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 «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент ___Оберган Татьяна Максимовна_____
фамилия, имя, отчество

Группа ____ИУ7-45Б_____

Тип практики _____стационарная_____

Название предприятия _____МГТУ им. Н. Э. Баумана, каф. ИУ7_____

Студент _____ ___Оберган Т.М. _____
подпись, дата *фамилия, и.о.*

Руководитель практики _____ ___Куров А.В._____
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2019 г.

Индивидуальное задание:

Разработать программу для трехмерной визуализации городской среды и погодных условий. Выбрать методы построения реалистичных изображений и визуализации погодных условий.

Оглавление

Введение

В современном мире компьютерная графика является неотъемлемой частью человеческой жизни. Она используется повсеместно: для наглядного отображения данных, в компьютерных играх и даже в кино для создания эффектов.

Вследствие этого перед людьми, создающими трехмерные сцены, встает задача создания реалистичных изображений, которые будут учитывать такие оптические явления как преломление, отражение и рассеивание света, а также выбранную текстуру или цвет. Для создания еще более реалистичного изображения учитывается дифракция, интерференция, вторичные отражения света.

Существует множество алгоритмов компьютерной графики, которые решают эту задачу. Зачастую эти алгоритмы ресурсозатратны: чем более качественное изображение мы получаем в итоге, тем больше времени и памяти мы тратим на его синтез. Это становится проблемой при создании динамической сцены, где на каждом временном интервале необходимо производить расчеты заново.

Моей целью во время практики будет выбор или модифицирование существующих алгоритмов и реализация данных алгоритмов для создания трехмерной сцены.

1. Аналитическая часть

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из:

- Источников света – представляют собой материальную точку, из которой исходят лучи света во все стороны (в частном случае, когда источник расположен в бесконечности, имеет направленность).

  Положение задается трехмерными координатами, цвет свечения описывается через RGB параметры.
- Солнце/луна – источник света, который движется по полуокружности и меняет свой цвет в зависимости от угла наклона.
- 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 и длиной и цветом RGB.
- Здания – являются правильной призмой, с основанием, параллельным плоскости земли, и боковыми ребрами, перпендикулярными плоскости земли.

  Здание задается координатами положения центра основания на плоскости земли (x, z), высотой h, количеством боковых граней.
- Крыша здания – часть здания.
  - Плоская (без крыши),
  - Пирамидальная (основание пирамиды совпадает с верхним основанием здания). Пирамидальная крыша задается высотой, положением центра основания (x, y, z), количеством боковых граней.
- Осадки – частички с заданной траекторией движения, цветом.

  Задаются интенсивностью и направлением движения.
  - Траектория движения – прямая. $Ax + By + C = 0$
  - Интенсивность осадков – количество частичек.

- Ветер – задает направление движения осадков. Задается вектором в трехмерном пространстве.

Здания и их крыши наилучшим образом описываются через поверхностные модели. Т.к. каркасные модели не дадут реалистичное изображение, которого мы стараемся достичь, а объемная модель будет излишеством, более затратным по памяти, чем поверхностная.

Поверхностную модель можно задать несколькими способами:

Параметрическим представлением – для получения поверхности нужно вычислять функцию, зависящую от параметра. Так как в сцене нет никаких поверхностей вращения использование параметрического представления будет затруднительно.

Полигональной сеткой – совокупностью вершин, ребер и граней, которые определяют форму объекта (Рис.1).

- Вершинное представление (вершины указывают на другие вершины, с которыми они соединены). Для генерации списка граней для рендеринга нужно обойти все данные, что затрудняет работу с ним.
- Список граней представляет объект как множество граней и вершин.
- Таблица углов (веер треугольников) - хранит вершины в предопределенной таблице. Не подойдет для моей задачи, так как изменение данного представления затратно по времени.

Наиболее удобным способом хранения моей сцены является список граней т.к. данные в нем можно эффективно преобразовывать, представление позволяет явный поиск вершин грани и граней, окружающих вершину.

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.

