Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Моделирование снега»

Студент ИУ7-54Б _____ Мишина Е.В.
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ Корниенко В.В.
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2020г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
_____ И.В.Рудаков
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Компьютерная графика

Студент группы ИУ7-54б

Мишина Елена Викторовна
(Фамилия, имя, отчество)

Тема курсового проекта моделирование снега

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный
Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание Разработать программное обеспечение, моделирующие природные осадки в виде снега. Реализовать интерфейс, который позволит выбирать из предложенного набора интенсивность осадков. Объектом на сцене является дом. Программный продукт должен иметь возможность просмотра сцены с фиксированного положения наблюдателя.

Оформление курсового проекта:

Расчетно-пояснительная записка на _____ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_____
_____
_____
_____

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсового проекта _____ В.В. Корниенко
(Подпись, дата) (И.О.Фамилия)

Студент _____ Е.В. Мишина
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

Введение.

Проблема моделирования природных явлений и объектов является одной из наиболее актуальных в компьютерной графике. Такие задачи характерны для сравнительно нового и быстро развивающегося класса трехмерных графических приложений для персональных компьютеров, которые принято называть системами виртуальной реальности.

Для решения задачи моделирования осадков используется много методов: от стандартной системы частиц до использования базы данных с нарисованными художниками изображениями осадков с различных ракурсов, положений камеры и источников света. Последний из перечисленных подход очень сложен, так как необходимо наличие наличия базы данных, на создание и наполнение которой может потребоваться больше времени, чем на разработку программного комплекса, однако этот метод предоставляет более реалистичную картину.

Цель данной работы – реализовать построение трехмерной сцены и визуализацию снега.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1) описать структуру трехмерной сцены, включая объекты, из которых состоит сцена

2) выбор существующих алгоритмов трехмерной графики, которые позволят визуализировать трехмерную сцену;

3) реализация данных алгоритмов для создания трехмерной сцены;

4) разработать программное обеспечение, которое позволит отобразить трехмерную сцену и визуализировать снег.

1. Аналитическая часть

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов.

- Плоскость земли – ограничивающая плоскость, предполагается, что под плоскостью земли не расположено объектов. Изначально расположена внизу экрана (на максимальной координате y), параллельна oXZ. Размеры задаются шириной, длиной и цветом RGB.
- Дом – здание является кубом с основанием, параллельным плоскости земли, и боковыми ребрами, перпендикулярными плоскости земли. Положение дома задается программно. Также дом имеет пирамидальную крышу.
- Снег – частицы с заданной траекторией движения, цветом.

1.2 Анализ алгоритмов удаления невидимых линий и поверхностей

При выборе алгоритма удаления невидимых линий нужно будет учесть особенности поставленной задачи. Речь идет о наличии частичек осадков, которые будут перемещаться. Ввиду этого нюанса алгоритм, который будет использоваться, должен работать быстро, иначе осадки будут выглядеть как набор кадров, а не анимация.

Рассмотрим ключевые алгоритмы построения трехмерной сцены [1].

1.2.1 Алгоритм обратной трассировки лучей

Для создания реалистичного изображения, по правилам обратной трассировки, каждую частицу снега нужно будет рассматривать, как отдельный объект сцены, внутри которого могут возникнуть явления дисперсии, преломления и внутреннего отражения.

Положительной стороной данного алгоритма является возможность использования в параллельных вычислительных системах (т.к. расчет отдельной точки выполняется независимо от других точек).

Серьезным недостатком этого алгоритма будет являться большое количество необходимых вычислений для синтеза изображения моей сцены. Алгоритм не подойдет для генерации динамических сцен и моделирования диффузного отражения [4].

Вывод: так как в сцене не подчеркиваются явления преломления и отражения света, использование алгоритмов трассировки будет излишним. При заметном замедлении работы программы, качество изображения заметно не улучшится.

1.2.2 Алгоритм, использующий Z буфер

Несомненным плюсом данного алгоритма может являться его простота, которая не мешает решению задачи удаления поверхностей и визуализации их пересечения. В этом алгоритме не тратится время на сортировку элементов сцены, что дает преимущество в скорости работы.

Так как размер синтезируемого изображения сравнительно мал, затраты по памяти, при хранении информации о каждом пикселе, в данном алгоритме незначительны для современных компьютеров.

1.2.3 Алгоритм Варнока

Алгоритм Варнока основывается на рекурсивном разбиении экрана. В зависимости от расположения объектов это может стать, как положительной, так и отрицательной стороной алгоритма. Чем меньше пересечений объектов, тем быстрее алгоритм завершит свою работу.

1.2.5 Вывод

В виду того, что реалистичное моделирование осадков в виде снега требует прежде всего достаточной скорости визуализации сцены со всеми объектами и частицами целесообразен выбор алгоритма, использующего z-буфер, который проигрывает по памяти таким алгоритмам, как трассировка лучей, но выигрывает по быстродействию.

1.3 Анализ алгоритмов моделирования осадков

Реалистичное и эффективное представление сцен осадков в виде снега крайне непростая задача из-за огромного количества снежинок. Каждая из этих частиц обладает такими свойствами, как: размер, масса, скорость, ускорение.

1.3.1 Система частиц

Каждая частица представляет собой материальную точку с дополнительными характеристиками: скорость, ориентация в пространстве, время и эффекты «жизни» частицы. Обычно все частицы в системе меняют скорость или размер по общему закону. Различная интенсивность осадков достигается за счет изменения общего количества частиц.

Для ускорения программы предполагают, что частицы не отбрасывают тени и не поглощаются свет. Без этих допущений придется проделать много вычислений.

Не существует единого стандарта по реализации системы частиц. Частица (представляется точкой в трехмерном пространстве), по сравнению с

полигоном, более простой примитив. Система частиц – это самая простая форма представления поверхности. Благодаря этой простоте экономится вычислительное время, что позволяет обрабатывать больше примитивов и получать наиболее реалистичное изображение. Также система частиц позволяет моделировать «живые» объекты, которые могут изменяться с течением времени, чего нельзя сказать об объектах, состоящих из поверхностей.

