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 в разных погодных условиях»

Студент _____ИУ7И-54Б_____ _____ _____Динь Вьет Ань_____
(Группа) (Подпись, дата) (И.О.Фамилия)

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

2022 г.

Содержание

Введение

В современном мире компьютерная графика является важной в человеческой жизни. Она используется повсеместно: в компьютерных играх, в кино для создания эффектов. Поэтому содателям трехмерных сцен необходимо решить задачу создания реалистичных изображений, учитывающих преломление, отражение и оптическое рассеяние света, а такде выбранный цвет. Для создания более реалистичного изображения учитываются дифракция, интерференция, вторичные отражения света.

Существует множество алгоритмов компьютерной графики, решающих эту задачу. Обычно эти алгоритмы ресурсозатратны: чем более качественное изображение необходимо получить, тем больше времени и памяти требуется для процесса синтеза. 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 структуры трехмерной сцены и объектов, 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:

- Солнца – источник света, который представляет собой материальную точку, предполагается, что солнце находится в бесконечности. Положение солнца задается трехмерными координатами, а цвет свечения описывается RGB параметрами.
- Плоскости земли – ограничивающая плоскость, предполагается, что под плоскостью земли не расположено объектов. Изначально расположена внизу экрана, параллельна Oxz. Размеры задаются шириной, длиной и цветом RGB.
- Зданий – каждое здание является прямоугольным параллелепипедом с основанием, параллельным плоскости земли, и боковыми ребрами, перпендикулярными плоскости земли. Здание задается координатами положения центра основания на плоскости земли (x, z), высотой h и видом крыши. Крыша здания – часть здания.
    - Плоская.
    - Пирамидальная крыша - задается высотой и основанием, которое совпадает с верхним основанием здания.
- Эффекта тумана – когда включен, загораживает солнце и создает эффект дымки.

Здания и их крыши лучше всего описываются поверхностными моделями. Потому что каркасные модели не дадут реалистичное изображение, а объемные модели потребуют больше памяти, чем поверхностные модели.

Поверхностную модель можно задать несколькими способами:

- **Параметрическим представлением** – для получения поверхности, нужно вычислить функцию, которая зависит от параметра. Т. к. в сцене нет поверхности вращения, использование параметрического представления будет затруднено [1].

- **Полигональной сеткой** – набор 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 набор граней и вершин.

Удобнее всего хранить мою сцену со сииском граней, т. к. данные в нем могут быть эффективно преобразованы, представление позволяет явный поиск вершин грани и граней, окружающих вершину.

1.2 Анализ алгоритмов удаления невидимых линий и поверхностей

Рассмотрим основные алгоритмы для построения трехмерных сцен [5]:

1.2.1 Алгоритм обратной трассировки лучей

В этом алгоритме для моей сцены за счет скорости работы достигается излишняя универсальность. Обратная трассировка позволяет работать с несколькими источниками света, передавать множество разных оптических явлений [3].

Преимуществом этого алгоритма является возможность его использования в параллельных вычислительных системах (т.к. расчет отдельной точки выполняется независимо от других).

Серьезным недостатком этого алгоритма является большой объем необходимых вычислений для синтеза изображения моей сцены. Алгоритм не подойдет для генерации динамических сцен и моделирования диффузного отражения [3].

1.2.2 Алгоритм, использующий Z буфер

Несомненным достоинством этого алгоритма является 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 [4]. Это может быть особенно полезным при большом количестве зданий в сцене.

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

1.2.3 Алгоритм Робертса

Серьезным недостатком является вычислительная сложность алгоритма. Теоретически она увеличивается пропорционально квадрату количества объектов. Поэтому при большом количестве зданий в сцене этот алгоритм покажет себя недостаточно быстрым. 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 оптимизации 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 и использование этого алгоритма будет излишним. Можно лишь его сымитировать используя зависимость туман от пройденного расстояния луча. Поэтому лушче использовать Z буфер для динамической сцены визуализации погоды, т.к. важна скорость работы алгоритма. На его основе будет легко визуализировать

туман. В случае тумана можно хранить глубину пикселя, с помощью которой можно будет вычислить насколько «затуманенным».

