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»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Трехмерная визуализация городской среды и погодных условий»

Студент _____ИУ7-55Б_____ _____ _____Т.М. Оберган_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____Т.Н. Романова_____
(Подпись, дата) (И.О.Фамилия)

2019 г.

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 ____ИУ7-55Б_____

_____Оберган Татьяна Максимовна_____
(Фамилия, имя, отчество)

Тема курсового проекта _____
_____Трехмерная визуализация городской среды и погодных условий_____
_____

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

График выполнения проекта: 25% к _4_ нед., 50% к _7_ нед., 75% к _11 нед., 100% к _14 нед.

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

Оформление курсового проекта:

Расчетно-пояснительная записка на _25-30_ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
На защиту проекта должна быть предоставлена презентация, состоящая из 15-20 слайдов.___
На слайдах должны быть отражены: постановка задачи, использованные методы и_____
алгоритмы, расчетные соотношения, структура комплекса программ, интерфейс,_____
результаты проведенных исследований._____

Дата выдачи задания «27 » ___мая_____ 2019г.

Руководитель курсового проекта _____ ___Т.Н. Романова____
(Подпись, дата) (И.О.Фамилия)

Студент _____ ___Т.М. Оберган____
(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение

В современном мире компьютерная графика является неотъемлемой частью человеческой жизни. Она используется повсеместно: для наглядного отображения данных, в компьютерных играх, в кино для создания эффектов. Вследствие этого перед людьми, создающими трехмерные сцены, встает задача создания реалистичных изображений, которые будут учитывать оптические явления преломления, отражения и рассеивания 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 требуется получить, тем больше времени и памяти тратится на его синтез. Это становится проблемой при создании динамической сцены, где на каждом временном интервале необходимо производить расчеты заново.

Цель данной работы – реализовать построение трехмерной сцены и визуализацию погодных эффектов в городском ландшафте.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1) описать структуру трехмерной сцены, включая объекты, из которых состоит сцена, и дать описание выбранных погодных явлений, которые будут визуализированы;

2) выбор и/или модифицирование 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 погодные эффекты в городском ландшафте.

1. Аналитическая часть

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов.

- Источников света – представляют собой вектор направления света, предполагается, что источник расположен в бесконечности. Цвет свечения описывается через RGB параметры.
- 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.
- Здания – каждое здание является 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 (тогда на здание можно не накладывать крышу),
  - 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.
- Туман – когда включен, загораживает солнце и создает эффект дымки.

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

Поверхностную модель можно задать несколькими способами [1].

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 [2].

- 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.

Рассмотрим ключевые алгоритмы построения трехмерной сцены [8].

1.2.1 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 [3].

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.

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 моей 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. 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.

1.2.3 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.

1.2.4 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

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, 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. Поэтому лучше всего использовать простую закраску.

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.

1.5 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 [6]. 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 [7]. 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 (см. рис. 2). Поэтому для моей сцены такая модель рассмотрения капли не подойдет.

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

1.6 Описание трехмерных преобразований сцены

Сдвиг точки:

$$\begin{cases} X = x + dx \\ Y = y + dy \\ Z = z + dz \end{cases}$$

Масштабирование относительно начала координат:

$$\begin{cases} X = x \cdot k_x \\ Y = y \cdot k_y \\ Z = z \cdot k_z \end{cases}$$

Поворот относительно осей x, y, z на угол φ:

$$\begin{cases} X = x \\ Y = y \cdot \cos\phi + z \cdot \sin\phi \\ Z = -y \cdot \sin\phi + z \cdot \cos\phi \end{cases} \quad \begin{cases} X = x \cdot \cos\phi - z \cdot \sin\phi \\ Y = y \\ Z = x\sin\phi + z\cos\phi \end{cases} \quad \begin{cases} X = x \cdot \cos\phi + y \cdot \sin\phi \\ Y = -x \cdot \sin\phi + y \cdot \cos\phi \\ Z = z \end{cases}$$

1.7 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, методом закраски – простой, построение теней будет выполняться с помощью теневых карт, построенных алгоритмом Zбуфера, для моделирования осадков была выбрана система частиц.

2. Конструкторская часть

В данном разделе будут рассмотрены требования к программе и алгоритмы визуализации сцены и погодных явлений.

2.1 Требования к программе

Программа должна предоставлять следующие возможности:

- визуальное отображение сцены;
- добавление нового объекта в сцену;
- активация режима тумана;
- активация режима дождя;
- поворот исходной сцены.