Чтобы избавиться от данного явления можно добавить предзагрузку кадров анимации движения осадков и показывать их только после полного рендеринга анимации. Для сокращения времени расчетов нужно будет рендерить только часть анимации, а потом повторять ее много раз. Однако при таком подходе нужно будет учесть один фактор: первый кадр созданного блока анимации должен совпадать с последним кадром созданного блока анимации. Иначе будут видны места «склейки» двух блоков.

Также ввиду особенностей сцены (здания на фоне остаются неподвижными во время анимации осадков) имеет смысл рендерить сначала сцену города, а потом добавлять осадки поверх нее.

Алгоритм обратной трассировки лучей

В этом алгоритме за счет скорости работы достигается излишняя, для моей сцены, универсальность. Обратная трассировка позволяет работать с несколькими источниками света, передавать множество разных оптических явлений.

Для создания реалистичного изображения, по правилам обратной трассировки, каждую частицу в осадках (капля дождя, снежинка) нужно будет рассматривать, как отдельный объект сцены, внутри которого могут возникнуть явления дисперсии, преломления и внутреннего отражения. Данный способ взаимодействия с осадками очень ресурсозатратен, хотя позволяет моделировать явления радуги, гало, что не является задачей программы.

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.

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

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. Особенно полезным это может стать при большом количестве домов в сцене.

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

Алгоритм Робертса

Серьезным недостатком является вычислительная трудоемкость алгоритма. В теории она растет как квадрат количества объектов. Поэтому при большом количестве домов в сцене, этот алгоритм будет показывать себя, как недостаточно быстрый. Можно использовать разные оптимизации для повышения эффективности, например сортировку по z.

Преимуществом данного алгоритма является точность вычислений. Она достигается за счет работы в объектном пространстве, в отличии от большинства других алгоритмов.

Некоторые из оптимизаций крайне сложны, что затрудняет реализацию этого алгоритма.

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.

Вывод

Алгоритм метода трассировки лучей не позволит в реальном времени смоделировать туман из-за рассеивания света. Можно лишь его сымитировать используя зависимость затуманенности от пройденного расстояния луча.

Поэтому предпочтительнее использовать Z буфер для динамической сцены визуализации погоды, т.к. важна скорость работы алгоритма. На его основе будет просто визуализировать туман и дождь. В случае тумана можно хранить глубину пикселя, с помощью которой можно будет вычислить насколько «затуманенным» он будет. В случае осадков, можно накладывать осадки на уже посчитанный Z буфер сцены, не проводя повторных расчетов.

1.3 Анализ методов закрашивания

Простая закраска

Вся грань закрашивается одним уровнем интенсивности, который высчитывается по закону Ламберта.

Этот метод крайне прост в реализации и совершенно не требователен к ресурсам. Однако плохо подходит для тел вращения (Рис.2), плохо учитывает отраженный свет.

Для моей задачи этот метод очень хорошо подходит, так как вся работа ведется с гранями зданий, тел вращения нет.

Закраска по Гуро

Основа закраски по Гуро – билинейная интерполяция интенсивностей, за счет которой устраняется дискретность изменения интенсивности и создается

иллюзия гладкой криволинейной поверхности. Хорошо сочетается с диффузным отражением.

Закраска по Фонгу

Основа закраски по Фонгу – билинейная интерполяция векторов нормалей. Достигается лучшая локальная аппроксимация кривизны поверхности. Изображение выходит более реалистичным, зеркальные блики выглядят правдоподобнее, чем в методе закраски по Гуро.

Однако по сравнению с методом Гуро, закраска по Фонгу требует больших вычислительных затрат, так как интерполируются значения векторов нормалей, на основе которых потом вычисляется интенсивность.

Вывод: так как фигуры сцены состоят из плоскостей закраска по Фонгу и Гуро будет скорее мешать: ребра зданий будут сглажены. Тело здания будет хуже восприниматься (Рис.1). Поэтому лучше всего использовать простую закраску.