1.3 Анализ методов закрашивания

Простая закраска

Вся грань закрашивается с 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. Но для моей сцены 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 интенсивности, за счет которой устраняется дискретность изменения интенсивности и создается иллюзия гладкой криволинейной поверхности. Хорошо сочетается с диффузным отражением [3].

Закраска по Фонгу

Основа закраски по Фонгу – билинейная интерполяция векторов нормалей. Достигается лучшая локальная аппроксимация кривизны поверхности. Изображение кажется более реалистичным, зеркальные блики выглядят правдоподобнее, чем в методе закраски по Гуро [3].

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 и уже был выбран для удаления невидимых линий и поверхностей.

1.5 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:

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

1.6 Выводы из аналитического раздела

В данном разделе были рассмотрены алгоритмы удаления невидимых линий и поверхностей, методы закрашивания поверхностей, алгоритмы построения теней. В качестве алгоритма удаления невидимых линий был выбран Zбуфер, методом закраски – простой, построение теней будет выполняться с помощью теневых карт, построенных алгоритмом Zбуфера.

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.
- Поворот исходной сцены.
- Изменение положения солнца.

2.2 Общий алгоритм визуализации трехмерной сцены

Визуализации трехмерной сцены городского ландшафта происходит поэтапно. Рассмотрим алгоритм визуализации:

1. Задать объекты сцены.

2. Задать источники света и положение наблюдателя.

3. Для каждой грани высчитать нормаль и интенсивность цвета потом 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 с помощью Z буфера изображения.

8. 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 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 [6].

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 программного обеспечения нужно будет реализовать следующие типы и структуры данных.

- Источник света – направленностью света.
- Сцена – задается объектами сцены.
- Объекты сцены – задаются вершинами и гранями.
- 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 ознакомился с этим языком программирования во время занятий по компьютерной графике;
- 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, не снижая эффективности.

В качестве среды разработки был выбран «Visual Studio 2022» т.к. :

- 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 знаком с данной средой разработки, что сократит время изучения возможностей среды.