1.4 Описание трехмерных преобразований сцены

Пространственные преобразования координат производятся с помощью матриц преобразования.

1.5 Выводы из аналитического раздела

В данном разделе были рассмотрены алгоритмы удаления невидимых линий и поверхностей, методы закрашивания поверхностей, алгоритмы построения теней и моделирования осадков. В качестве алгоритма удаления невидимых линий был выбран Zбуфер, для моделирования осадков была выбрана система частиц.

2. Конструкторская часть.

2.1 Требования к программе

Программа должна предоставлять следующие возможности:

- визуальное отображение сцены;
- выбор интенсивности снега;
- поворот исходной сцены.

2.2 Описание базовых алгоритмов.

2.2.1 Моделирование системой частиц.

2.2.1.1 Общая схема работы системы частиц.

С течением времени частицы генерируются системой, двигаются, изменяются внутри системы и умирают в ней. Для того чтобы посчитать на каждом кадре, нужно выполнить следующую последовательность действий:

1. новая частица генерируется системой;
2. каждая новая частица получает свой набор атрибутов;
3. частицы системы, время жизни которых превышает максимальное, угасают;
4. оставшиеся частицы перемещаются согласно некоторым законам изменения;
5. финальное изображение «живых» частиц рисуется в буфер кадра.

Система частиц на каждом шаге может быть запрограммирована выполнять любую последовательность инструкций, поэтому ее процедурность позволяет внедрить любую вычислительную модель, которая описывает поведение, динамику или внешний вид объекта. Далее описывается некая базовая модель для системы частиц и даны некоторые ее детали.

2.2.1.2 Генерация частиц.

Частицы генерируются некоторой системой частиц, и процесс генерации состоит из нескольких этапов:

Для начала необходимо определить количество частиц в системе. Наиболее простым вариантов является использование постоянного количества частиц на каждом шаге.

2.2.1.3 Атрибуты частиц.

У системы частиц имеется набор собственных атрибутов: средний цвет и отклонение от него, положение в пространстве и т.д.

Сначала рассмотрим атрибуты каждой, отдельно взятой частицы:

1. положение в трехмерном пространстве;
2. начальная скорость;
3. цвет;
4. время жизни.

Параметры всей системы частиц так же управляют атрибутами каждой, отдельно взятой частицы. У каждой частицы есть скорость, которая определяется как средняя скорость системы частиц плюс некоторое коэффициент, умноженный на среднюю вариацию. Система частиц обладает набором всех атрибутов частицы, но при этом они у нее имеются в двух вариантах: среднее значение атрибута и допустимое отклонение атрибута. И для каждой частицы, чтобы получить ее значение атрибута, необходимо взять среднее значение того же атрибута системы частиц плюс некоторая случайная величина в диапазоне [-1, 1], умноженная на значение вариации системы частиц.

2.2.1.4 Динамика частиц.

Отдельные частицы внутри системы частиц двигаются в трехмерном пространстве, изменяют свой цвет, прозрачность и размер. Для движения частицы из кадра в кадр необходимо добавить вектор скорости к вектору положения.

2.2.1.5 «Умирание» частиц

При создании частицы ее время жизни измеряется в кадрах. Как только один кадр посчитан, текущее время жизни увеличивается, и, как только оно достигнет максимально возможного, частица «умирает». Другие механизмы позволяют убивать частицы, как только они перестают вносить вклад в изображение. В данной программе реализован механизм убивания частиц, покидающих пределы сцены. Это позволяет экономить вычислительную мощность.

2.2.1.6 Отображение частиц

Производить синтез изображения необходимо, когда произведены все расчеты, а именно посчитаны положение и внешние параметры каждой частицы. Эта задача является более сложной, чем задача визуализации

объектов, составленных из таких примитивов как: полигоны или кривые поверхности. Так как частицы могут закрывать другие частицы, могут находиться за ними, они могут быть прозрачны, могут отбрасывать тень на другие частицы, более того, они могут сосуществовать в сцене, где есть объекты, составленные из стандартных примитивов. Для упрощения в курсовой работе было принято два утверждения:

1. частицы не пересекаются с другой геометрией в сцене;
2. частица – это источник освещения.

Благодаря этим утверждения не было необходимости обрабатывать случай, когда частица пересекает треугольник, и на этом месте появляется жесткая граница. А также при таком подходе нет необходимости сортировать частицы, а они могут быть сразу выведены в буфер кадра, как только их значения посчитаны.