2.2 Общий алгоритм визуализации трехмерной сцены

Визуализации трехмерной сцены городского ландшафта с осадками происходит поэтапно. Рассмотрим алгоритм визуализации.

1. Задать объекты сцены
2. Задать источники света и положение наблюдателя
3. Для каждого полигона высчитать нормаль и интенсивность цвета, найти внутренние пиксели
4. Используя алгоритм Z буфера получить изображение сцены, и буфер глубины
5. Используя алгоритм Z буфера получить буфер глубины для источника света
6. Найти затененные участки при помощи наложения двух буферов
7. Если присутствует туман, то наложить туман на изображение используя Z буфер изображения
8. 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).

    8.1 Используя систему частиц, наложить осадки на полученное изображение

    8.2 Отобразить изображение

    8.2 Обновить данные о положении частиц

Иначе отобразить изображение.

2.3 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

2.4 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

2.5 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

2.6 Явление тумана

Для того, чтобы создать эффект дымки или плотного тумана нужно знать глубину (отдаленность от наблюдателя) видимого пикселя. Используя это значение можно вычислить интенсивность тумана для этого пикселя. При z >= zдальнее интенсивность тумана будет равна 1, иначе

$$k = (z_{пикс} - z_{дальнее}) / (z_{наблюдателя} - z_{дальнее}),$$

где k – интенсивность пикселя, 1-k – интенсивность тумана.

Расчет глубины пикселя не требуется производить, при условии, что сохранен Z-буфер сцены, при удалении невидимых линий.

2.6 Выбор используемых типов и структур данных

Для разрабатываемого ПО нужно будет реализовать следующие типы и структуры данных.

- Источник света – направленностью света.
- Сцена – задается объектами сцены
- 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 кода.

В качестве среды разработки была выбрана «Visual Studio 2017» по следующим причинам:

- она бесплатна в пользовании студентами;
- она имеет множество удобств, которые облегчают процесс написания и отладки кода;
- она 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

В этом разделе будут рассмотрена структура и состав классов, см. рис. 3.1, рис. 3.2, рис 3.3.