3.2 Структура и состав классов

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

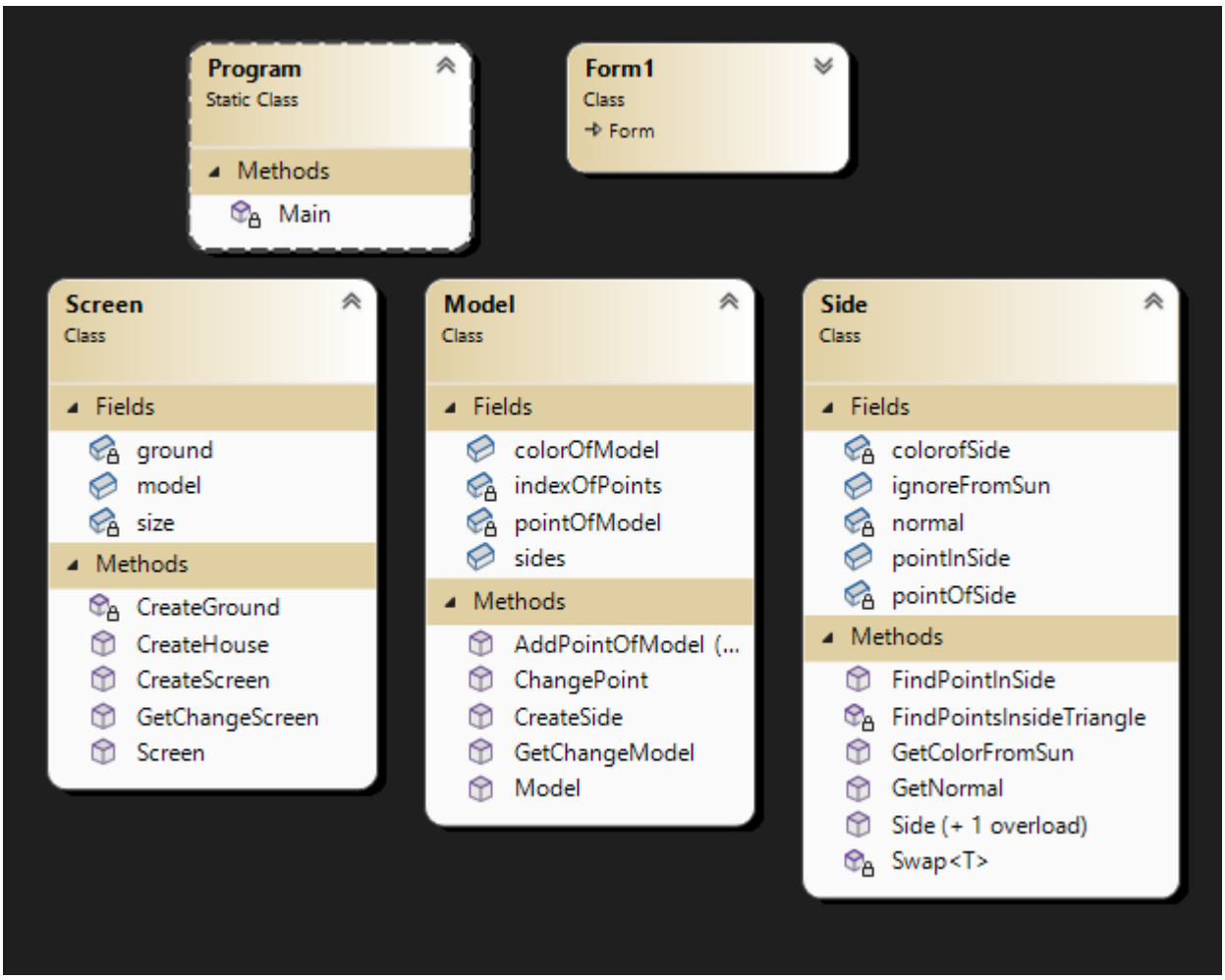


Рис. 3.1 Структура классов Program, Form1, Screen, Model, Side

Program – входная точка в программу.

Form1 – класс интерфейса.

Screen – хранит информацию о сцене: размеры, модели внутри сцены, имеет методы создания сцены, поверхности земли, ее преобразования.

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

Side – класс граней, хранит цвет, вершины, нормали и точки внутри, имеет методы вычисления нормали и получения точек внутри многоугольника.

