титульный лист

Оглавление

Введение

В современном мире компьютерная графика находит свое применение в самых различных областях. Типичный пример – это кино и игры.

Алгоритмы получения реалистичных изображений на сегодняшний день получают очень много внимания. Эти алгоритмы сложны в реализации и являются одними из самых затратных по ресурсам. Они должны учитывать не только формы и текстуры объектов, но и множество других физических явлений, таких как преломление, отражение, рассеивание света.

Очевидно, что получение более качественного изображения на выходе алгоритма, требует больше времени и памяти для синтеза. Хотя для генерации статических изображений это не является проблемой, при создании динамической сцены могут возникать большие сложности, так как каждый раз на очередном временном интервале придется рассчитывать все заново.

Целью данной курсовой работы является создание реалистичной сцены, визуализирующей такое явление, как распространение света в неравномерно задымленном помещении. Явление будет продемонстрировано на примере комнаты, в которой находятся источник дыма и источник света.

Для достижения поставленной цели, необходимо решить следующие задачи:

— проанализировать существующие алгоритмы построения изображения и выбрать из них те, что в лучше помогут в решении поставленной задачи;

— описать модель трехмерной сцены, в том числе и объекты, из которых она состоит;

— реализовать выбранные алгоритмы;

— разработать программу для отображения сцены;

— провести эксперимент по замеру производительности полученного программного обеспечения.

1 Аналитическая часть

В данном разделе представлено описание объектов сцены, а также обоснован выбор алгоритмов, которые будут использованы для ее визуализации.

1.1 Формализация объектов сцены

Сцена состоит из следующих объектов:

— точечного источника света — задается положением в пространстве и интенсивностью;

— источника дыма — задается положением в пространстве;

— модели дыма;

— комнаты — пространство в пределах, которого будет распространяться дым;

— предметов интерьера – объекты внутри комнаты, мешающие распространению дыма и света.

1.2 Обоснование выбора формы задания трехмерных моделей

Отображением формы и размеров объектов являются модели. Обычно используются три формы задания моделей.

1. Каркасная (проволочная) модель. Одна из простейших форм задания модели, так как мы храним информацию только о вершинах и ребрах нашего объекта.

2. Поверхностная модель. Поверхностная модель объекта — это оболочка объекта, пустая внутри. Такая информационная модель содержит

данные только о внешних геометрических параметрах объекта. При этом могут использоваться различные типы поверхностей, ограничивающих объект, такие как полигональные модели, поверхности второго порядка и др.

3. Объемная (твердотельная) модель. При твердотельном моделировании учитывается еще материал, из которого изготовлен объект, т.е. у нас есть информация о том, с какой стороны поверхности расположен материал. Для решения данной задачи лучше всего подойдет поверхностная модель. Каркасные модели могут привести к неправильному восприятию формы объекта, а реализация объемной модели избыточна и потребует дополнительного количества ресурсов.

1.2.1 Способы задания поверхностных моделей

На следующем шаге необходимо определиться со способом задания поверхностной модели.

1. Аналитический способ. Этот способ задания модели характеризуется описанием модели объекта, которое доступно в неявной форме, то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от параметра.

2. Полигональная сетка. Данный способ характеризуется совокупностью вершин, граней и ребер, которые определяют форму многогранного объекта в трехмерной компьютерной графике.

1.2.2 Способы хранения поверхностных моделей

Существует несколько способов хранения информации о сетке.

1. Список граней. Объект – это множество граней и множество вершин. В каждую грань входят как минимум 3 вершины.

2. «Крылатое» представление. Каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые её касаются.

3. Полурёберные сетки. То же «крылатое» представление, но информация обхода хранится для половины грани.

4. Таблица углов. Это таблица, хранящая вершины. Обход заданной таблицы неявно задаёт полигоны. Такое представление более компактно и более производительно для нахождения полигонов, но, в связи с тем, что вершины присутствуют в описании нескольких углов, операции по их изменению медленны.

5. Вершинное представление. Хранятся лишь вершины, которые указывают на другие вершины. Простота представления даёт возможность проводить над сеткой множество операций.

Вывод

Одним из важнейших факторов в выборе способа реализации модели является скорость вычисления преобразований над объектом. Поэтому для задания модели наилучшим представлением будет полигональная сетка. При этом способ хранения полигональной сетки — список граней. Такое представление является наиболее простым. Также этот метод дает возможность быстрого преобразования модели, потому что структура будет включать в себя список вершин.

1.3 Обоснование выбора формы задания дыма

Дым представляет собой дисперсную систему, состоящую из твердых частиц, находящихся во взвешенном состоянии.

1.3.1 Уравнения Навье-Стокса

Поведение частиц дыма аналогично поведению легких газов. Частицы дыма легкие и мелкие, и при испускании из неподвижного источника, такого как свеча, не будут иметь скорости. Движение частиц в основном вызывают окружающие силы. Частицы дыма не влияют на свое собственное движение, а зависят исключительно от внешних сил, толкающих и тянущих их в разных направлениях. Дым, как правило, вытягивается вверх из-за малого веса частиц, которые следуют за теплым воздушным потоком, поднимающимся вверх. Дым, как и жидкости, также будет растекаться по поверхностям при взаимодействии с объектами. Для моделирования такого поведения часто используется вычислительная гидродинамика. Обычно используемыми уравнениями для решения CFD являются уравнения Навье-Стокса и Эйлера. Оба уравнения описывают, как изменяются скорости и массы. При условии, что вязкость равна нулю, удаление теплопроводности из уравнения Навье-Стокса дает уравнение Эйлера [1].

$$\frac{\partial \vec{u}}{\partial x} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + \vec{f}, \tag{1.1}$$

где ρ — плотность,
p — давление,
$\vec{f}$ — произвольная сила,
$\vec{u}$ — скорость частицы.

1.3.2 Система частиц

Движение частиц [2].

Основное движение дыма происходит вверх по спирали. Очевидно, что для достижения более хаотичного движения, потребуется добавление 3D-шума. Спираль, по которой движутся частицы, управляется простым векторным полем 1.2.

$$F(xpos, ypos, zpos) = \{2 \cdot zpos \cdot ypos, yvel, -2 \cdot xpos \cdot ypos\}, \tag{1.2}$$

где *xpos*, *ypos* и *zpos* - мировые координаты частицы, а *yvel* - скорость, с которой частицы движутся вверх.

Это значение одинаково везде в пространстве. Сила тяжести не оказывает значительное влияние на частицы дыма и не принимается во внимание. Также не происходит столкновений между частицами дыма.

Частицам присваиваются скорости в соответствии с их положением в векторном поле. Затем вычисляются их новые позиции.

$$p(t + dt) = p(t) + \vec{v} \cdot dt, \tag{1.3}$$

где $p(t)$ - положение частицы в момент времени t, а $\vec{v}$ - трехмерный вектор, представляющий скорость частицы, заданную векторным полем.

Конечно, это не конечное положение частицы. Шум добавляется для создания более хаотичного движения. Все частицы рождаются со скоростью (0,1,0). Частицы перемещаются в соответствии с их текущей скоростью, прежде чем им присваивается новая скорость.

Шум частиц.

Шум, используемый для частиц, отличается от обычного шума главным образом возвращаемым из него значением. Обычный шум возвращает значение от нуля до единицы, тогда как шум, используемый для системы частиц, возвращает 3D-вектор.

Время жизни частицы.

Все частицы абсолютно идентичны при рождении. В промежутках между кадрами сохраняются положение, скорость и возраст частиц. В начале каждого кадра рождается фиксированное количество частиц. Частицы удаляются, когда они превысили максимальный возраст частиц, который является глобальным и, следовательно, одинаковым для всех частиц.

Вывод

Хотя CFD метод даст более реалистичное изображение, он очень сложен в реализации и требует большой вычислительной мощности. Из-за таких серьезных недостатков для выполнения проекта был выбран метод, основанный на частицах.

1.4 Выбор алгоритма удаления невидимых ребер и поверхностей

Выбранный алгоритм удаления невидимых ребер должен обладать некоторым радом свойств, а именно:

— из-за большого количества частиц, алгоритм должен использовать минимум памяти;

— алгоритм должен быть достаточно быстрым;

— как результат работы алгоритм должен выдавать максимально реалистичное изображение.

