннотация: Пользователь дает команду системе об импорте сохраненной модели с устройства.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Пользователь нажимает кнопку «L» на клавиатуре. 2. Система загружает состояние модели из файла.
Постусловия: Файл состояния системы успешно импортирован.
Альтернативные потоки: Нет.

Таблица 8 – Спецификация ВИ «Остановить симуляцию»

UseCase: Остановить симуляцию
ID: 8
Аннотация: Пользователь останавливает обновление модели.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Нет.
Основной поток: 1. Пользователь нажимает кнопку «space» на клавиатуре. 2. Система прекращает обновление физической модели.
Постусловия: Обновление модели приостановлено.
Альтернативные потоки: Нет.

Таблица 9 – Спецификация ВИ «Продолжить симуляцию»

UseCase: Продолжить симуляцию
ID: 9
Аннотация: Пользователь возобновляет обновление модели.
Главные актеры: Пользователь.
Второстепенные актеры: Нет.
Предусловия: Модель должна быть приостановлена.
Основной поток: 1. Пользователь нажимает кнопку «space» на клавиатуре. 2. Система возобновляет обновление физической модели.
Постусловия: Обновление модели возобновлено.
Альтернативные потоки: Нет.

Приложение Б. Скриншоты приложения

Скриншот разработанного приложения приведен на рисунке 5.

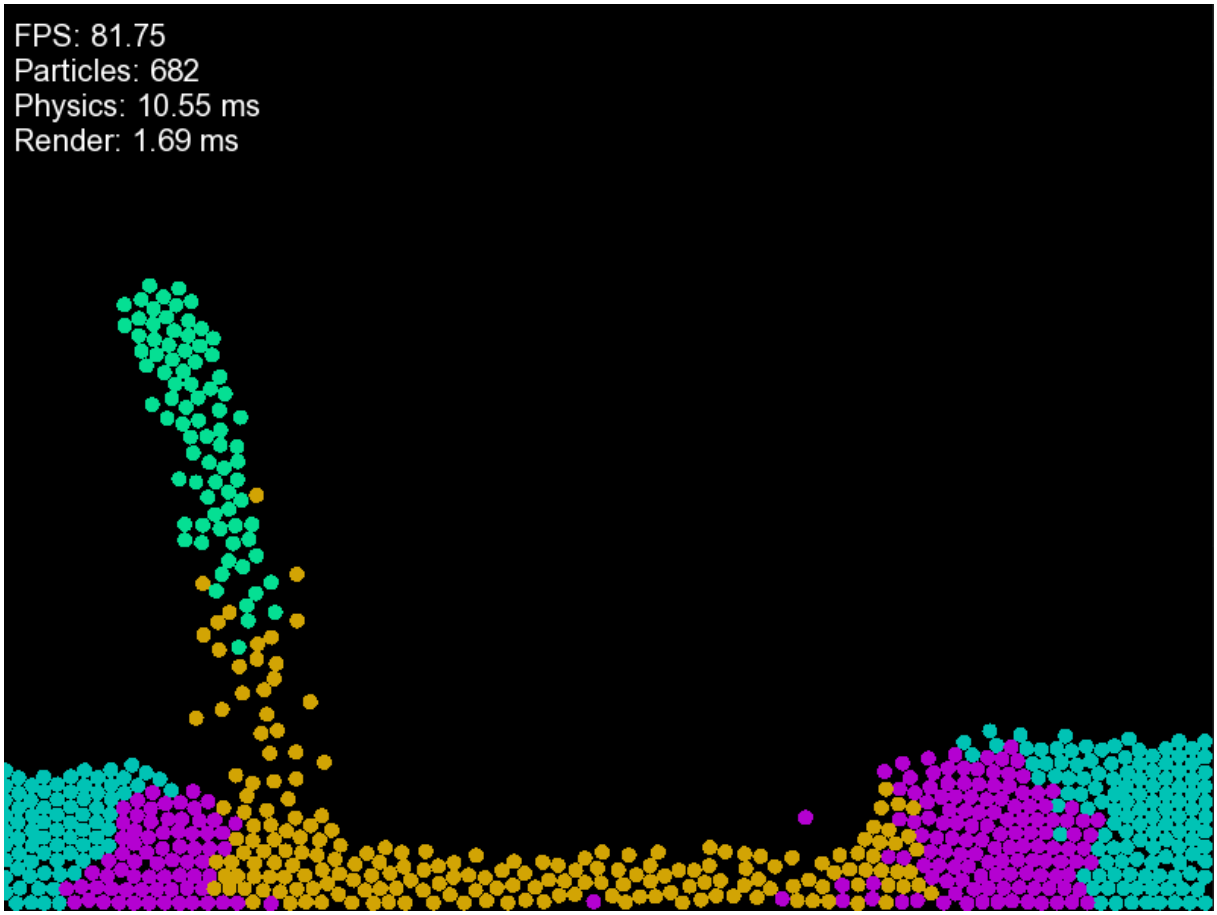

Рисунок 5 – скриншот программы