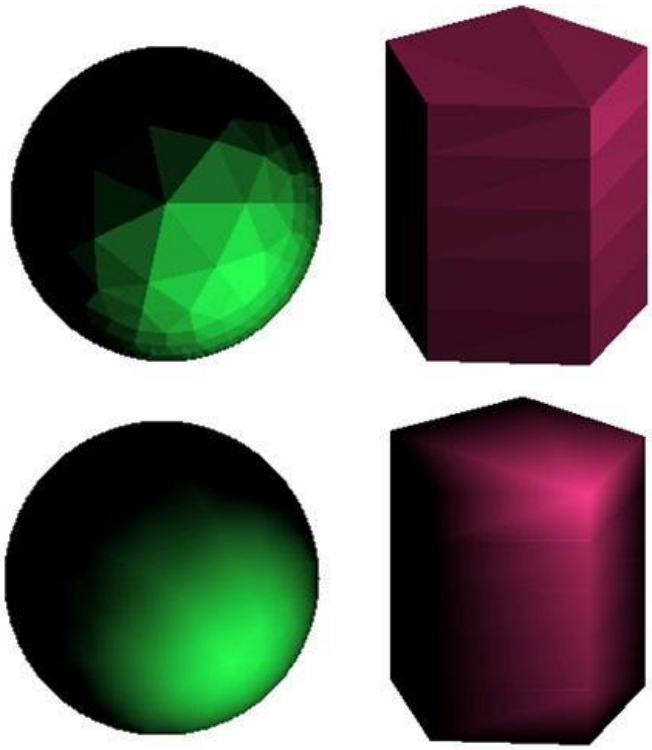

Рисунок 1 Сравнение методов закрашивания

1.4 Анализ алгоритмов построения теней

При трассировке лучей тени получаются без дополнительных вычислений. Пиксел затенен, когда луч попадает на объект и позже не попадает ни в объект, ни в источник света.

Так как в анализе алгоритмов трассировка лучей не была выбрана в качестве алгоритма синтеза сцены, то тени нужно вычислять отдельно.

Один из способов нахождения теней – вычисление проекций тел.

Можно использовать метод теневых карт, в котором предполагается, что освещены только те фрагменты, которые видны из положения источника. Находить видимость можно с помощью алгоритма Робертса, алгоритма Z буфера и других.

Для создания теневых карт будет использоваться алгоритм Z буфера так как этот алгоритм позволит быстро найти видимость объектов сцены.

1.3 Анализ алгоритмов моделирования осадков

Реалистичное и эффективное представление сцен с атмосферными осадками крайне непростая задача из-за огромного количества капель дождя / снежинок. Каждая из этих частиц обладает такими свойствами, как: размер, масса, скорость, ускорение.

Система частиц

В системе частиц частицы (капли дождя, снежинки) рассматриваются как материальные точки с разными атрибутами. Сама же система частиц — это совокупность всех частиц явления. 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.

Метод Кшитиза и Шри

Кшитиза и Шри предложили модель падающей дождевой капли, которая учитывает сложные взаимодействия света, положения наблюдателя. В этом алгоритме капля представлена колеблющейся формой, что делает ее визуально реалистичнее.

Недостатком является недостаточная эффективность метода для создания динамических сцен в реальном времени и непредусмотренность ветра (капля может падать только вниз, но не в сторону). Поэтому для моей сцены такая модель рассмотрения капли не подойдет.

Вывод: система частиц – гибкий способ представления осадков, который хорошо подойдет для моей сцены.

Рисунок 2 Визуализация дождя по методу Кшитиза и Шри

2. Конструкторская часть

2.1 Общий алгоритм решения задачи

1. Задать объекты сцены

2. Задать источники света (учет перемещения солнца) и положение наблюдателя

3. Для каждого полигона высчитать нормаль и интенсивность цвета, найти внутренние пиксели

4. Найти тени

5. Используя алгоритм Z буфера получить изображение сцены, сохранить Z буфер для дальнейших расчетов

6. Если присутствуют осадки, то выполнять пункты 6.1 и 6.2 до тех пор, пока не изменится один из параметров, влияющих на изображение сцены -> переход в 1 (перемещение объектов, вращение камеры, перемещение источников света).

    6.1 Используя систему частиц, наложить осадки на полученное изображение

    6.2 Отобразить изображение

    6.2 Обновить данные о положении частиц

Иначе отобразить изображение.