1.4.1 Алгоритм, использующий Z-буфер

Суть данного алгоритма [3] заключается в использовании двух буферов: буфера кадра, в котором хранятся атрибуты каждого пикселя, и Z-буфера, в котором хранится информация о координате Z для каждого пикселя.

Первоначально в Z-буфере находятся минимально возможные значения Z, а в буфере кадра располагаются пиксели, описывающие фон. Каждый многоугольник преобразуется в растровую форму и записывается в буфер кадра.

В процессе подсчета глубины нового пикселя, он сравнивается с тем значением, которое уже лежит в Z-буфере. Если новый пиксель расположен ближе к наблюдателю, чем предыдущий, то он заносится в буфер кадра и происходит корректировка Z-буфера.

Для решения задачи вычисления глубины Z каждый многоугольник описывается уравнением 1.4.

$$ax + by + cz + d = 0. \tag{1.4}$$

При $c = 0$ многоугольник для наблюдателя вырождается в линию.

Для некоторой сканирующей строки $y = const$, поэтому имеется возможность рекуррентно высчитывать z' 1.5.

$$\forall x' = x + dx : z' - z = -\frac{ax' + d}{c} + \frac{ax + d}{c} = \frac{a(x - x')}{c} \tag{1.5}$$

Получим $z' = z - \frac{a}{c}$, так как $x - x' = dx = 1$.

При этом стоит отметить, что для невыпуклых многогранников предварительно потребуется удалить не лицевые грани.

Преимущества

— простота реализации;

— оценка трудоемкости линейна.

Недостатки

— сложная реализация прозрачности;

— большой объем требуемой памяти.

1.4.2 Алгоритм обратной трассировки лучей

Алгоритмы трассировки лучей на сегодняшний день считаются наиболее мощными при создании реалистичных изображений. Изображение формируется из-за того, что свет попадает в камеру.

Выпустим из источников света множество лучей (первичные лучи). Часть этих лучей не встретит никаких препятствий, а часть попадет на объекты. На них лучи могут преломляться и отражаться. При этом часть энергии луча поглотится. Преломленные и отраженные лучи образуют новое поколение лучей. Далее эти лучи опять же преломятся, отразятся и образуют новое поколение лучей. В конечном итоге часть лучей попадет в камеру и сформирует изображение. Это описывает работу прямой трассировки лучей.

Метод обратной трассировки лучей позволяет значительно сократить перебор световых лучей. В этом методе отслеживаются лучи не от источников, а из камеры. Таким образом, трассируется определенное число лучей, равное разрешению картинки.

Преимущества

— высокая реалистичность синтезируемого изображения;

— работа с поверхностями в математической форме;

— вычислительная сложность слабо зависит от сложности сцены.

Недостатки — скорость выполнения.

1.4.3 Алгоритм Робертса

Данный алгоритм работает в объектном пространстве, решая задачу только с выпуклыми телами. Алгоритм выполняется в 3 этапа.

Этап подготовки исходных данных.

На данном этапе должна быть задана информация о телах. Для каждого тела сцены должна быть сформирована матрица тела V. Размерность матрицы — $4n$, где n — количество граней тела.

Каждый столбец матрицы представляет собой четыре коэффициента уравнения плоскости $ax + by + cz + d = 0$, проходящей через очередную грань.

Матрица тела должна быть сформирована корректно, то есть любая точка, расположенная внутри тела, должна располагаться по положительную сторону от каждой грани тела. В случае, если для очередной грани условие не выполняется, соответствующий столбец матрицы надо умножить на -1.

Этап удаления рёбер, экранируемых самим телом.

На данном этапе рассматривается вектор взгляда $E = \{0, 0, -1, 0\}$. Для определения невидимых граней достаточно умножить вектор E на матрицу тела V. Отрицательные компоненты полученного вектора будут соответствовать невидимым граням.

Этап удаления невидимых рёбер, экранируемых другими телами сцены.

На данном этапе для определения невидимых точек ребра требуется построить луч, соединяющий точку наблюдения с точкой на ребре. Точка

будет невидимой, если луч на своём пути встречает в качестве преграды рассматриваемое тело.

Преимущества

— работа в объектном пространстве;

— высокая точность вычисления.

Недостатки

— рост сложности алгоритма – квадрат числа объектов;

— тела сцены должны быть выпуклыми (усложнение алгоритма, так как нужна будет проверка на выпуклость);

— сложность реализации.

1.4.4 Алгоритм художника

Предназначен для изображения произвольных поверхностей. Принцип его работы заключается в выводе на экран объектов по мере их приближения к наблюдателю. Наиболее распространенная реализация алгоритма – сортировка по глубине, которая заключается в том, что произвольное множество граней сортируется по ближнему расстоянию от наблюдателя, а затем отсортированные грани выводятся на экран в порядке от самой дальней до самой ближней.

Данный метод работает лучше для построения сцен, в которых отсутствуют пересекающиеся грани.

Преимущества — требование меньшей памяти, чем, например, алгоритм Z-буфера.

Недостатки

— невысокая реалистичность изображения;

— сложность реализации при пересечения граней на сцене.

1.4.5 Алгоритм Варнока

Алгоритм Варнока является одним из примеров алгоритма, основанного на разбиении картинной плоскости на части, для каждой из которых исходная задача может быть решена достаточно просто. Поскольку алгоритм Варнока нацелен на обработку картинки, он работает в пространстве изображения.

В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на фрагменты до тех пор, пока содержимое фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения. Сравнивая область с проекциями всех граней, можно выделить случаи, когда изображение, получающееся в рассматриваемой области, определяется сразу:

— проекция ни одной грани не попадает в область;

— проекция только одной грани содержится в области или пересекает область, то в этом случае проекции грани разбивают всю область на две части, одна из которых соответствует этой проекции;

— существует грань, проекция которой полностью накрывает данную область, и эта грань расположена к картинной плоскости ближе, чем все остальные грани, проекции которых пересекают данную область, то в данном случае область соответствует этой грани.

Если ни один из рассмотренных трех случаев не имеет места, то снова разбиваем область на четыре равные части и проверяем выполнение этих условий для каждой из частей. Те части, для которых таким образом не удалось установить видимость, разбиваем снова и т. д.

Преимущества — меньшие затраты по времени в случае области, содержащий мало информации.

Недостатки

— алгоритм работает только в пространстве изображений;

— большие затраты по времени в случае области с высоким информационным содержимым.

Вывод

Для удаления невидимых линий выбран алгоритм обратной трассировки лучей. Этот алгоритм не только поможет добиться наибольшей реалистичности изображения, но и также позволит легко смоделировать распространение света в пространстве, согласно законам геометрической оптики. Данный алгоритм можно улучшить, если добавить в него обработку особых световых явлений. Помимо всего прочего данный алгоритм позволит построить качественные тени. Также немаловажен тот факт, что алгоритм трассировки лучей намного менее требователен к памяти, чем тот же алгоритм Z-буфера.

1.5 Анализ и выбор модели освещения

Физические модели материалов стараются аппроксимировать свойства некоторого реального материала. Такие модели учитывают особенности поверхности материала или же поведение частиц материала. Эмпирические модели материалов устроены иначе, чем физически обоснованные. Данные модели подразумевают некий набор параметров, которые не имеют физической интерпретации, но которые позволяют с помощью подбора получить нужный вид модели. В данной работе следует делать выбор из эмпирических моделей, а конкретно из модели Ламберта и модели Фонга.

1.5.1 Модель Ламберта

Модель Ламберта моделирует идеальное диффузное освещение, то есть свет при попадании на поверхность рассеивается равномерно во все стороны. Интенсивность диффузного света приблизительно подчиняется закону Ламберта, то есть интенсивность пропорциональна только косинусу угла между направлением падающего света и нормалью к поверхности в точке отражения 1.1.