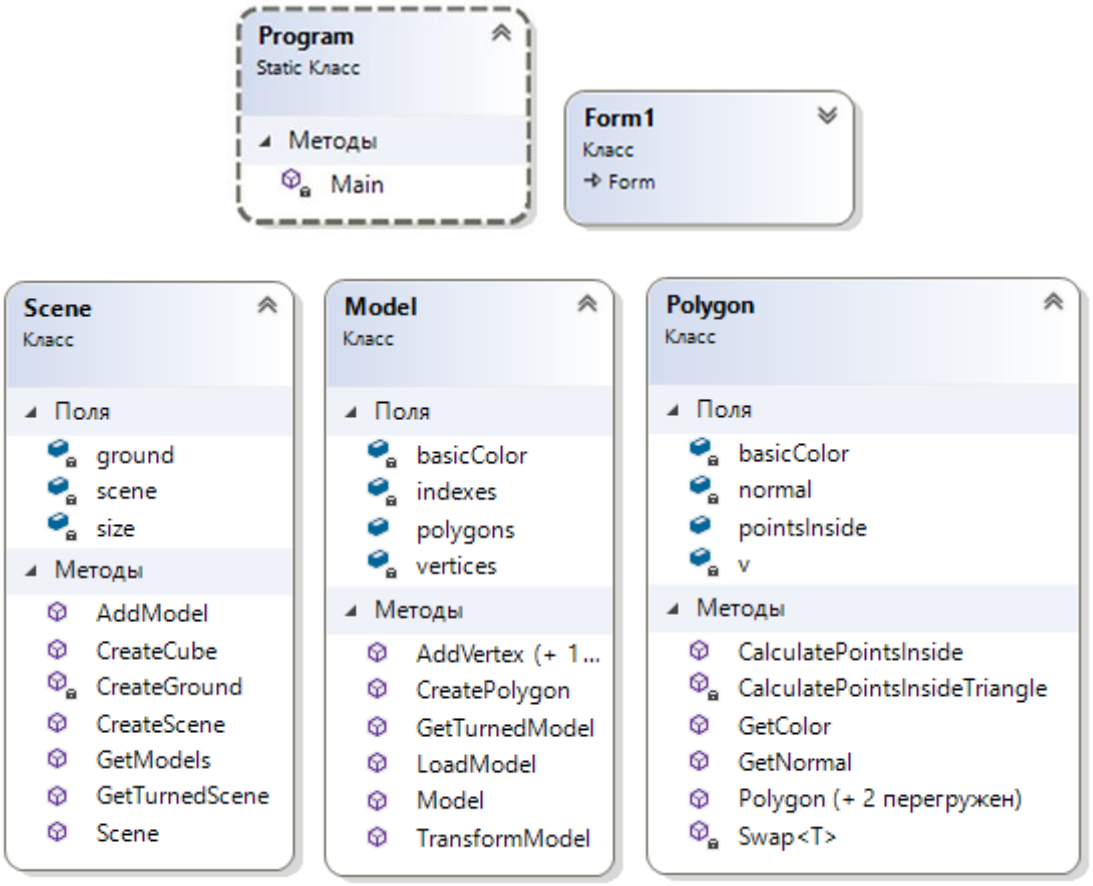

Рис. 3.1 структура классов Program, Form1, Scene, Model, Polygon

Program – входная точка в программу

Form1 – класс пользовательского интерфейса

Scene – хранит информацию о сцене: размеры, модели внутри сцены, имеет методы создания сцены, ее преобразования;

Model – хранит информацию о модели: ее вершины, индексы вершин для создания многоугольников, многоугольники, имеет методы загрузки и преобразования модели;

Polygon – класс многоугольника, хранит цвет, вершины, нормали и точки внутри, при необходимости, имеет методы вычисления нормали и получения точек внутри многоугольника;

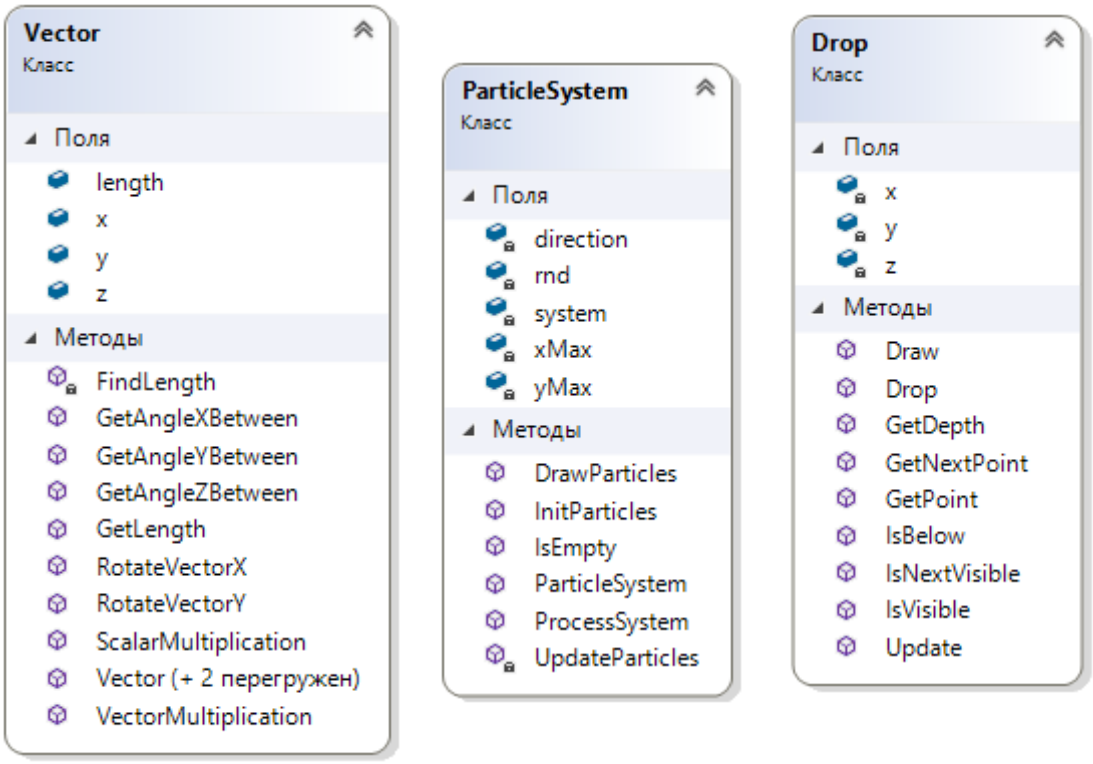

Рис. 3.2 структура классов Vector, ParticleSystem, Drop

Vector – хранит направление вектора и его длину, имеет методы поиска длины вектора, нахождение углов между двумя векторами, векторное и скалярное умножение векторов;

ParticleSystem – хранит информацию о системе частиц: направление движения, частицы и методы: инициализация, обновление и отрисовка системы частиц;