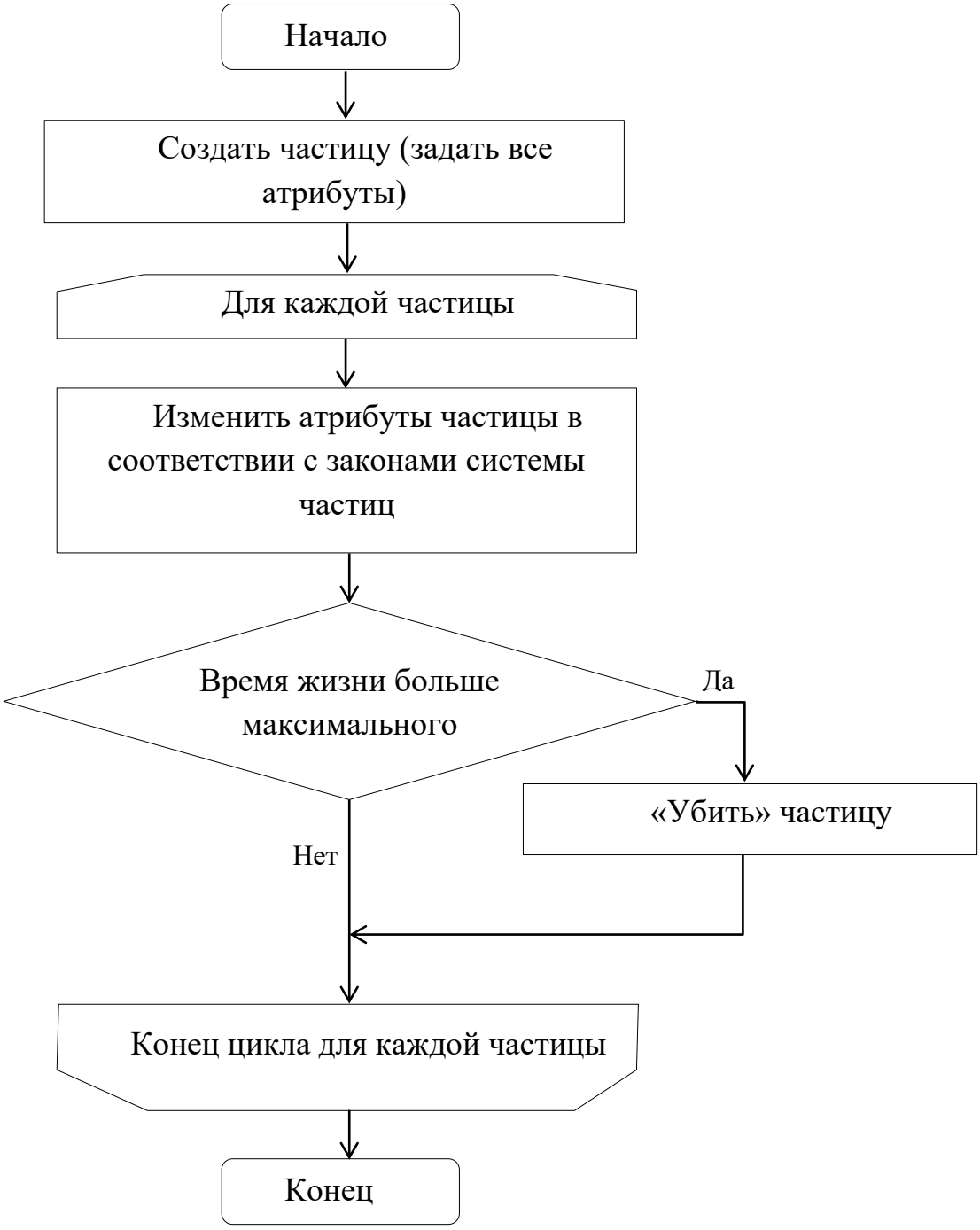

Рис. 2.2.1. Структура алгоритма системы частиц.

2.2.2 Алгоритм z-буфера.

1. Заполнить z-буфер элементами с фоновым значением цвета и минимальным значением z;
2. все точки системы частиц занести в z-буфер;
3. декомпозировать каждый многоугольник на треугольники:
    a. запомнить первую вершину;
    b. запомнить треугольник, который первая вершина образует с каждой следующей парой вершин;
4. Для каждого треугольника:
    a. найти *ymin* и *ymax*;
    b. пустить сканирующую строку от *ymin* до *ymax*;
    c. для каждой сканирующей строки найти глубину *z* каждой точки *(x, y);*

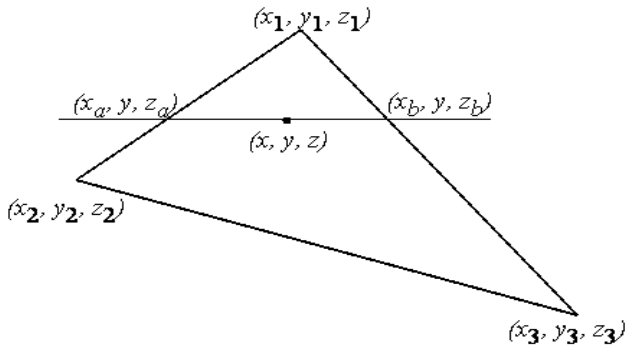

Рис. 2.2.2.1. Сканирующая строка по грани

Для рисунка *y* меняется от y_1 до y_2 и далее до y_3, при этом для каждой строки определяется x_a, z_a, x_b, z_b:

$$x_a = x_1 + (x_2 - x_1) * \frac{(y - y_1)}{(y_2 - y_1)};$$
$$x_b = x_1 + (x_3 - x_1) * \frac{(y - y_1)}{(y_3 - y_1)};$$
$$z_a = z_1 + (z_2 - z_1) * \frac{(y - y_1)}{(y_2 - y_1)};$$
$$z_b = z_1 + (z_3 - z_1) * \frac{(y - y_1)}{(y_3 - y_1)};$$

    d. получить *z* для каждого x из (x_a, x_b) сравнить со значением в z-буфере:
        a) если посчитанное значение больше – занести его и соответствующий пикселю цвет в z-буфер;
        b) иначе никаких действий не производить.