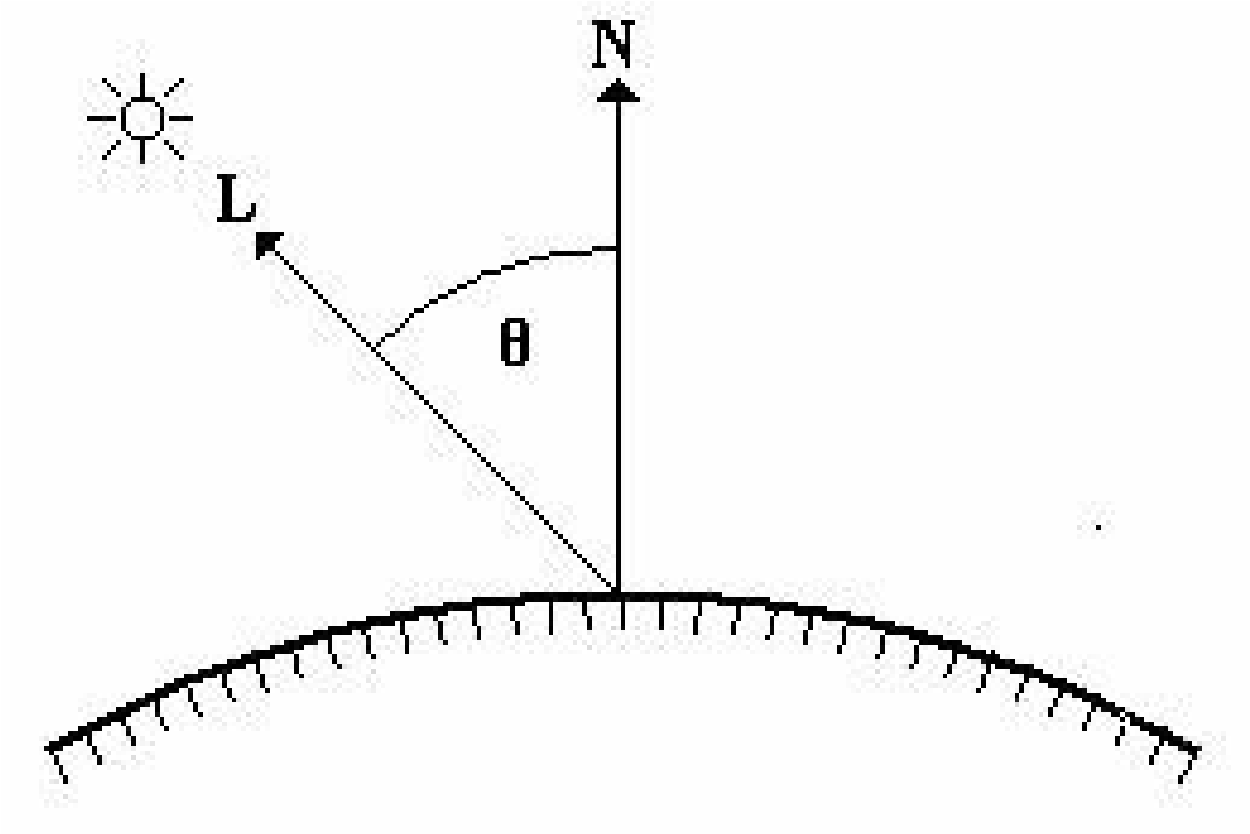

Рисунок 1.1 – Модель освещения Ламберта

$$I_d = I_0 \cdot \cos(\theta) \tag{1.6}$$

$$\cos(\theta) = \vec{L} \cdot \vec{N} \tag{1.7}$$

$$I_d = I_0 \cdot (\vec{L} \cdot \vec{N}) \tag{1.8}$$

Эта модель является одной из самых простых моделей освещения и очень часто используется в комбинации с другими моделями. Она может быть очень удобна для анализа свойств других моделей, за счет того, что ее легко выделить из любой модели и анализировать оставшиеся составляющие.

1.5.2 Модель Фонга

Это классическая модель освещения. Модель представляет собой комбинацию диффузной и зеркальной составляющих. Работает модель таким об-

разом, что кроме равномерного освещения на материале могут появляться блики. Местонахождение блика на объекте определяется из закона равенства углов падения и отражения. Чем ближе наблюдатель к углам отражения, тем выше яркость соответствующей точки. Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения 1.2.

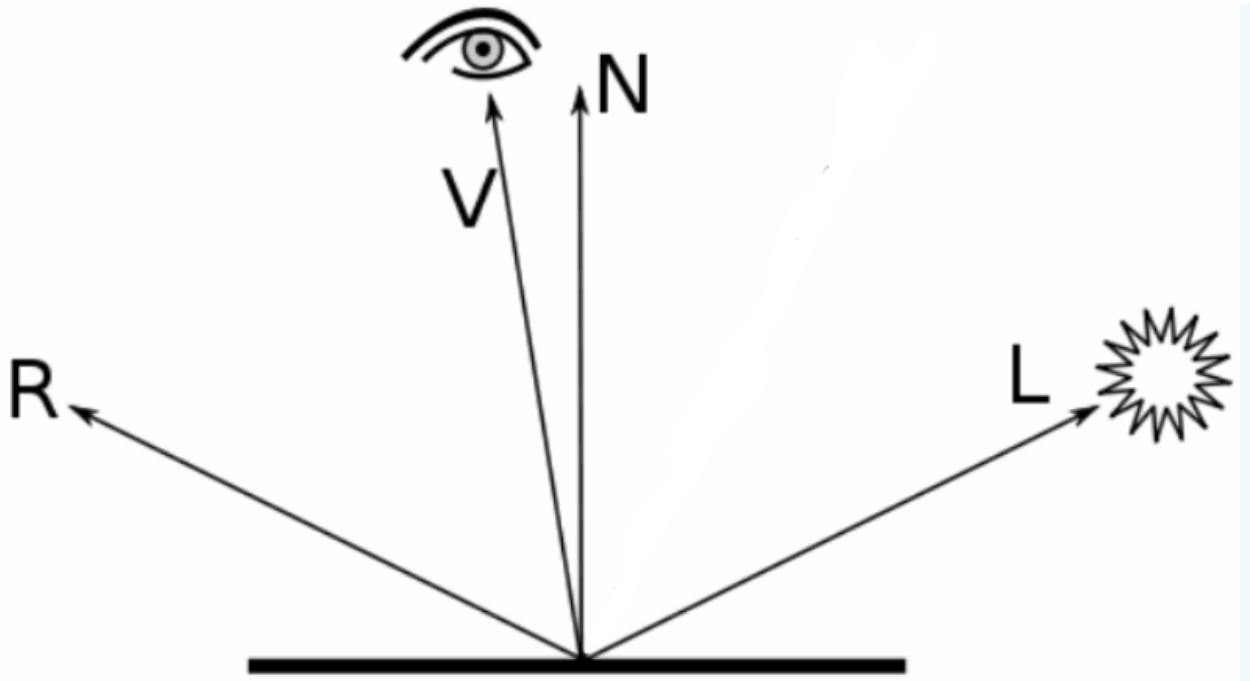

Рисунок 1.2 – Модель освещения Фонга

$$I = I_a k_a + I_d k_d \cdot \left(\vec{N},\ \vec{L}\right) + I_s k_s \cdot (\vec{V}, \vec{R})^p, \tag{1.9}$$

где k_a — коэффициент фонового освещения,
k_d — коэффициент рассеянного освещения,
k_s — коэффициент бликового освещения,
$\vec{N}$ — вектор нормали к поверхности,
$\vec{L}$ — вектор направления падающего луча,
$\vec{R}$ — вектор направления отраженного луча,
$\vec{V}$ — вектор взгляда,
p — контрастность блика.

Вывод

В качестве модели освещения в данной работе была выбрана модель Ламберта из-за своей простоты по сравнению с моделью Фонга. На расчет данных для модели Ламберта потребуется выполнить меньше вычислений, а значит ее реализация будет проще и потребует меньшего количества времени.

Вывод

В данном разделе был проведен анализ алгоритмов удаления невидимых линий и модели освещения, которые возможно использовать для решения поставленных задач, а также выбраны формы задания дыма и объектов на сцене. В качестве ключевого алгоритма, выбран алгоритм обратной трассировки лучей, который будет реализован в рамках курсового проекта по компьютерной графике.