Drop – хранит текущее положение частицы, методы отрисовки, обновления и проверки различных условий частицы;

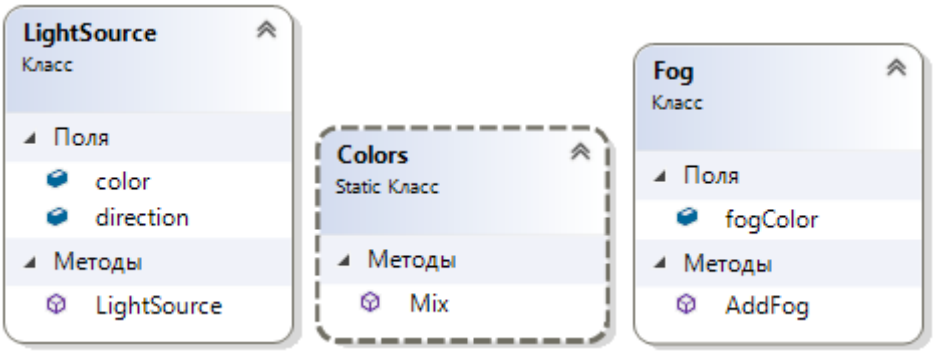

Рис. 3.3 структура классов LightSource, Colors, Fog

LightSource – хранит в себе цвет света и направление света;

Colors – статический класс имеющий метод смешивания цветов;

Fog – класс для добавления тумана на сцену.

3.3 Сведения о модулях программы

Program.cs – главная точка входа в приложение;

Form1.cs – интерфейс;

Light.cs – описание источников света;

Scene.cs – описание сцены, методы взаимодействия с ней;

Model.cs – описание объектов сцены, методы взаимодействия с моделью и ее частями;

Rain.cs – описание генерации дождя и его наложения на изображение;

Fog.cs – описание тумана, методы его наложения на изображение;

Zbuffer.cs – алгоритм Z буфера;

Colors.cs – взаимодействие цветов;

Transformation.cs – функции преобразования координат;

3.4 Интерфейс программы

Интерфейс будет включать следующие группы элементов интерфейса(рис. 3.4).

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.

Группа «Туман»: позволяет задать параметры тумана и наложить туман на текущую сцену.

Группа «Поворот»: позволяет выполнить операции поворота сцены в нужных направлениях.

Группа «Режим»: позволяет посмотреть на текущую сцену в разных режимах: обычный режим – посмотреть визуализацию сцены,

корректность работы z буфер, тени – визуализация с цены с наложением теневой карты, солнце – визуализация сцены от лица источника света.

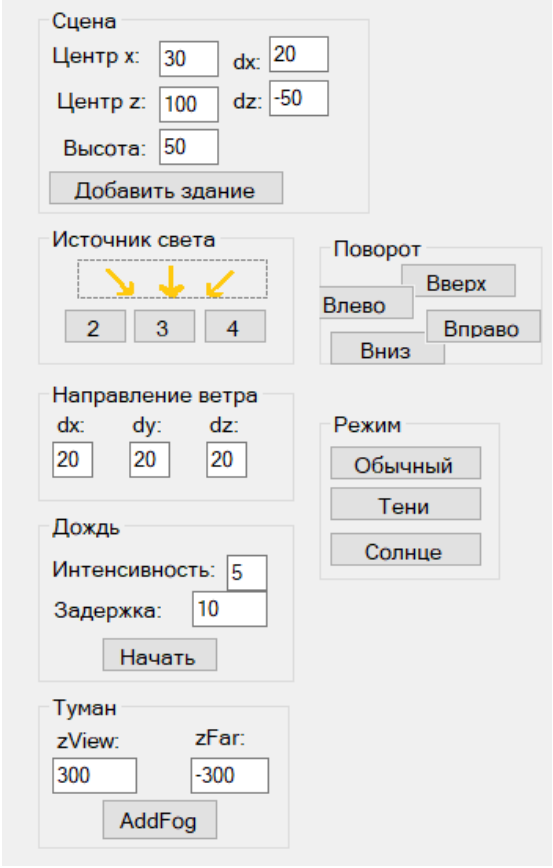

Рис. 3.4 Интерфейс программы – группы настроек

4. Экспериментальная часть

В данном разделе будет проведена апробация реализованной программы для проверки корректности ее работы и поставлен эксперимент по оценки эффективности работы программы.

4.1 Цель эксперимента

Целью эксперимента является проверка правильности выполнения поставленной задачи, оценка эффективности при многопоточной реализации просчета теней.