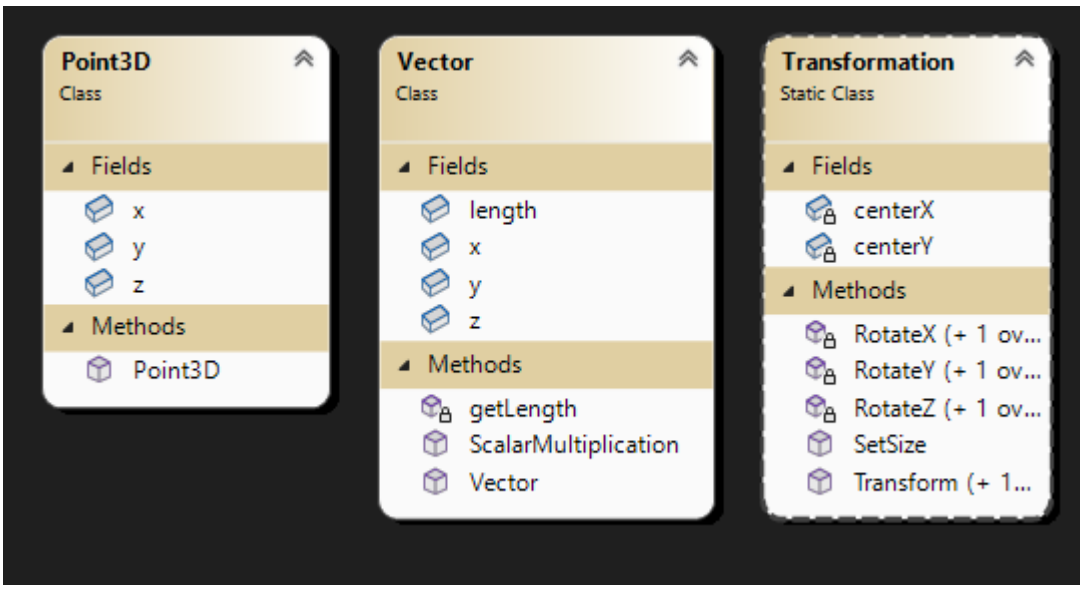


Рис. 3.2 Структура классов Vector, Point3D, Transformation

Vector – хранит направление вектора и его длину, имеет методы поиска длины вектора и скалярного умножения векторов.

Point3D – хранит информацию о точке - координаты токи, имеет метод создания точки.

Transformation – хранит значения координат центра сцены и имеет методы для поиска новых координат точек после поворота сцены.

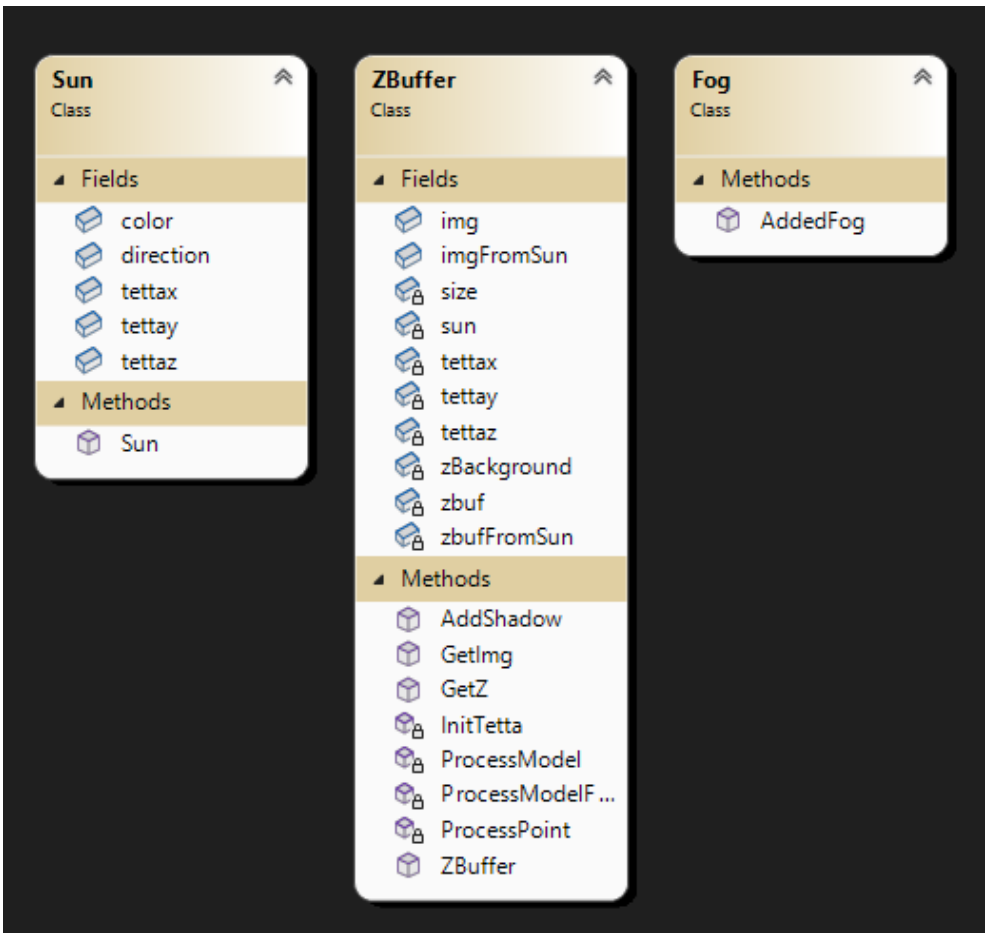


Рис. 3.3 структура классов Sun, ZBuffer, Fog

Sun – хранит в себе цвет света и направление света.