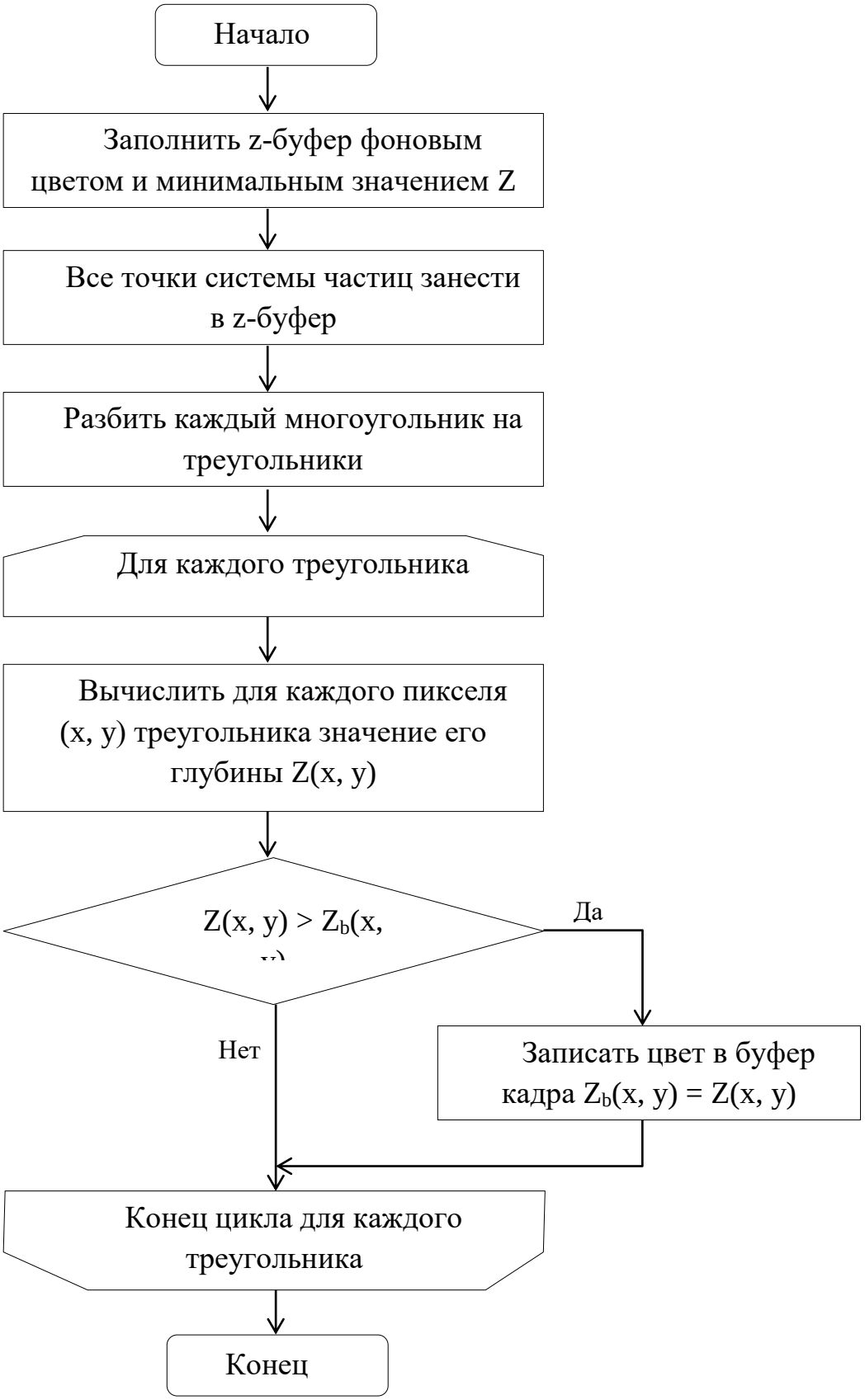

Рис. 2.2.2.2. Алгоритм z-буфера.

3 Технологическая часть.

3.1 Выбор и обоснование языка и среды программирования.

Для разработки данной программы применён язык C++ и среда Qt Creator по следующим причинам:

- широкие возможности реализации алгоритмов с высокой эффективностью;
- значительный набор стандартных классов и процедур;
- большое количество виджетов и их гибкость;
- удобство отладки и редактирования программы;
- активное использование сигнально-слотовой связи для реализации асинхронного взаимодействия классов.

3.2 Структура классов программы

В силу того, что при написании программы использовалась парадигма объектно-ориентированного программирования, особое внимание при рассмотрении ее структуры должно быть уделено структуре ее классов.

Все классы, присутствующие в программе, можно разделить на несколько частей по выполняемым функциям:

- менеджер – связующее звено между пользователем и объектом, менеджером которого является;
- камера – «смотрит» сцену с разных сторон;
- полотно – пространство для рисования;
- частица – «живёт» по законам, определённым для своего класса;
- система частиц – контролирует жизнь частиц, задаёт правила существования;
- сцена – хранит в себе объекты, является объектом обзора камеры;
- объекты сцены;
- алгоритмы визуализации – визуализируют то, что «видит» камера;
- пользовательские структуры данных – для функционирования программы должным образом;
- классы интерфейса – позволяют задавать параметры работы программы.

3.2.1 Менеджеры камеры и системы частиц

В программе работают два менеджера: менеджер *камеры* и менеджер *системы частиц*, так как пользователь осуществляет взаимодействие посредством своего интерфейса только с этими классами.

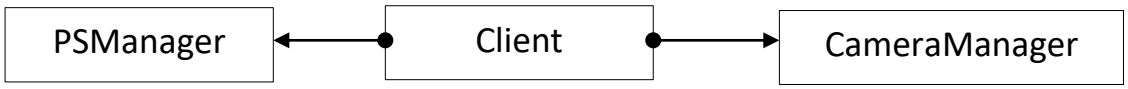

Рис. 3.2.1. Иерархия менеджеров

Назначение каждого из этих классов дается в таблице 3.2.1:

Таблица 3.2.1. Назначение классов-менеджеров.

Класс	**Комментарии**
PSManager	Реализует взаимодействие с системой частиц: создаёт указанные пользователем осадки, определяет поведение системы по тику таймера
CameraManager	Реализует взаимодействие с камерой: её создание, удаление, аффинные преобразования координат, отправку объектов сцены для обработки визуализатором.

3.2.2 Камера

Выполняет указания менеджера камеры, имеет доступ к сцене.

Рис. 3.2.2. Иерархия камеры

В таблице 3.2.2 даётся назначение камеры:

Таблица 3.2.2. Назначение камеры.