4.2 Апробация

На рисунках 4.1 и 4.2 показан один и тот же городской ландшафт в разных режимах освещения: обычный и с тенями. Можно заметить, что тени отображаются корректно.

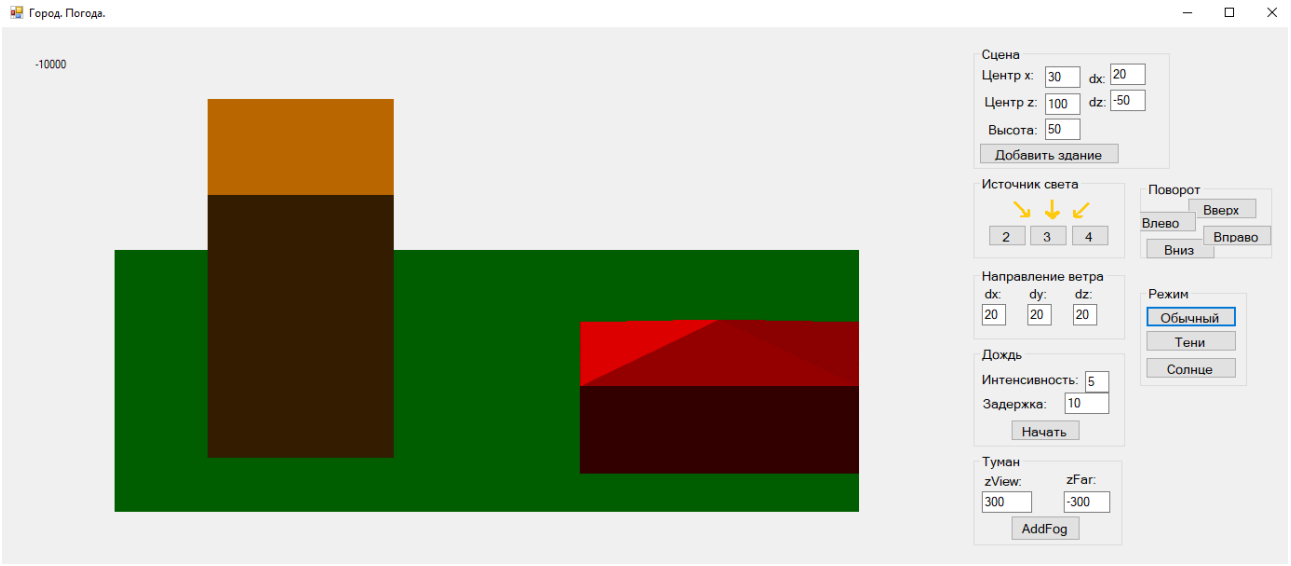

Рис. 4.1 Визуализация сцены в обычном режиме

На рисунках 4.2 и 4.3 показан один и тот же городской ландшафт при разном положении источника. Смена положения источника освещения работает корректно. При смене источника возникает дефект теней на крыше правого дома. Это может быть объяснено при помощи рисунка 4.4: из источника света видна только часть пикселей крыши правого дома, поэтому остальная часть считается невидимой и при наложении теневой карты в этих местах появится тень.