2 Конструкторская часть

В данном разделе будут представлены требования к программному обеспечению, а также схемы алгоритмов, выбранных для решения задачи.

2.1 Разработка алгоритмов

В данном разделе будут представлены схемы реализации выбранных алгоритмов.

2.1.1 Общий алгоритм программы

1. Задать объекты сцены (интерьер комнаты, расположение источников света, характеристики источников дыма).
2. Задать параметры частиц дыма.
3. Запустить симуляцию.
4. Отобразить полученное изображение.

2.1.2 Система частиц для реализации дыма

На рисунке 2.1 показана схема алгоритма реализации движения частицы.

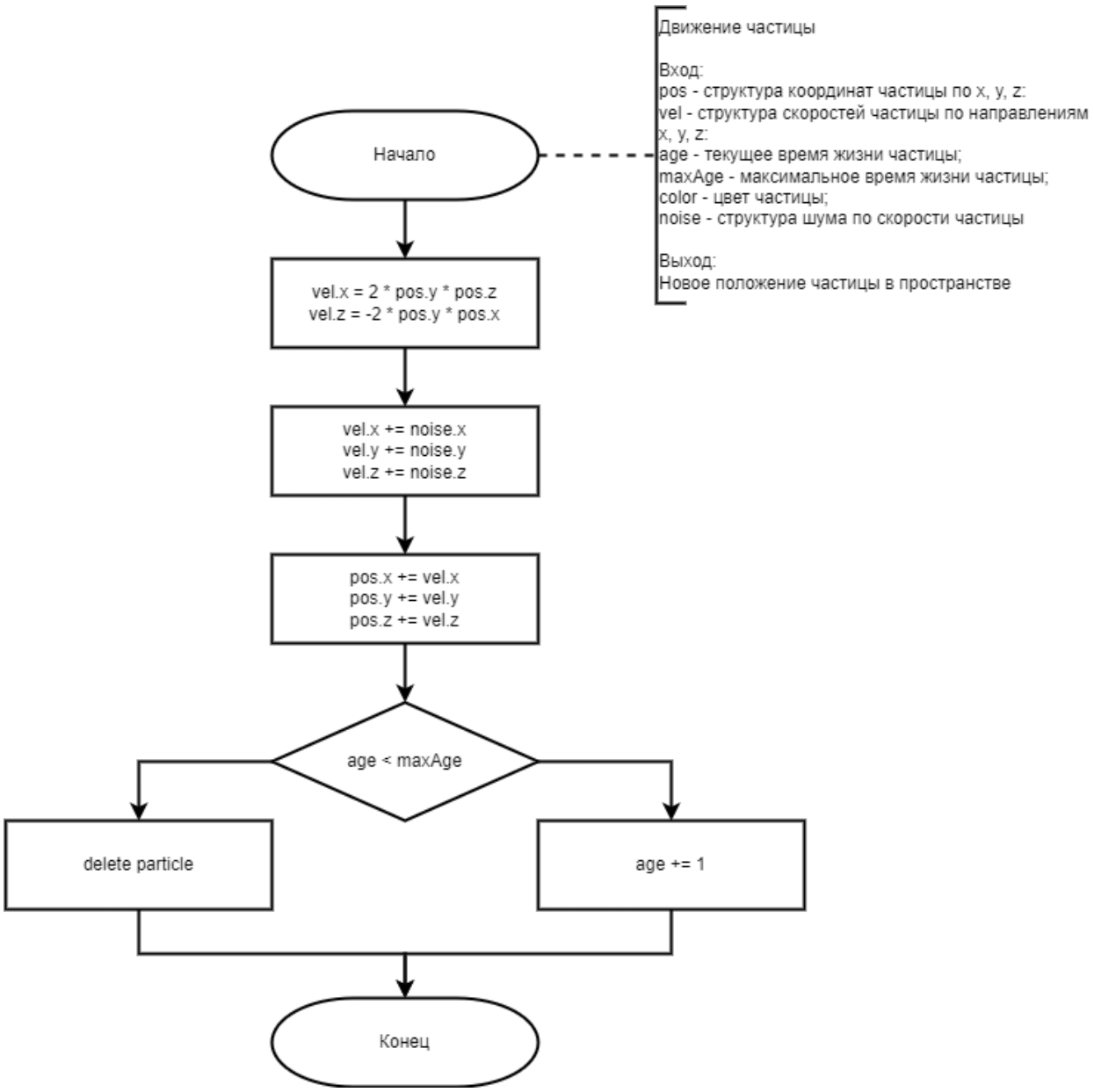

Рисунок 2.1 – Схема алгоритма системы движения частиц

2.1.3 Алгоритм обратной трассировки лучей

На рисунке 2.2 представлена схема алгоритма обратной трассировки лучей.

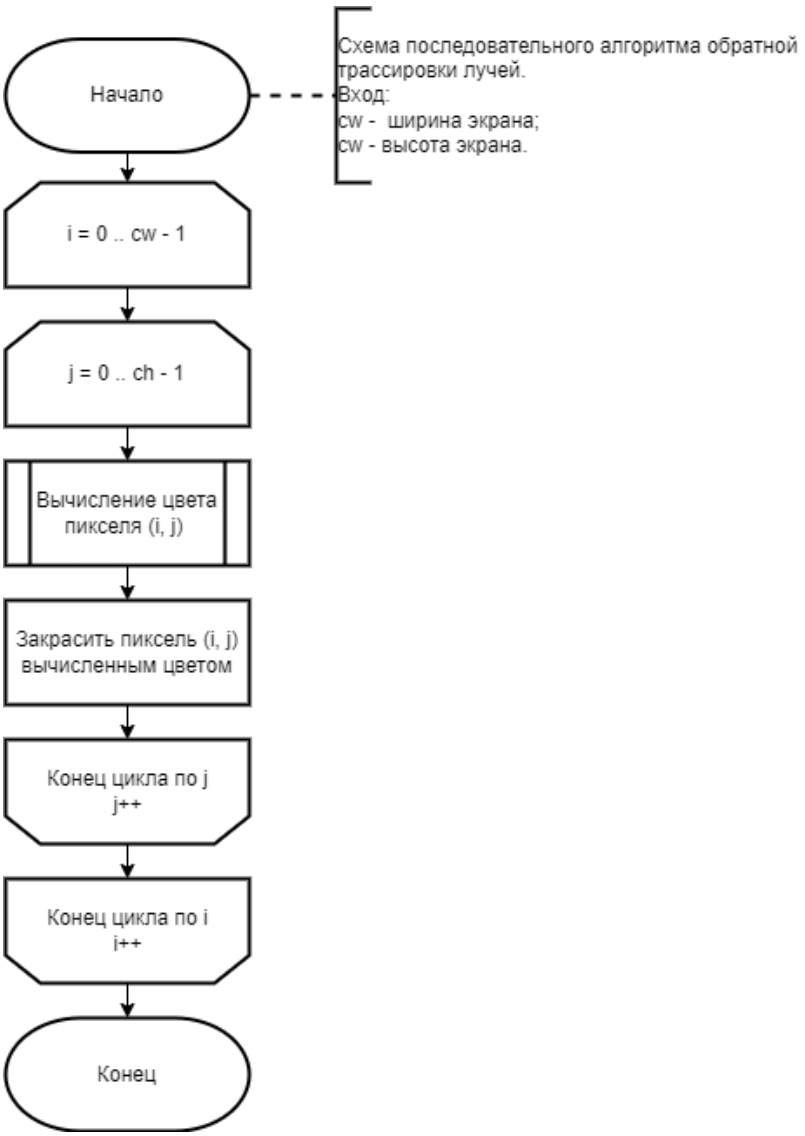

Рисунок 2.2 – Схема алгоритма обратной трассировки лучей

2.2 Структура классов

Классы в программе можно условно разделить на несколько групп по выполняемым функциям.