ZBuffer – хранит информации для алгоритма Z буфера.

Fog – класс для добавления тумана на сцену.

3.3 Сведения о модулях программы

Program.cs – главная точка входа в приложение.

Form1.cs – интерфейс.

Sun.cs – описание источника света.

Screen.cs – описание сцены, методы взаимодействия с ней.

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

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

Zbuffer.cs – алгоритм Z буфера.

Transformation.cs – функции преобразования координат точек.

Vector.cs – описание векторов.

3.4 Интерфейс программы

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

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

Группа «Положения солнца»: позволяет быстро выбрать положение и направление источника света.

Группа «Сдвиг солнца»: позволяет изменить положение солнца.

Группа «Добаваить здание»: позволяет добавить новое здание с заданными размерами и положением на сцену.

Группа «Туман»: позволяет задать параметр тумана и наложить туман на текущую сцену или очистить его.

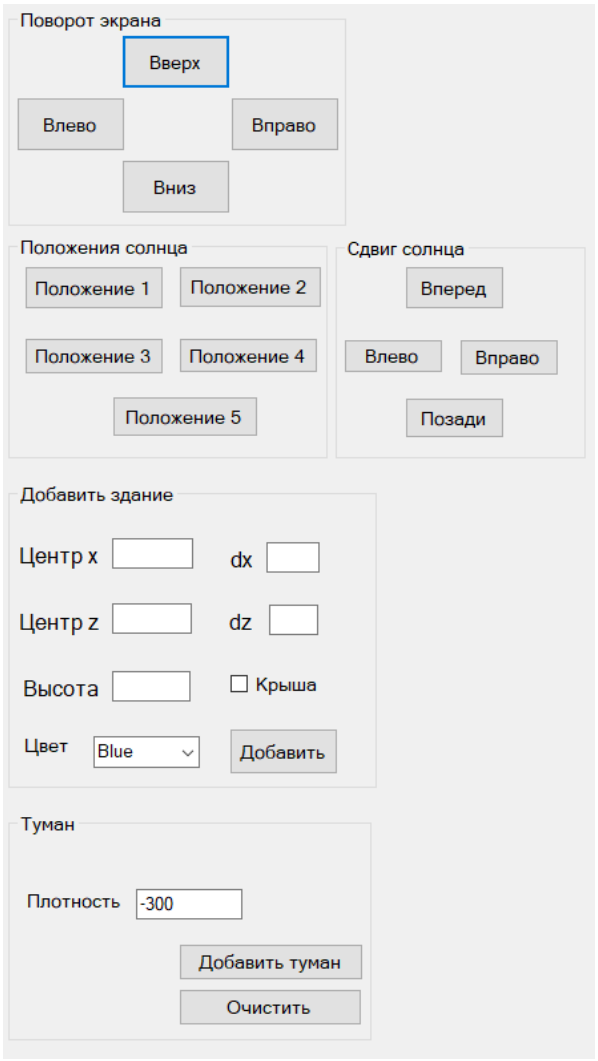


Рис. 3.4 Интерфейс программы – группы настроек

4. Экспериментальная часть

В данном разделе будет проведены примеры работы 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 Демонстрация работы программы

На рисунках 4.1 показан городской ландшафт в обычном режиме освещения с тенями. Можно заметить, что тени отображаются корректно.