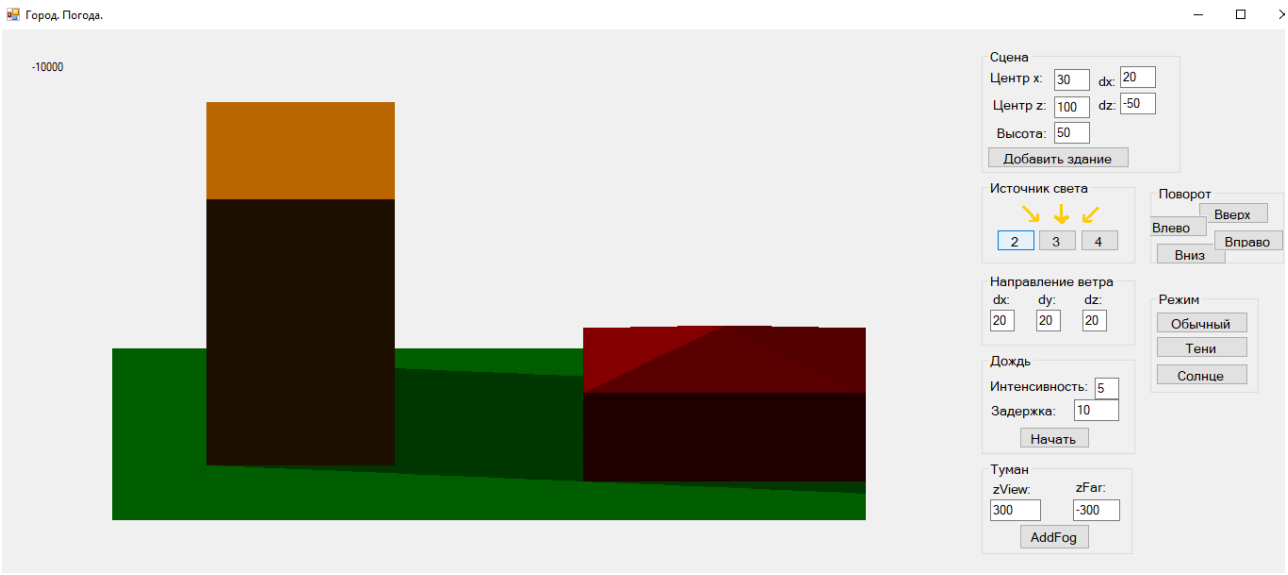

Рис. 4.2 Визуализация сцены в режиме теней

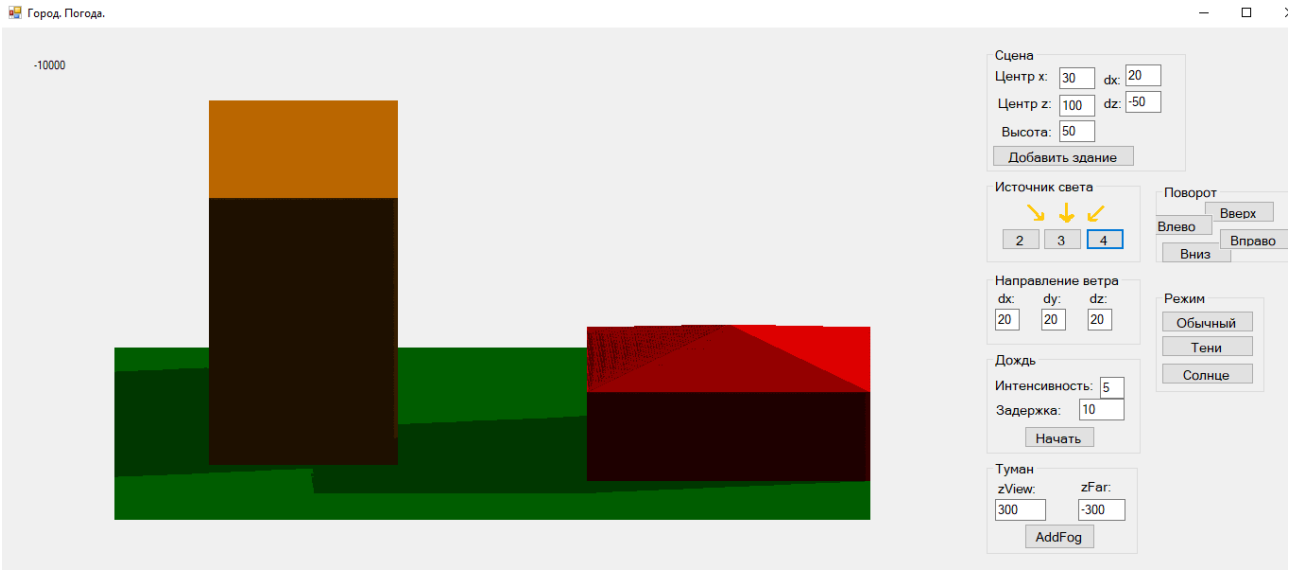

Рис. 4.3 Смена направленности источника освещения

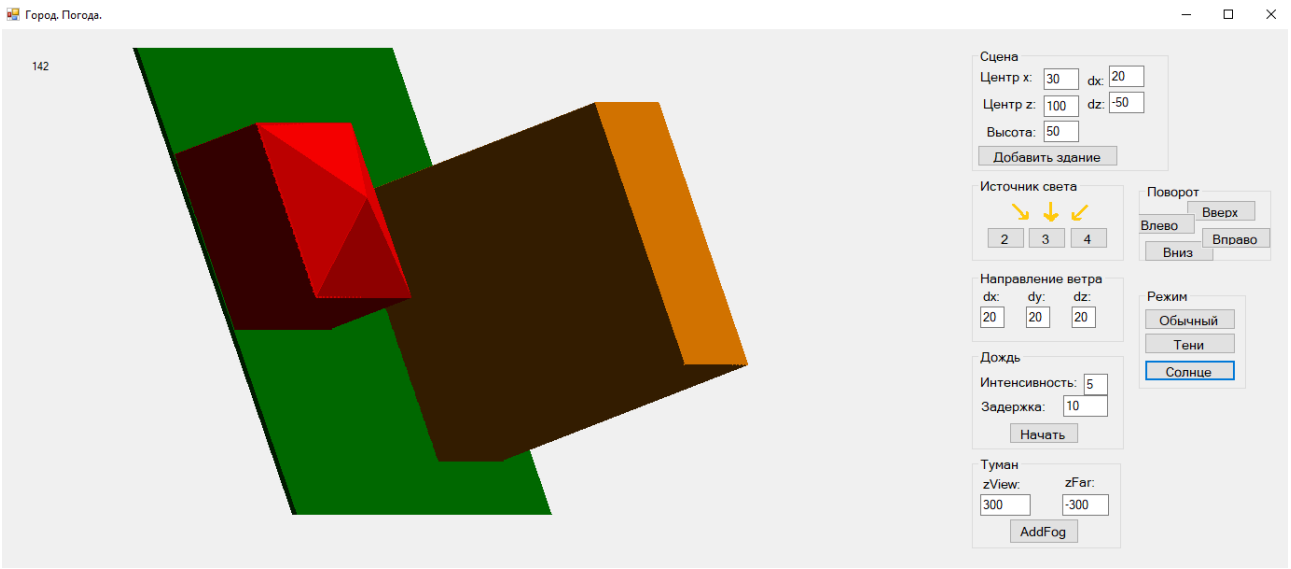

Рис. 4.4 Смена режима отображения на «Солнце»

На рисунке 4.5 показан эффект дождя. Из рисунка видно, что эффект дождя работает корректно.

На рисунке 4.6 показан эффект тумана, на котором видно, что туман реализован правильно: дальние объекты сцены менее видимы, скрываются за дымкой; теней у объектов сцены нет, а значит источник света скрыт за туманом, как и было заявлено.