1. Математические абстракции.
   - Ray - трехмерный луч, задающийся точкой начала луча, направляющим вектором;
   - Point – радиус-вектор точки сцены.
2. Трехмерные объекты.
   - Model – класс хранящий в себе статический декоративный объект;
   - Light – класс источника света, содержащий информацию о положении и интенсивности;
   - Smoker – класс источника дыма, содержащий информацию о скорости генерации частиц и положении.
   - Particle – класс частицы дыма, содержащий информацию о положении, скорости, времени жизни, размерах, цвете частицы.
3. Scene - характеризует набор объектов и их свойств.
4. Интерфейс пользователя. Взаимодействие с интерфейсом происходит через диалоговые окна, которые в свою очередь взаимодействуют с классом Scene.

2.3 Диаграмма классов

На рисунке 2.3 представлена диаграмма классов

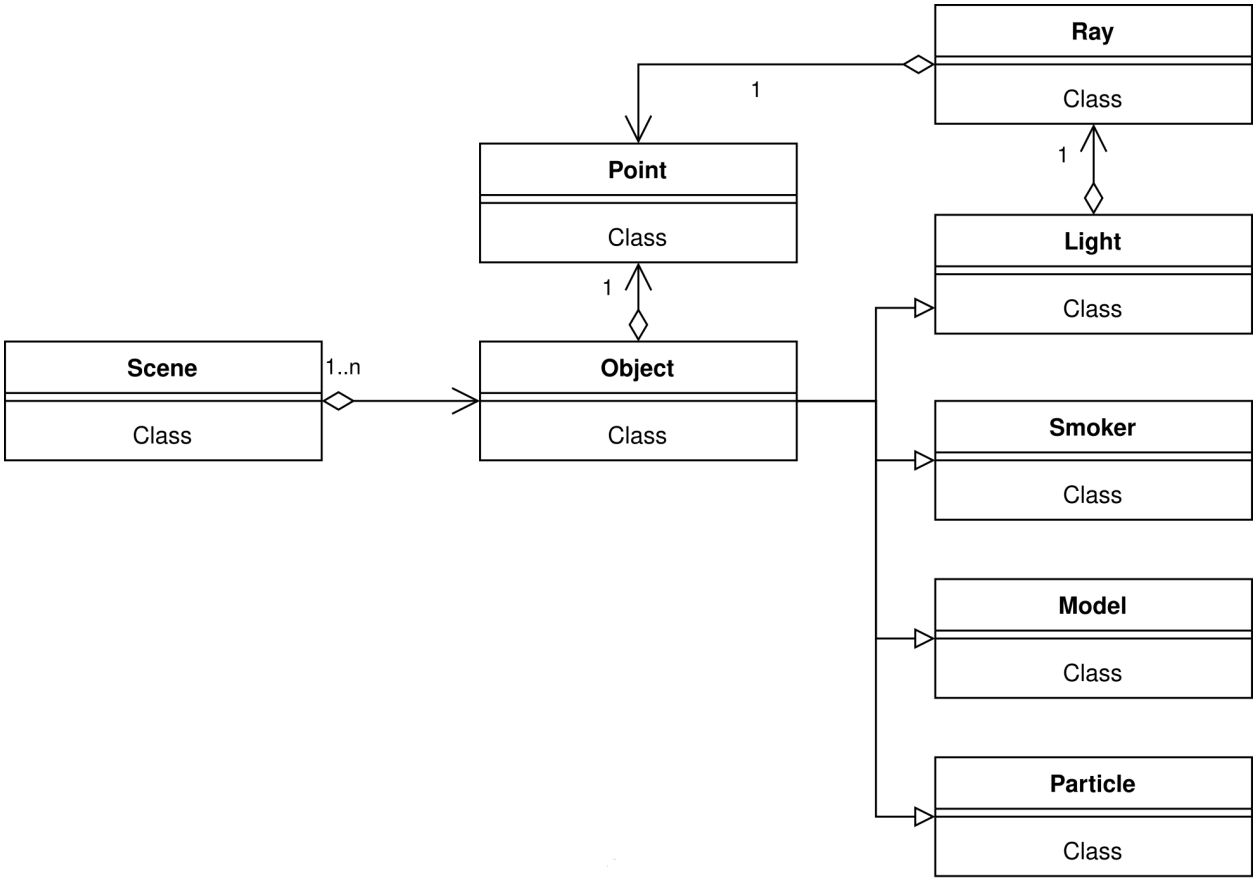

Рисунок 2.3 – Диаграмма классов

Вывод

В данном разделе были рассмотрены требования, которые выдвигаются программному продукту, схемы алгоритмов, а также типы и структуры данных, которые были использованы при реализации программного обеспечения.

3 Технологическая часть

В данном разделе будут приведены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к программному обеспечению

Программа должна предоставлять доступ к функционалу:

— изменение параметров источника дыма в активном режиме: интенсивность генерации частиц дыма, расположение в комнате, начальная скорость вылета частиц;

— изменение параметров частиц, из которых состоит дым, в активном режиме: количество частиц, размер частиц;

— запуск и постановка на паузу сцены;

— вращение, перемещение и масштабирование модели.

3.2 Средства реализации

Для разработки данной программы был выбран язык C# [4]. Данный выбор обусловлен следующими факторами:

— язык предоставляет программисту широкие возможности реализации самых разнообразных алгоритмов. Он обладает высокой эффективностью и большим набором стандартных классов и процедур;

— язык C# является полностью объектно-ориентированным;

— Все необходимые инструменты для реализации поставленной задачи находятся в стандартной библиотеке.

В качестве среды разработки была выбрана Visual Studio 2022. Среда позволяет работать с .NET Framework, что предоставляет широкий функционал.

1. Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов.

2. Автоматическая сборка мусора, как особенность языка C# и фреймворка .NET.

3. JIT-компиляция (Just-In-Time), перекомпиляция исходных файлов во время работы программы.

3.3 Реализации алгоритмов

В листинге 3.1 приведена реализация алгоритма системы движения частиц.

Листинг 3.1 – Метод с реализацией ситемы движения частиц.

```
1  public bool Update(double dtime, Vector3 smokerPosition)
2  {
3      float fdtime = (float) dtime;
4      Vector3 pos = position - smokerPosition;
5      velocity = new(2f * pos.Y * pos.Z, velocity.Y, -2f * pos.X *
           pos.Y);
6      velocity += new Vector3(Random(-3, 3), 0.1f * Random(-1, 1),
           Random(-3, 3));
7      position += velocity * fdtime;
8      if (lifeTime > maxLifeTime)
9          return true;
10     lifeTime += dtime;
11     return false;
12 }
```

В листингах 3.2, 3.3, 3.4, 3.5, 3.6 приведены методы реализующие алгоритм трассировки лучей.

Листинг 3.2 – Основной метод.

```
1  private void RenderPiece(object? param)
2  {
3      int[] p = (int[])(param ?? new[] { 0, 0 });
4      int i = p[0];
5      int j = p[1];
6      Trace t = TraceRay(i - (Cw / 2), -j + (Ch / 2));
7      CastShadow(t);
8      RenderSmoke(t);
9      lbmp.SetPixel(i, j, t.Color);
10 }
```

Листинг 3.3 – Метод создания луча взгляда.

```
1  private Trace TraceRay(int x, int y)
2  {
3      Ray r = new(new(x * Vw / Cw, y * Vh / Ch, d), new(0, 0, 0));
4      return Composite.TraceRay(r, light);
5  }
```

Листинг 3.4 – Метод нахождения пересечения луча взгляда с объектом сцены Composite.TraceRay.

```
1  public Trace TraceRay(Ray ray, LightSource l)
2  {
3      Trace close = Objects[0].Intersection(ray, l);
4      foreach (var obj in Objects)
5      {
6          Trace t = obj.Intersection(ray, l);
7          if (t < close)
8          close = t;
9      }
10     return close;
11 }
```

Листинг 3.5 – Метод затенения пересечения луча взгляда с объектом сцены.

```
1  private void CastShadow(Trace trace)
2  {
3      trace.IsShadowed = Composite.CastShadow(trace,
           light.Position);
4      if (!trace.IsShadowed)
```

```
5       {
6           trace.Color = smoke.CastShadow(trace.Point,
                light.Position, trace.Color);
7           smoke.EnableShadowedParticles(trace.Point,
                light.Position);
8       }
9       else
10      {
11          trace.Color = Color.FromArgb((int)(trace.Color.R * 0.3f),
                (int)(trace.Color.G * 0.3f), (int)(trace.Color.B *
                0.3f));
12      }
13  }
```

Листинг 3.6 – Метод отрисовки частиц дыма.

```
1  private void RenderSmoke(Trace trace)
2  {
3      trace.Color = smoke.Intersection(new(0, 0, 0), trace.Point,
           trace.Color);
4  }
```

3.4 Тестирование

Тестирование выполнено по методологии белого ящика. Полученное изображение 3.1 совпадает с ожидаемым.