2.2 Алгоритм Z-буфера

1. Всем элементам буфера кадра присвоить фоновое значение
2. Инициализировать Z буфер минимальными значениями глубины
3. Выполнить растровую развертку каждого многоугольника сцены:
    a. Для каждого пикселя, связанного с многоугольником вычислить его глубину z(x, y)
    b. Сравнить глубину пискселя со значением, хранимым в Z буфере. Если z(x, y) > zбуф(x, y), то zбуф(x,y) =z(x,y), цвет(x, y) = цветПикселя.
4. Отобразить результат

2.3 Простой метод освещения

В простом методе освещения интенсивность рассчитывается по закону Ламберта:

$$I = I_0 * \cos(\alpha), \text{ где}$$

I – результирующая интенсивность света в точке

I_0 – интенсивность источника

α – угол между нормалью к поверхности и вектором направления света

2.4 Генерация осадков

1. Инициализация начальных данных (направления и скорости ветра, интенсивности осадков)
2. Пока не получена команда прекращения осадков:
    2.1 Обновление положения частиц по заданному закону
    2.2 Инициализация новых частиц
    2.3 Отображение частиц на дисплее
3. Пока система частиц не пуста
    3.1 Обновление положения частиц по заданному закону
    3.2 Отображение частиц на дисплее

2.5 Выбор используемых типов и структур данных

- Источник света – задается расположением и направленностью света.
- Объекты сцены – задаются вершинами и гранями.
- Система частиц – хранит в себе частицы, направление движения
- Математические абстракции
    - Точка – хранит координаты x, y, z
    - Вектор – хранит направление по x, y, z
    - Многоугольник – хранит вершины, нормаль, цвет
- Интерфейс – используются библиотечные классы для предоставления доступа к интерфейсу.

3. Технологическая часть

3.1 Выбор и обоснование языка программирования и среды разработки

В качестве языка программирования был выбран C# т.к.:

- Я ознакомилась с этим языком программирования во время занятий по компьютерной графике, что сократит время написания программы
- Данный язык программирования объектно-ориентирован, что даст в полной мере
    - использовать наследование, абстрактные классы и т.д.
    - представлять трехмерные объекты сцены в виде объектов классов, что позволит легко организовать взаимодействие между ними, положительно влияя на читабельность, не снижая эффективности.

В качестве среды разработки была выбрана «Visual Studio 2017» т.к.:

- Она бесплатна в пользовании студентами;
- имеет множество удобств, которые облегчают процесс написания и отладки кода;
- обеспечивает работу с Windows Forms – интерфейсом, который упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обертки для существующего Win32 API в управляемом коде;
- я знакома с данной средой разработки, что сократит время изучения возможностей среды.

3.2 Структура и состав классов

Источник света

```
class LightSource
{
        public Point3D position;
        public Vector direction;
        public Color color;
}
```

Модель

```
class Model
{
        List<Point3D> vertices;
```

```
        public List<Polygon> polygons;
        private Color basicColor;
}
```

Система частиц

```
class ParticleSystem
{
        List<Drop> system;
        Vector direction;
        int xMax, yMax;
        int intensity;
}
class Drop
{
        int x, y, z;
}
```

Математические абстракции

```
class Point3D
{
        public int x, y, z;
}
class Vector
{
        public double x, y, z;
        public double length;
}
class Polygon
{
        List<Point3D> vertices;
        Color basicColor;
        public List<Point3D> pointsInside;
        Vector normal;
}
```

Класс обработки сцены

```
class Zbuffer
{
        private Bitmap res;
        private int[][] Zbuf;
}
```

3.3 Сведения о модулях программы

Program – главная точка входа в приложение;

Form1 – интерфейс;

Light – описание источников света;

Model – Описание объектов сцены;

Rain – Описание дождя;

Zbuffer – Алгоритм Z буфера.

3.4 Интерфейс программы

Группа «Сцена»

Позволяет добавить параллелепипед с заданными размерами и положением на сцену.

Группа «Источник света»

Позволяет выбрать направление источника света.

Группа «Направление ветра»

Позволяет задать направление ветра, который, в свою очередь, задает направление движения и длину капель.

Группа «Дождь»

Позволяет задать интенсивность дождя – количество капель, добавляемых при обновлении кадра.