Класс	Функции
Camera	• toHomePosition • move • rotateX • rotateY • rotateZ • scale • sendPoints • sendPolygons

3.2.3 Частица

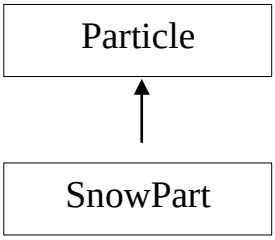

Рис. 3.2.3. Иерархия частицы

Класс *Particle* обладает полным набором свойств, атрибутов и методов частицы. В наследуемых от него классе снега переопределяются методы и устанавливаются атрибуты, специфические для данного типа частиц.

3.2.4 Система частиц

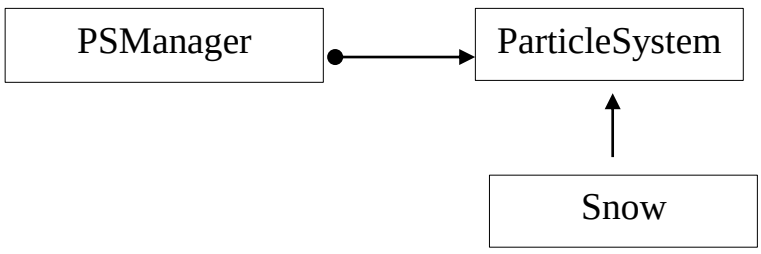

Рис. 3.2.4. Иерархия системы частиц

В таблице 3.2.4 даётся назначение этих классов:

Таблица 3.2.4. Назначение классов, реализующих систему частиц.

Класс	**Комментарии**
ParticleSystem	Реализует взаимодействие с системой частиц: создаёт указанные пользователем осадки, определяет поведение системы по тику таймера, инкапсулирует атрибуты, свойства и методы общего понятия «система частиц».
Snow	Определяет набор свойств, атрибутов и методов, характерных для снега. Хранит в динамическом списке коллекцию частиц снега.

3.2.5 Сцена

Для быстрого и удобного доступа к сцене при учёте того обстоятельства, что сцена в программе представлена единственным экземпляром, инициализируемым только один раз, а далее используемым на протяжении всей работы программы, было принято решение реализовать паттерн Singleton («одиночка»), который бы хранил и предоставлял доступ к объекту сцены.

Сцена хранит массивы точек системы частиц и полигонов, из которых состоят объекты сцены. Имеет слоты предоставления и установки этих массивов по соответствующим сигналам камеры.

3.2.6 Алгоритмы визуализации

Абстрактный визуализатор предоставляет в распоряжение пользователя слот, который устанавливает изображение, сгенерированное визуализатором. Пользователь, запросив это изображение, может передать его на полотно для отрисовки на форме.

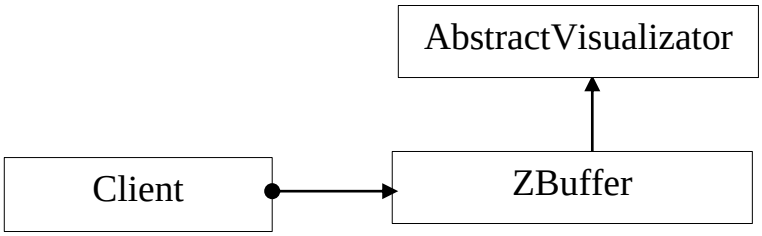

Рис. 3.2.6. Иерархия визуализаторов

Таблица 3.2.6. Класс z-буфера.

Класс	**Функции**
ZBuffer	Методы класса реализуют алгоритм z-буфера, описанный в конструкторской части. Что касается технологической части, классу предоставлены сигналы, адресованные пользователю, для предоставления информации о текущем размере полотна и цвете фона. Посредством слотов реализован приём от пользователя объектов сцены, который, в свою очередь, получил их от камеры.

3.2.7 Пользовательские структуры данных

Таблица 3.2.7. Типы и структуры данных пользователя и их назначение.

Название	Тип	Назначение
Intensity	Перечисление	Задаёт интенсивность осадков от 0 (отсутствие) до 4 (максимум)
Point	Класс	Инкапсулирует работу с точками в трёхмерном пространстве
Color	Класс	Цвет в схеме RGB. Имеет несколько конструкторов и атрибуты для хранения значений составляющих цвета. Имеет метод, возвращающий объект QColor, необходимый для рисования на виджете в среде Qt Creator.
Size	Класс	Описывает размеры трёхмерных объектов.
PointsToSend	Класс	Хранит в себе точку в трёхмерном пространстве и её цвет. Используется для отправки данных.
Cell	Структура	Структура элемента z-буфера. Хранит глубину пиксела и его цвет.
Triangle	Класс	Класс, описывающий работу с треугольниками. Чтобы работать с объектами другой формы, их разбивают на треугольники, а дальше используют методы данного класса.

3.2.8 Классы интерфейса

Интерфейс данной программы строился путём создания элементов управления, унаследованных от стандартных классов библиотеки Qt: использовались объекты классов QMainWindow, QWidget, QLayout, QGroupBox, QAbstractSlider и QLabel.

3.2.9 Распределение классов по файлам

Таблица 2.9. Распределение классов по файлам.

Класс	**Заголовочный файл**	**Файл реализации**
Менеджеры		
PSManager	particle_system.h	particle_system.cpp
CameraManager	camera.h	camera.cpp
Камера		
Camera	camera.h	camera.cpp
Полотно		
Canvas	canvas.h	canvas.cpp
Частица		
Particle	particle.h	particle.cpp
SnowPart	particle.h	particle.cpp
Система частиц		
ParticleSystem	particle_system.h	particle_system.cpp
Snow	particle_system.h	particle_system.cpp
Сцена		
Scene	scene.h	scene.cpp
SingleScene	scene.h	scene.h
Объекты сцены		
House	scene_object.h	scene_object.cpp
Алгоритмы визуализации		
AbstractVisualizator	visualizator.h	

ZBuffer	visualizator.h	visualizator.cpp
Пользовательские структуры данных		
Point	user_definitions.h	user_definitions.cpp
Color	user_definitions.h	user_definitions.cpp
Size	user_definitions.h	user_definitions.cpp
PointsToSend	user_definitions.h	user_definitions.cpp
Triangle	user_definitions.h	user_definitions.cpp

3.3Общая схема взаимодействия классов.