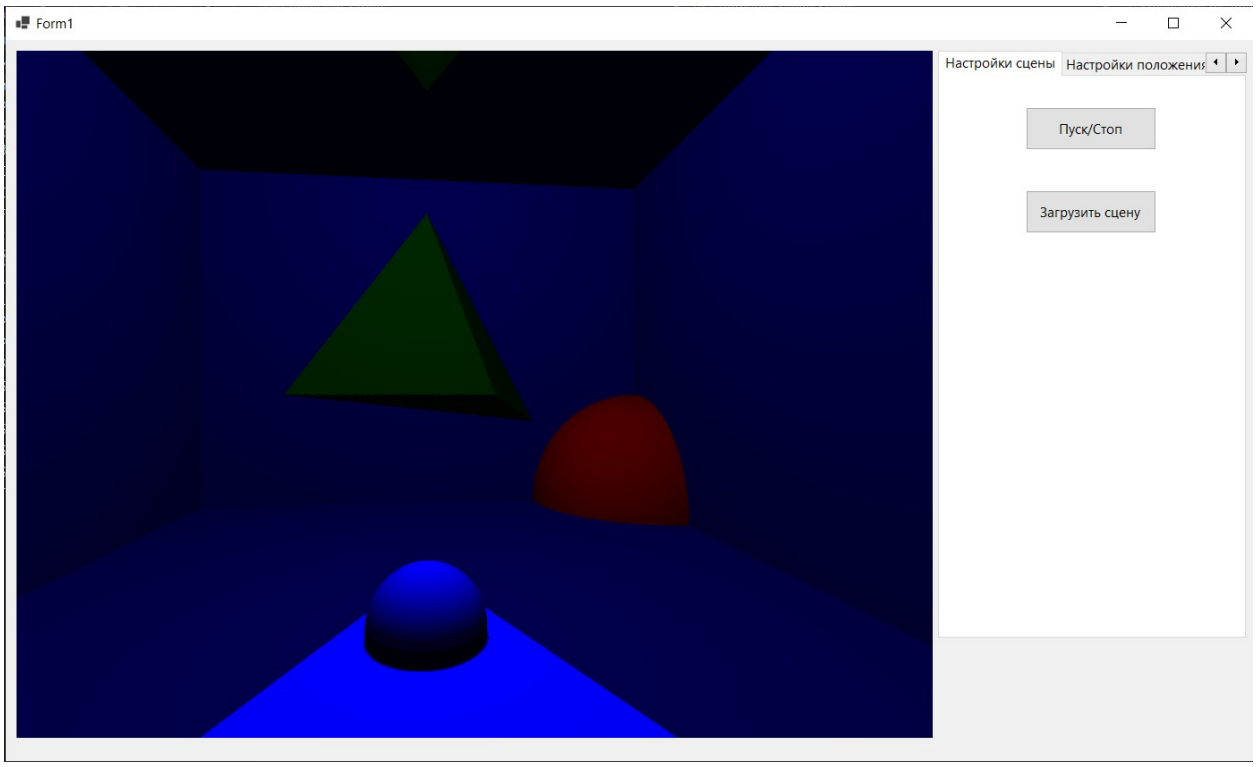

Рисунок 3.1 – Полученное изображение

3.5 Демонстрация работы программы

На рисунках 3.2, 3.3, 3.4 представлен результат работы программы

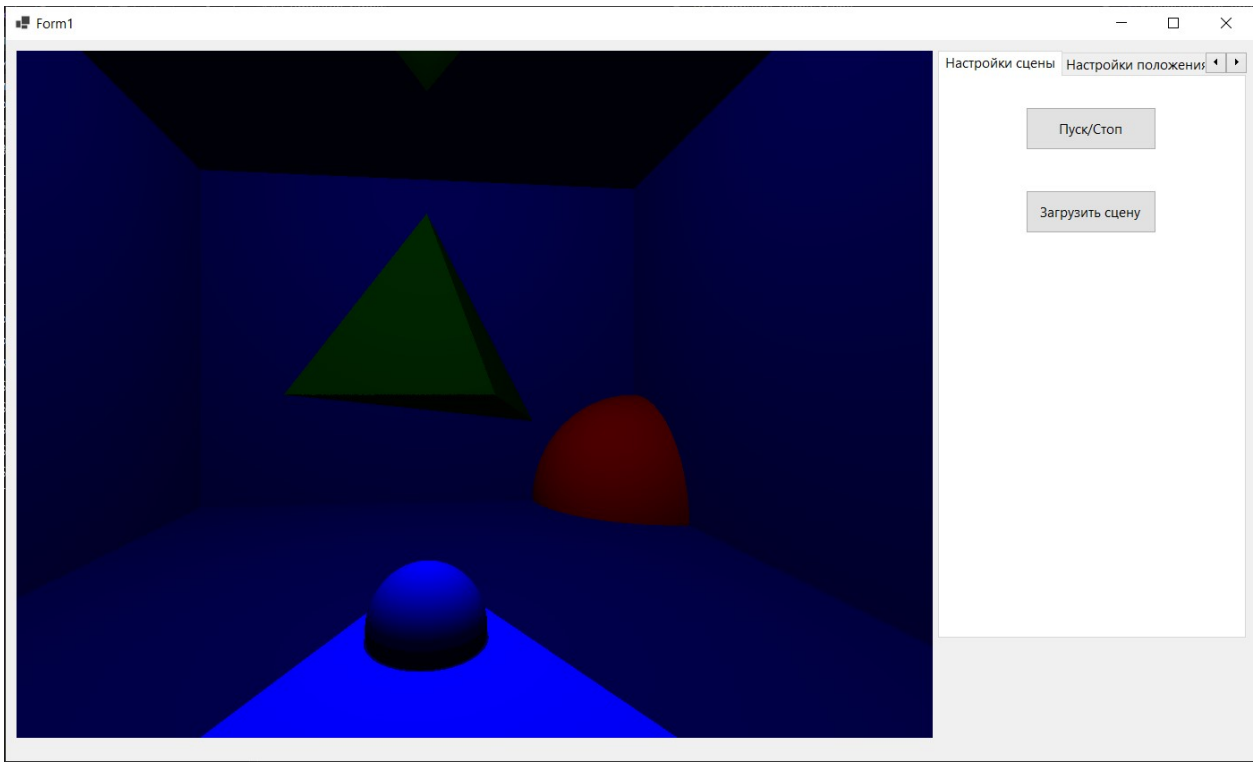

Рисунок 3.2 – Результат работы программы в комнате без дыма

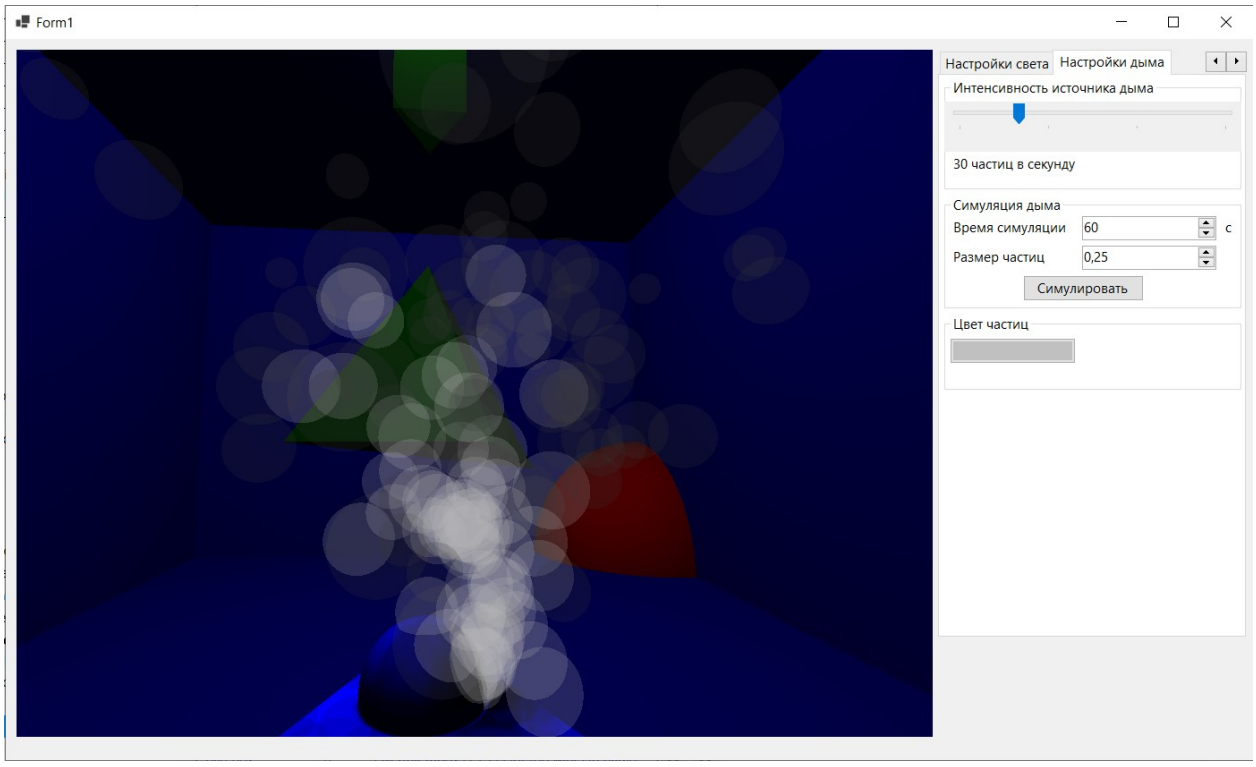

Рисунок 3.3 – Результат работы программы с дымом

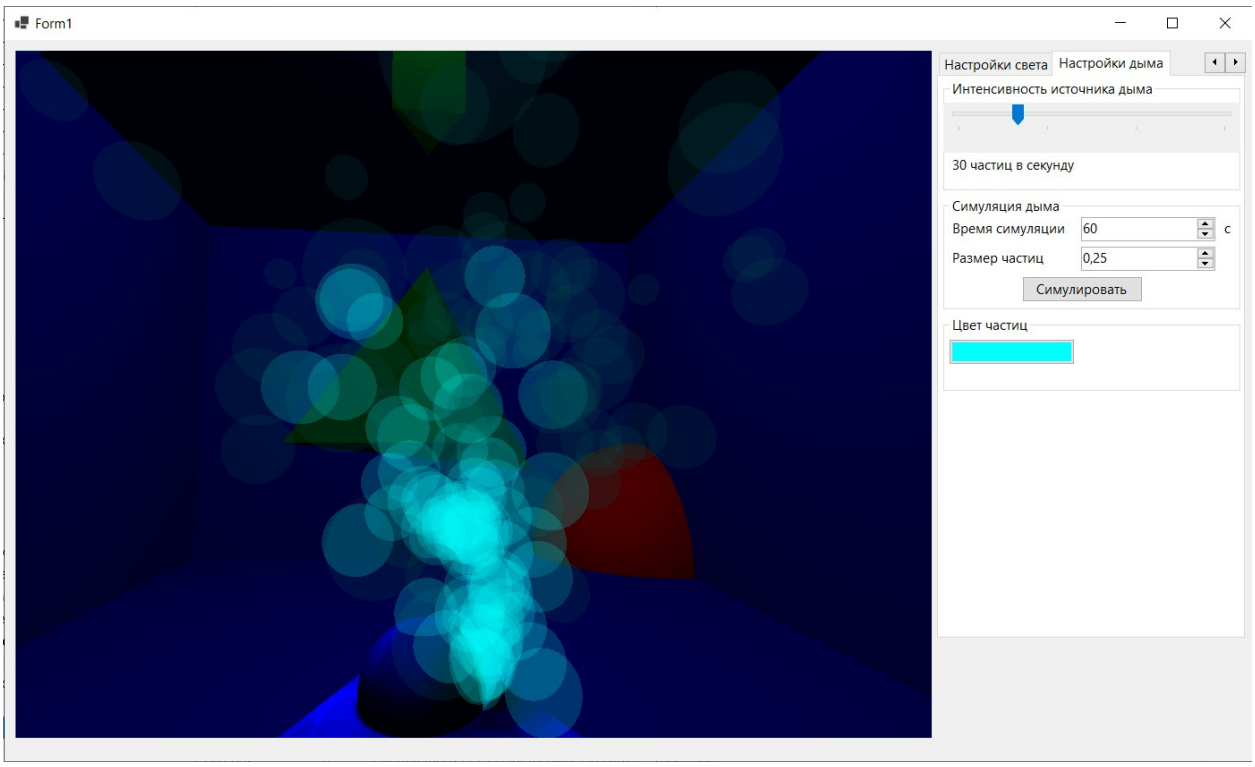

Рисунок 3.4 – Результат работы программы с голубым дымом

Вывод

В данном разделе были рассмотрены средства реализации программного обеспечения и листинги исходных кодов программного обеспечения, разработанные на основе алгоритмов, изложенных в конструкторской части., а также проведено тестирование.

4 Исследовательская часть

В данном разделе будут приведены примеры работы программы, поставлено исследование по сравнению времени синтеза изображения по количеству созданных частиц, а также исследование по сравнению времени синтеза изображения на разном количестве потоков и разном количестве частиц.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось исследование:

— операционная система Windows 10 Корпоративная, Версия 21H1, Сборка ОС 19043.2006;

— память 8 ГБ;

— процессор AMD Ryzen 5 4600H с видеокартой Radeon Graphics 3.00 ГГц [5].

Исследование проводилось на ноутбуке, включенном в сеть электропитания. Во время исследования ноутбук был нагружен только встроенными приложениями окружения, а также непосредственно системой.

4.2 Исследование по сравнению времени синтеза изображения по количеству созданных частиц

Исследование времени синтеза изображения при создании сцен различной нагруженности. Нагрузка будет меняться в зависимости от количества частиц, из которых состоит дым при разном количестве потоков.

Замеры времени синтеза изображения выполнены при помощи класса Stopwatch [6]. Предоставляет набор методов и свойств, которые можно использовать для точного измерения затраченного времени.

Замеры времени для каждого количества частиц проводились 50 раз. В качестве результата взято среднее время синтеза изображения на данном количестве частиц. на данном количестве потоков.

Результаты замеров приведены в таблицах 4.1 и на рис. 4.1.

Таблица 4.1 – Зависимость производительности от количества частиц

Частиц, штук	Время синтеза, мс
1000	3609,8
2000	7054,9
3000	10667,4
4000	14087,2
5000	17562,4
6000	21814,1
7000	25446,5
8000	29256,9
9000	33760,7
10000	37528,8
11000	41513,6
12000	47304,2
13000	50515,8
14000	54719,8
15000	63106,3