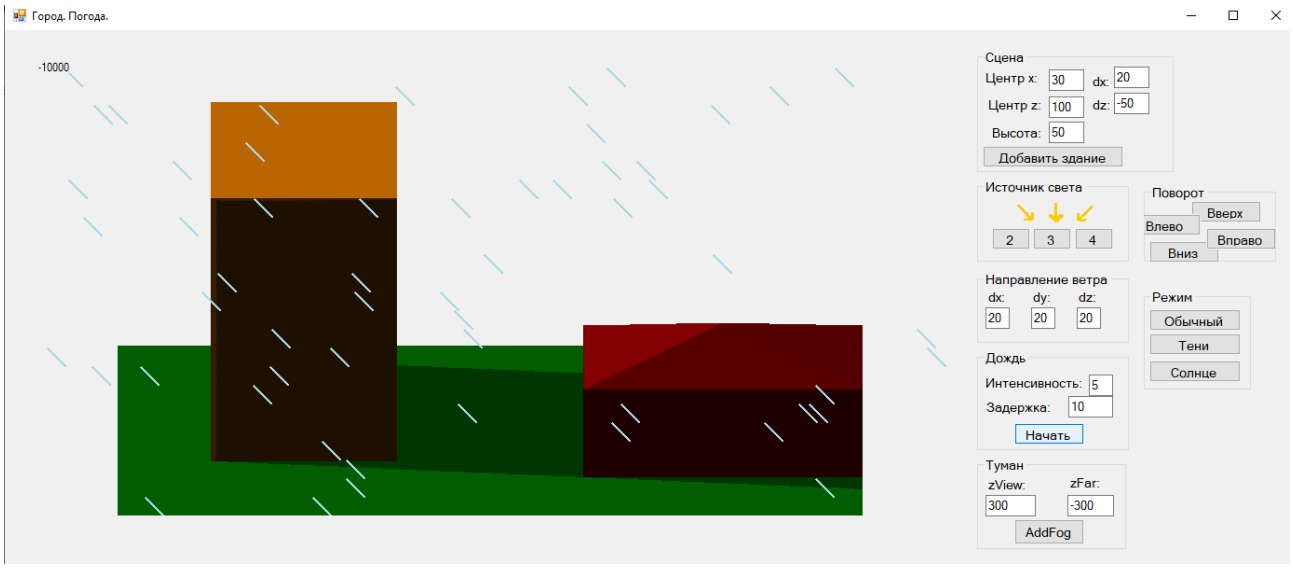

Рис. 4.5 Эффект дождя

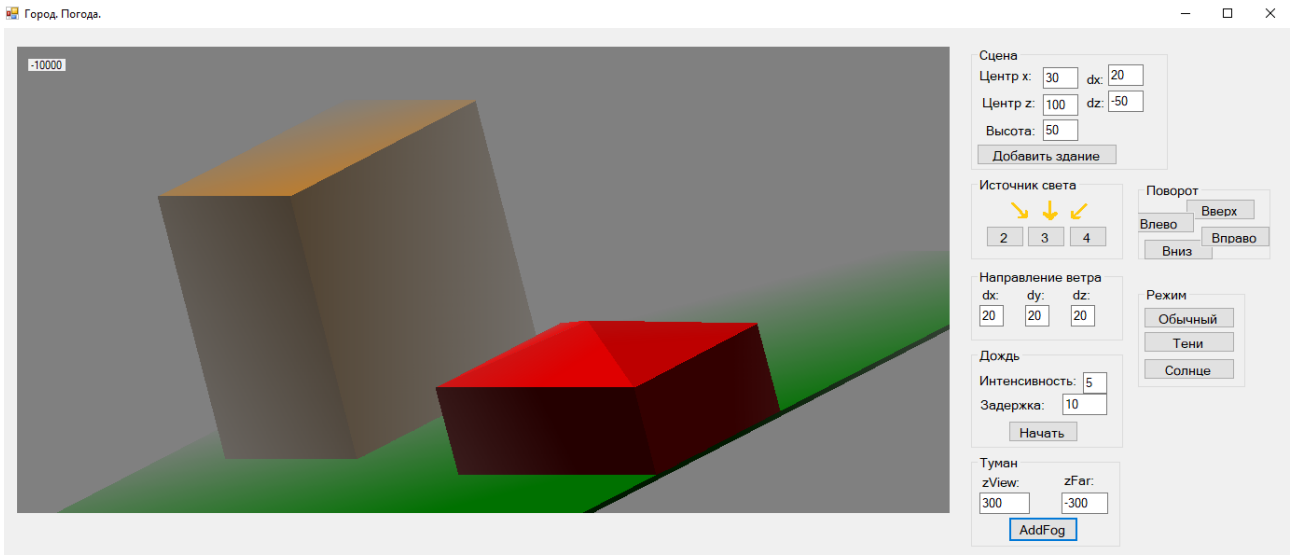

Рис. 4.6 Эффект тумана

4.3 Описание эксперимента

Была реализована функция параллельного нахождения теней. Для этого была использована библиотека System.Threading, функция Parallel.For. Эта функция производит итерации цикла параллельно.

Эксперимент проводился на компьютере со следующими характеристиками:

- Intel® Core™ i5-8300H
- 4 ядра
- 8 логических процессоров
- 12гб оперативной памяти

В данном эксперименте, полученное изображение разбивалось на массивы рядов, во избежание проблем с разделяемой памятью (класс Bitmap не может использоваться для чтения параллельно), а по окончанию эксперимента собиралось обратно.

Данный эксперимент показал ухудшение производительности при внедрении параллельного подсчета итераций цикла:

- 1666536 тиков – обычное нахождение теней;
- 4169953 тиков – параллельное нахождение теней, с разбиением изображения;
- 7452667 тиков – параллельное нахождение теней.

Из предоставленных данных видно, что параллельное нахождение теней, даже с оптимизацией разбиения, работает медленнее в 2.5 раза чем обычное.

Заключение

Во время выполнения курсового проекты была описана структура трехмерной сцены, были рассмотрены основные алгоритмы удаления невидимых линий, построения теней, методы закрашивания, методы генерации осадков. Были проанализированы их достоинства и недостатки, выбраны и реализованы наиболее подходящие для решения поставленной задачи. Было разработано программное обеспечение для визуализации сцены и погодных эффектов.

Программа реализована таким образом, что пользователь может добавлять новые объекты на сцену, изменять характеристики ветра и дождя, добавлять туман, изменять положение источника света.

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

В ходе выполнения экспериментальной части, было установлено, что параллельное нахождение теней с помощью функции Parallel.For разбиения цикла, даже с оптимизацией разбиения, работает медленнее в 2.5 раза чем обычное.

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. Bruce Baumgart, Winged-Edge Polyhedron Representation for Computer Vision. National Computer Conference, May 1975
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.
8. D. F. Rogers. Procedural Elements for Computer Graphics. 2nd ed., 1998 – p.457-517