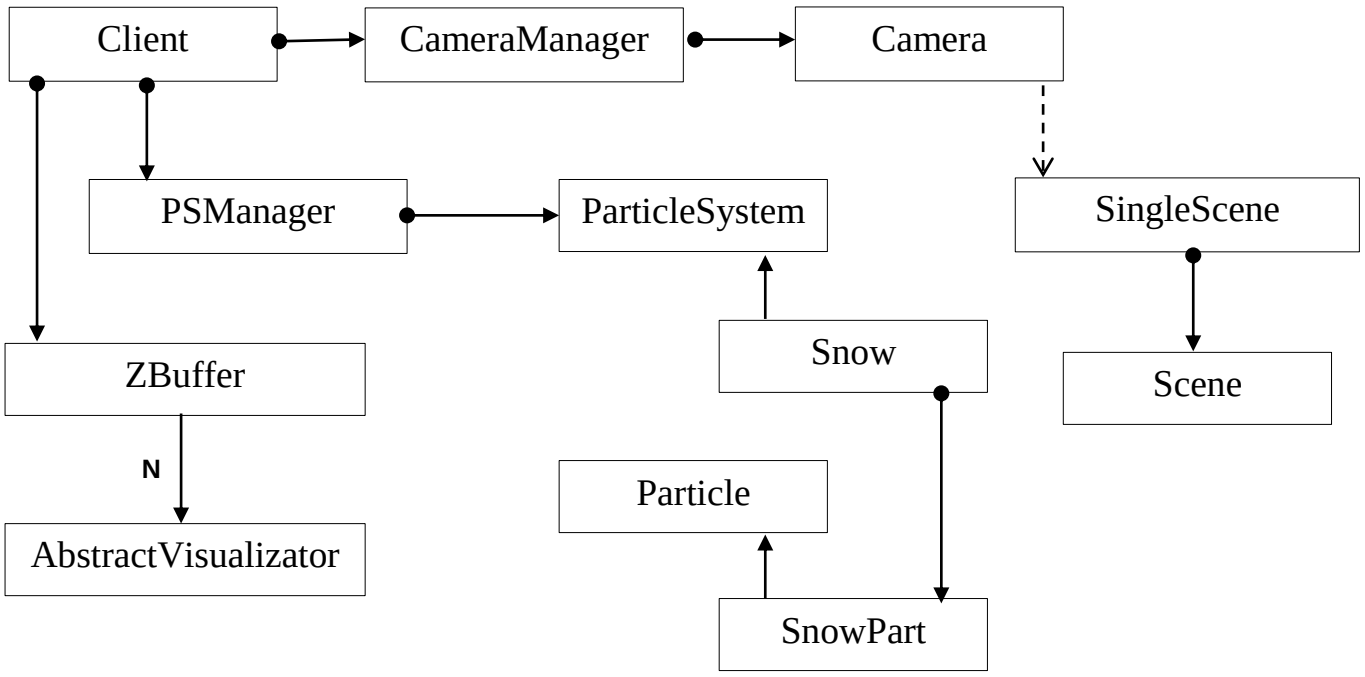

Рис. 3.3. Общая схема взаимодействия классов.

Замечание: использованы следующие обозначения:

- включение класса A в класс B: B ●⟶ A;
- включение классом B массива объектов класса A: B ●—N→ A ;
- наследование класса A классом B: B ⟶ A;
- использование класса A классом B: B - - - - - - ➢ A .

3.4 Пользовательский интерфейс.

Программа имеет русскоязычный интерфейс, организованный в виде набора виджетов, позволяющих пользователю после запуска программы задавать интенсивность снега и смотреть на сцену с любого ракурса.

Доступ ко всем функциям и настройкам осуществляется непосредственно через компоненты основного окна программы.

3.4.1 Установка интенсивности.

Для того чтобы задать интенсивность осадков, необходимо передвинуть ползунок до соответствующего уровня.

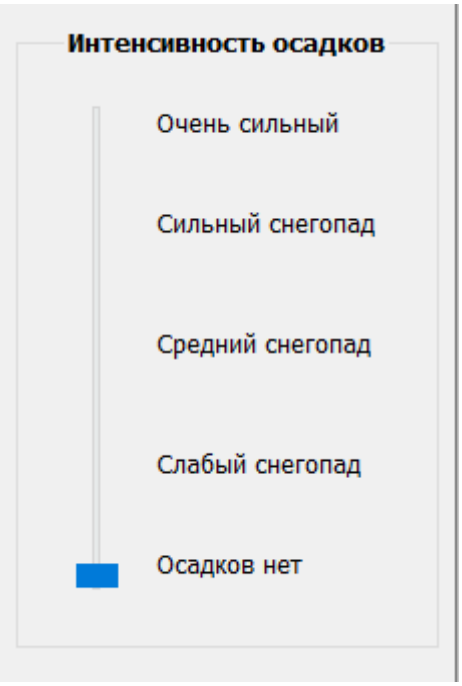

Рис. 3.4.3.1

3.4.2 Обзор сцены.