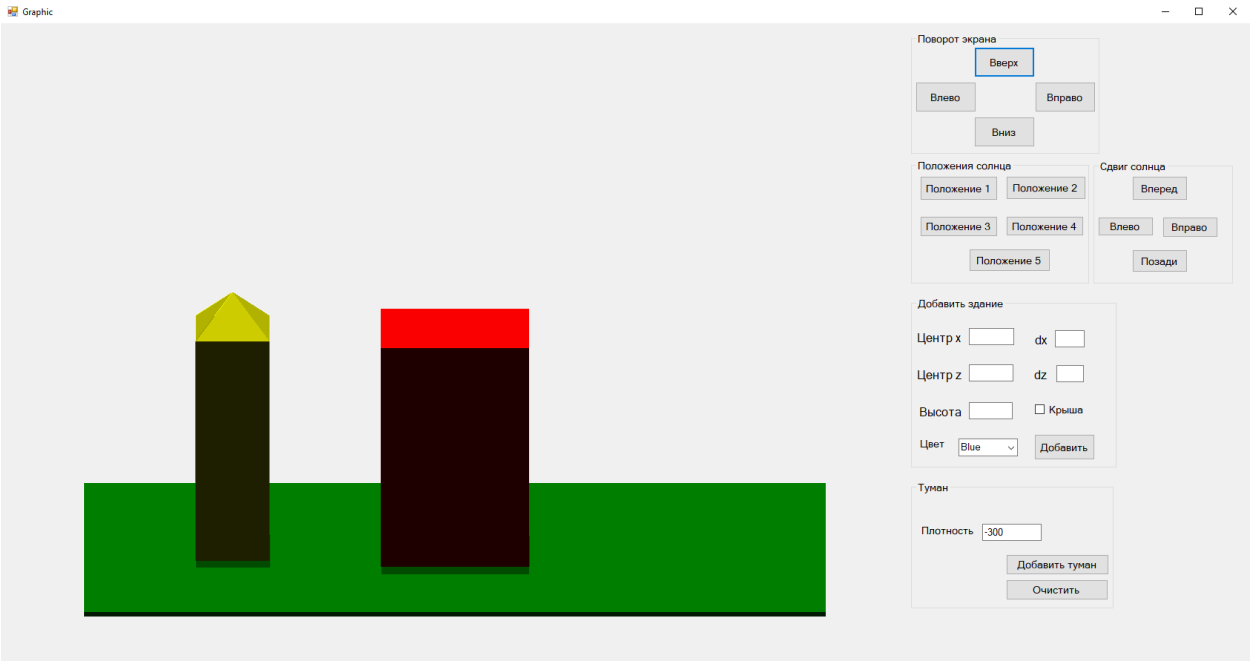


Рис. 4.1 Визуализация сцены в обычном режиме

На рисунках 4.2 и 4.3 показан один и тот же городской ландшафт при разном положении источника. Смена положения источника освещения работает корректно.