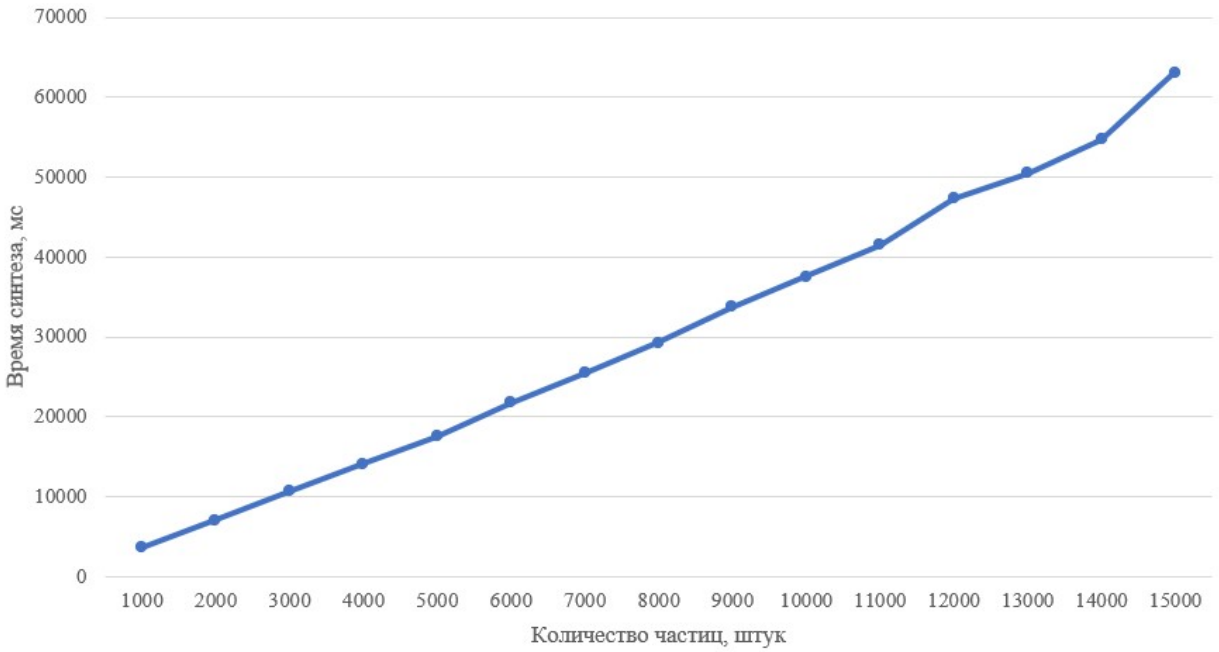

Рисунок 4.1 – Зависимость времени синтеза изображения от количества частиц на сцене

Как видно из результатов, время синтеза изображения увеличивается линейно при линейном увеличении количества частиц в дыме.

4.3 Исследование по сравнению времени синтеза изображения на разном количестве потоков и разном количестве частиц

Результаты замеров приведены в таблице 4.2. На рисунке 4.2 приведена зависимость времени работы реализации алгоритма от количества потоков и количества частиц в дыме.

Таблица 4.2 – Время выполнения реализации алгоритма обратной трассировки лучей при количества потоков и количества частиц в дыме

Количество частиц, штук	Количество потоков, штук				
	6	12	24	48	96
0	1042,34	846,74	948,4	1060,27	1100,81
100	4296,44	3985,65	4652,67	5525,89	8310,33
200	7452,28	6961,98	7774,77	9657,25	15114,46
300	10535,29	9723,64	10917,22	13735,63	21985,89
400	13554,05	12617,40	14056,36	17697,46	28090,16
500	16598,27	15430,81	17155,82	21554,69	34441,75
600	19478,49	18153,36	20216,83	25357,24	40506,90
700	22439,01	20859,34	23142,67	28801,52	46301,25
800	25183,41	23443,41	26192,44	32830,08	51871,79

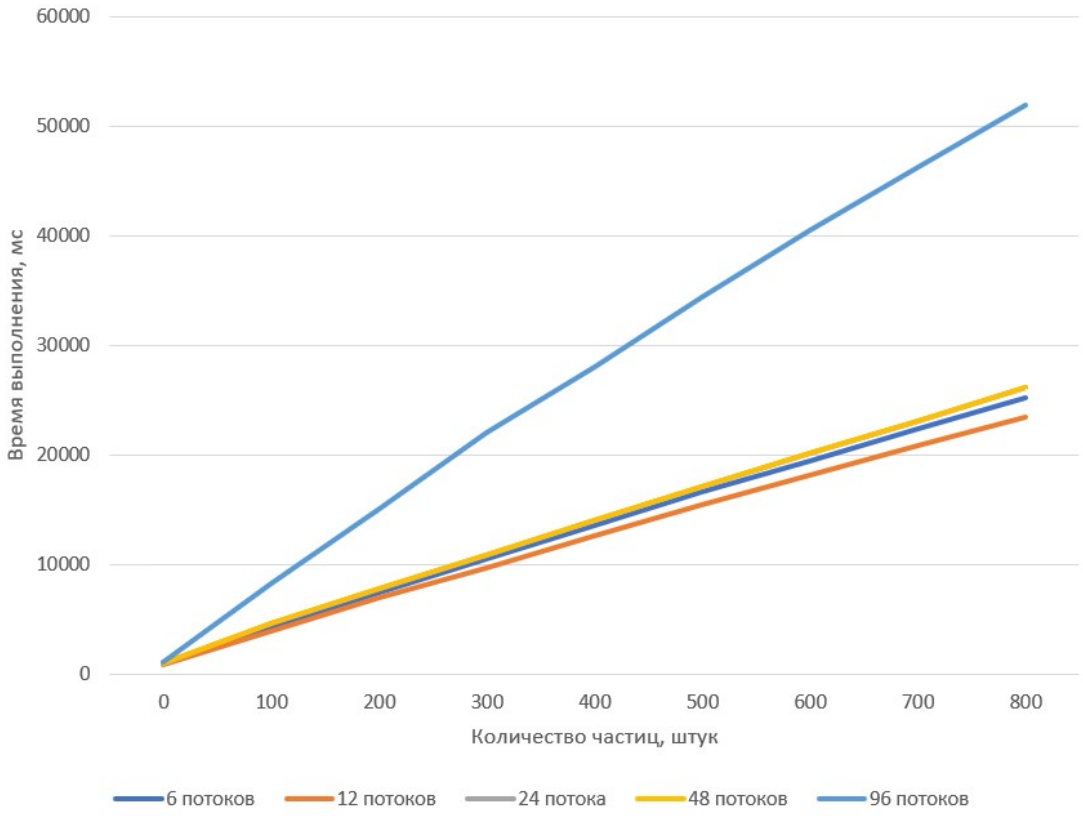

Рисунок 4.2 – Зависимость времени выполнения реализации алгоритма обратной трассировки лучей от количества потоков и количества частиц в дыме

При 12 потоках достигается пик, при котором все логические ядра процессора одновременно выполняют трассировку лучей. Далее при увеличе-

нии числа потоков производительность падает. Это объясняется тем, что создается очередь потоков, которая замедляет работу программы.

Вывод

В данном разделе были рассмотрены примеры работы программного обеспечения, а также выяснено в результате эксперимента, что время синтеза изображения увеличивается линейно при линейном увеличении количества частиц.

Заключение

Поставленная цель была достигнута.

Были решены следующие задачи:

- описана модель трехмерной сцены, в том числе и объекты, из которых она состоит;
- проанализированы существующие алгоритмы построения изображения и выбрать из них те, что в лучше помогут в решении поставленной зада-чи;
- реализованы выбранные алгоритмы;
- разработано программное обеспечение для отображения сцены;
- проведен эксперимент по замеру производительности полученного программного обеспечения.

В процессе исследования было выявлено, что время синтеза изображения увеличивается линейно при линейном увеличении количества частиц. При этом дым визуально выглядит лучше при наибольшем количестве маленьких частиц, которые вместе составляют единую структуру.

Список использованных источников

[1] Visual Simulation of Smoke [Электронный ресурс]. Режим доступа: https://web.stanford.edu/class/cs237d/smoke.pdf (дата обращения: 15.07.2022).

[2] Visualization of smoke using particle system [Электронный ресурс]. Режим доступа: https://www.diva-portal.org/smash/get/diva2:676008/FULLTEXT01.pdf (дата обращения: 15.07.2022).

[3] Курс лекций Курова А. В. по компьютерной графике 2022.

[4] Краткий обзор языка C# [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата обращения: 15.07.2022).

[5] AMD Ryzen™ 5 4600H [Электронный ресурс]. Режим доступа: https://www.amd.com/ru/products/apu/amd-ryzen-5-4600h (дата обращения: 15.07.2022).

[6] Stopwatch Класс [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/api/system.diagnostics.stopwatch?redirectedfrom=MSDN&view=netcore-3.1 (дата обращения: 15.07.2022).