Задержка (в мс) – задает частоту смены кадров.

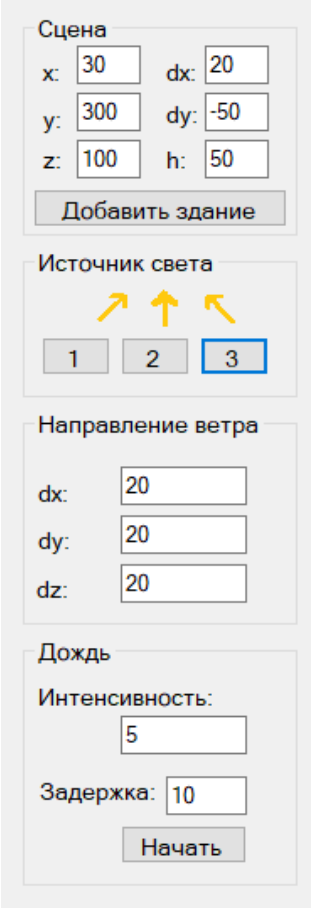

Рисунок 3 Интерфейс программы

Заключение

Во время выполнения поставленной задачи были рассмотрены и проанализированы основные алгоритмы удаления невидимых линий, построения теней, методы закрашивания, методы генерации осадков. Были проанализированы их достоинства и недостатки, выбраны наиболее подходящие для решения поставленной задачи.

Разработанный программный продукт синтезирует трехмерное изображение при помощи алгоритмов компьютерной графики. Программа реализована таким образом, что пользователь может добавлять новые объекты на сцену, изменять характеристики ветра и дождя, изменять положение источника света.

В ходе выполнения поставленной задачи были изучены возможности Windows Forms, получены знания в области компьютерной графики.

Список использованной литературы

1. Методы представления дискретных данных [Электронный ресурс]. – Режим доступа: https://www.graphicon.ru/oldgr/ru/library/multires_rep/index.html (дата обращения 27.06.19)
2. Полигональная сетка [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%B3%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D0%BA%D0%B0 (дата обращения 27.06.19)
3. Е. А. Снижко. Компьютерная геометрия и графика [Текст], 2005. - 17 с.
4. Проблемы трассировки лучей – из будущего в реальное время. [Электронный ресурс]. – Режим доступа: https://nvworld.ru/articles/ray-tracing/3/ (дата обращения 28.06.19)
5. RayTracing – царь света и теней, Лев Дымченко [Электронный ресурс]. – Режим доступа: https://old.computerra.ru/206167/ (дата обращения 28.06.19)
6. Реалистичная визуализация атмосферных осадков в приложениях реального времени, В. В. Карабчевский, А.А. Лунтовская, Донецк ДонНТУ [Текст], 2015
7. K. Garg, S. K. Nayar. Photorealistic rendering of rain streaks. In ACM SIGGRAPH 2006 Papers. SIGGRAPH '06. ACM, New York, NY, 996- 1002 с.