Для просмотра сцены с разных ракурсов предоставляются следующие возможности:

- зумирование:
  - удаление от сцены – *«-»;*
  - приближение – *«+»;*
- перемещение сцены вправо, влево, вверх и вниз по области полотна осуществляется соответствующими стрелочками на клавиатуре;
- вращение сцены осуществляется кнопками на NumPad’e:
  - *«8»* - против часовой стрелки относительно оси oX;
  - *«2»* - по часовой стрелке относительно оси oX;

- «6» - против часовой стрелки относительно оси oY;
- «4» - по часовой стрелке относительно оси oY;

Совмещая эти преобразования, можно получить, к примеру, следующий вид (рис. 3.4.3)

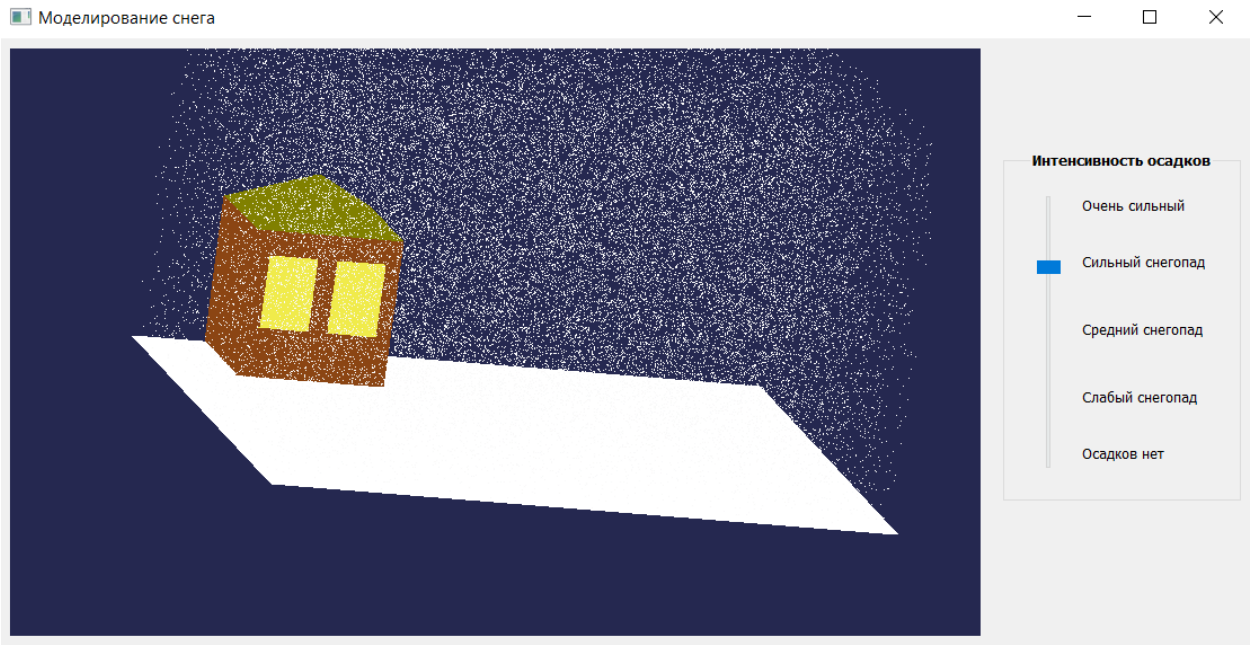

Рис. 3.4.4.1 Комбинированный вид

4 Экспериментально-исследовательская часть.

4.1 Исследование зависимостей времени генерации сцены от интенсивности.

Суть эксперимента заключается в наблюдении зависимости изменения времени генерации частиц и отрисовки сцены от интенсивности выпадения осадков. При этом учитывается и время формирования системы частиц.

Точные результаты исследования представлены в *таблице 4.1*

Интенсивность	Время генерации и отрисовки снега, мс
низкая	54,4
средняя	57,9
высокая	67,6
очень высокая	98,6

Таблица 4.1. Зависимость времени генерации и отрисовки сцены от интенсивности.

Замечание: в соответствии с данными об интенсивности снега приводятся следующие расшифровки:

- низкая: интенсивность снега не превышает снега – 0,1 мм/ч;
- средняя: интенсивность снега не превышает снега – 1 мм/ч;
- высокая: интенсивность снега не превышает снега – 1,2 мм/ч;
- очень высокая: интенсивность снега не превышает снега – 1,6 мм/ч.

Для наглядности результаты представлены на *диаграмме 4.1*.

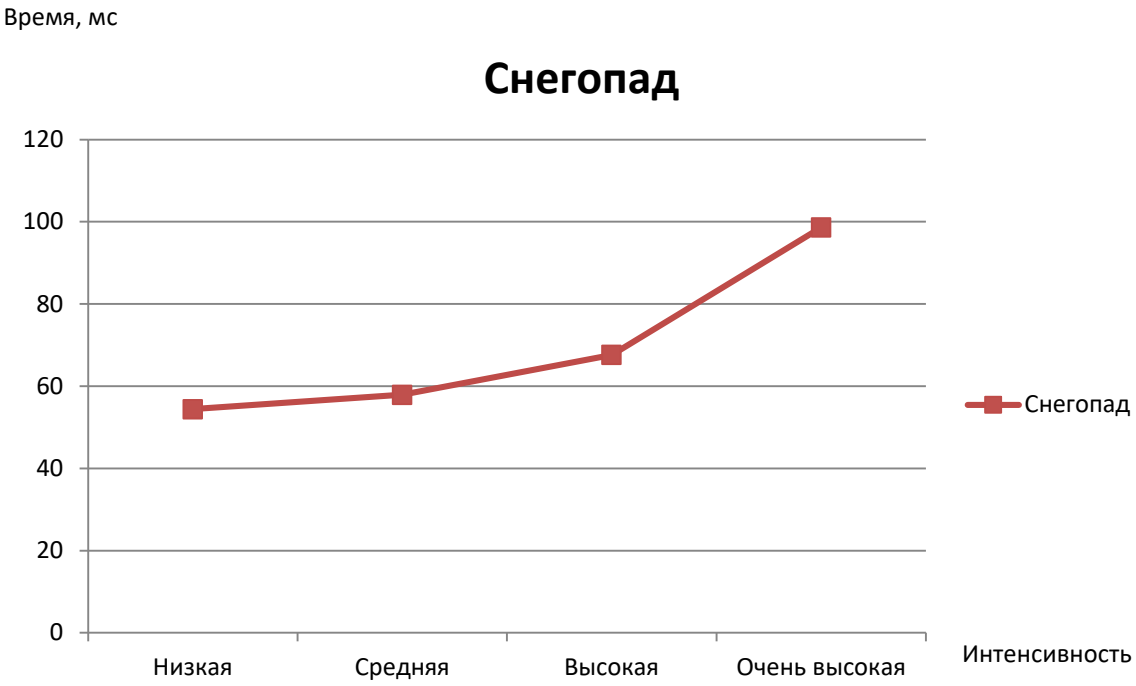

Диаграмма 4.1. Зависимость времени генерации и отрисовки сцены от интенсивности.

Результаты измерения времени отрисовки без учёта времени генерации частиц.

Интенсивность	Время генерации и отрисовки снега, мс
низкая	51,1
средняя	52,8
высокая	54,1
очень высокая	62

Таблица 4.2. Зависимость времени генерации и отрисовки сцены от интенсивности.

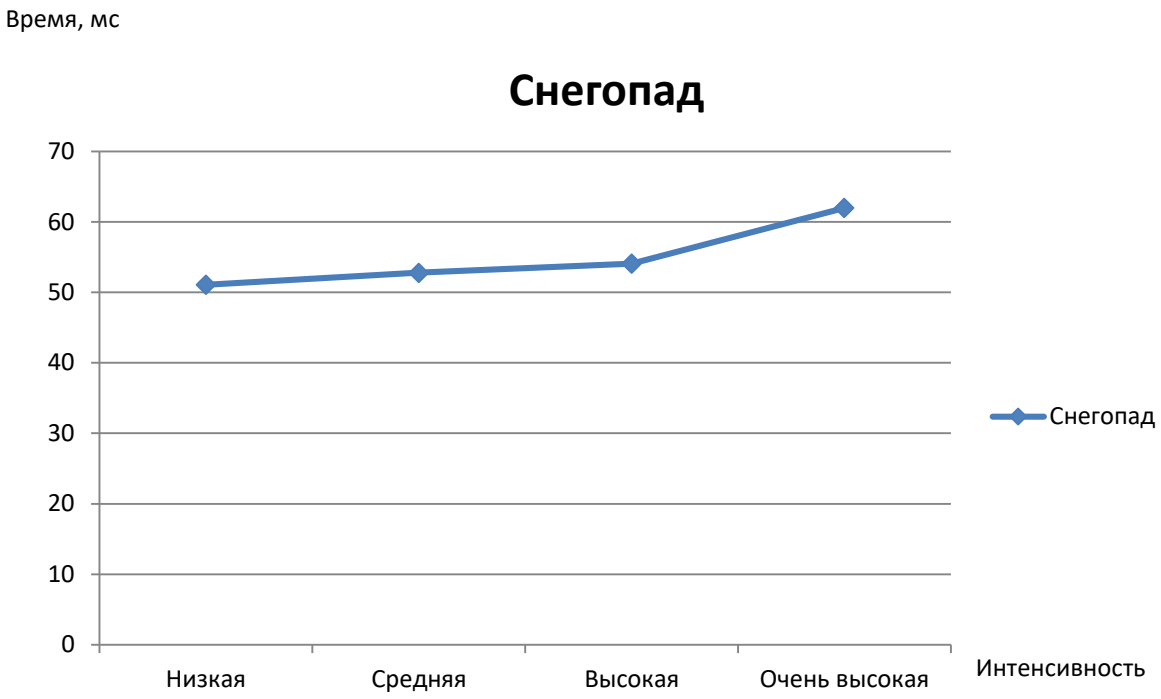

Диаграмма 4.2. Зависимость времени отрисовки сцены от интенсивности.

Заключение.

Разработанный программный комплекс отвечает всем предъявляемым к нему требованиям и обеспечивает возможность моделирования снега путём задания уровня интенсивности выпадения осадков. Проведены исследования работы алгоритмов на различных наборах исходных данных.

Моделирование на основе системы частиц обладает гибкостью и удобством использования, но требует больших ресурсных затрат. Если учитывать внешние силовые поля, оптические характеристики частиц и различную, меняющуюся в течении жизни, форму частицы, то реализация в режиме реального времени будет возможна только на параллельных вычислительных машинах, таких как кластеры.

Алгоритм z-буфера показал себя с лучшей стороны скоростью отрисовки сцены, но он не характеризуется реалистичностью синтезированного изображения из-за отсутствия таких эффектов, как освещённость, отражение, преломление и т.д. Сымитировать подобные эффекты возможно, накладывая на объекты заранее просчитанные текстуры освещённости, что в совокупности с эффективными алгоритмами оптимизации по быстродействию приближает изображение к реальности.

Список использованной литературы

1. Роджерс Д., Алгоритмические основы машинной графики: пер. с англ.— М.: Мир, 1989.— 512 с.: ил.
2. Шлее М., Qt 4.8 Профессиональное программирование на C++. – СПб.:БХВ-Петербург, 2012 – 912 с.:
3. Страуструп Б., Язык программирования C++. Специальное издание. Пер. с англ. – М.:Издательство Бином, 2012 г. – 1136 с.: ил.
4. Проблемы трассировки лучей – из будущего в реальное время. [Электронный ресурс]. – Режим доступа: https://nvworld.ru/articles/ray-tracing/3/
5. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж., Приёмы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с.: ил. (Серия «Библиотека программиста»)
6. Костикова Е., Методы интерактивной визуализации динамики жидких и газообразных сред. – М: МГУ, 2009 – с. 20-23.
7. Particle Systems. [Электронный ресурс]. – Режим доступа: http://homepages.inf.ed.ac.uk/tkomura