Приложения

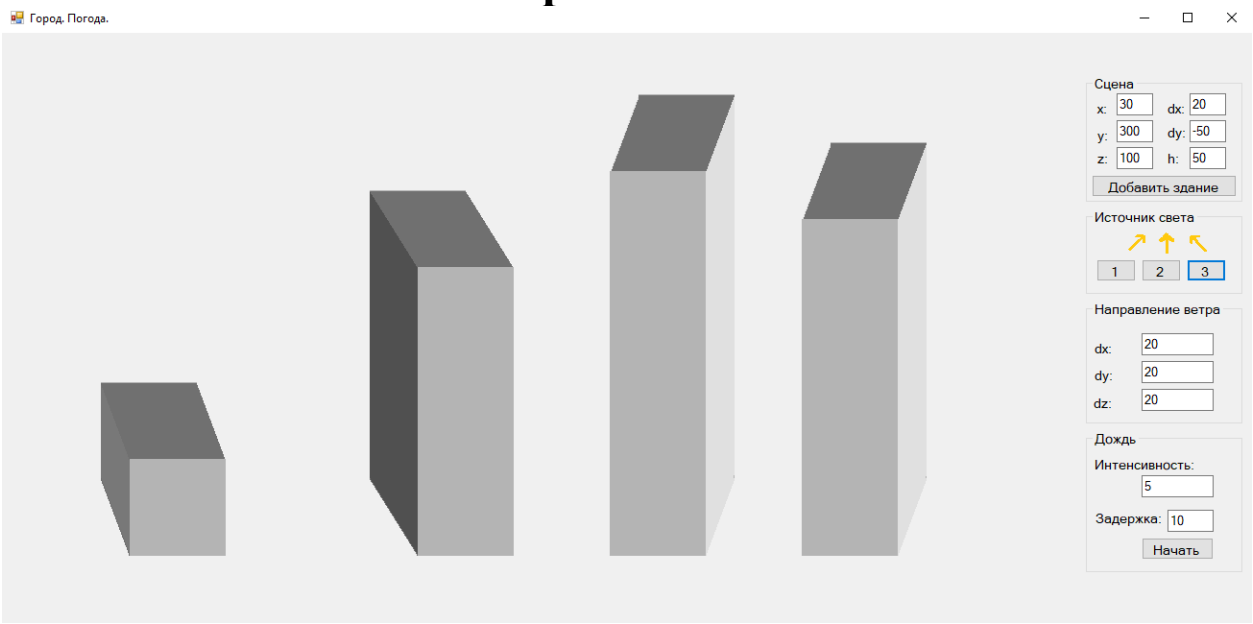

Рисунок 4 Визуализация сцены

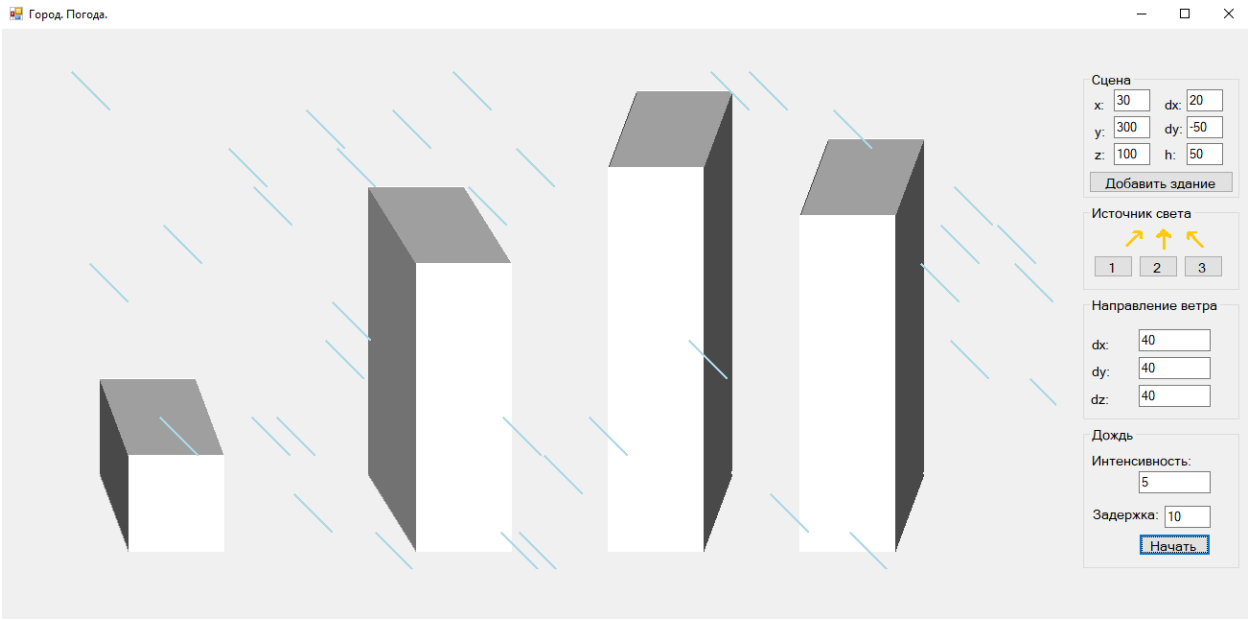

Рисунок 5 Визуализация сцены с дождем