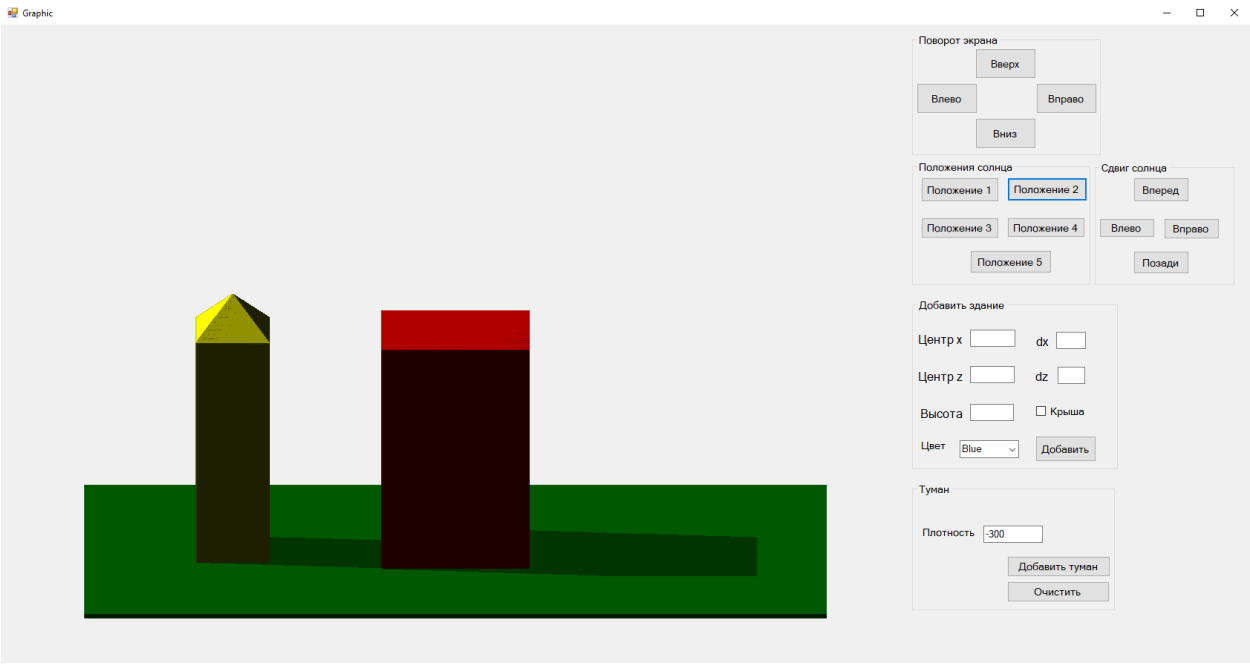


Рис. 4.2 Смена одной направленности источника освещения

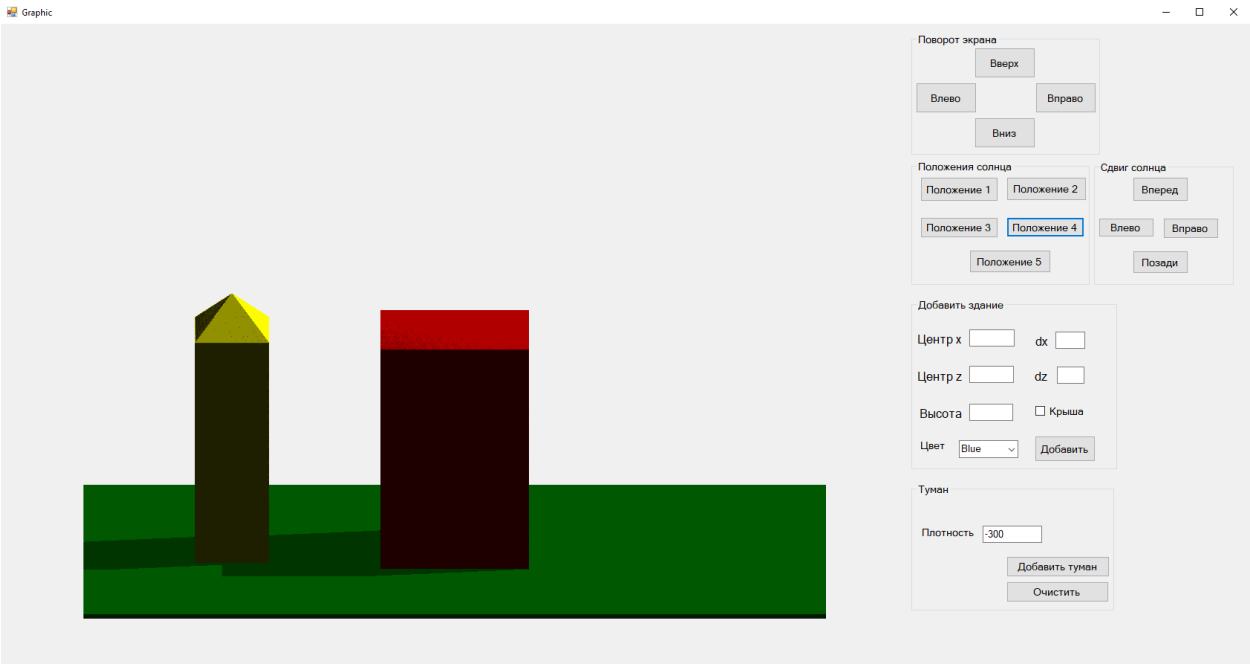


Рис. 4.3 Смена другой направленности источника освещения

На рисунке 4.4 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.

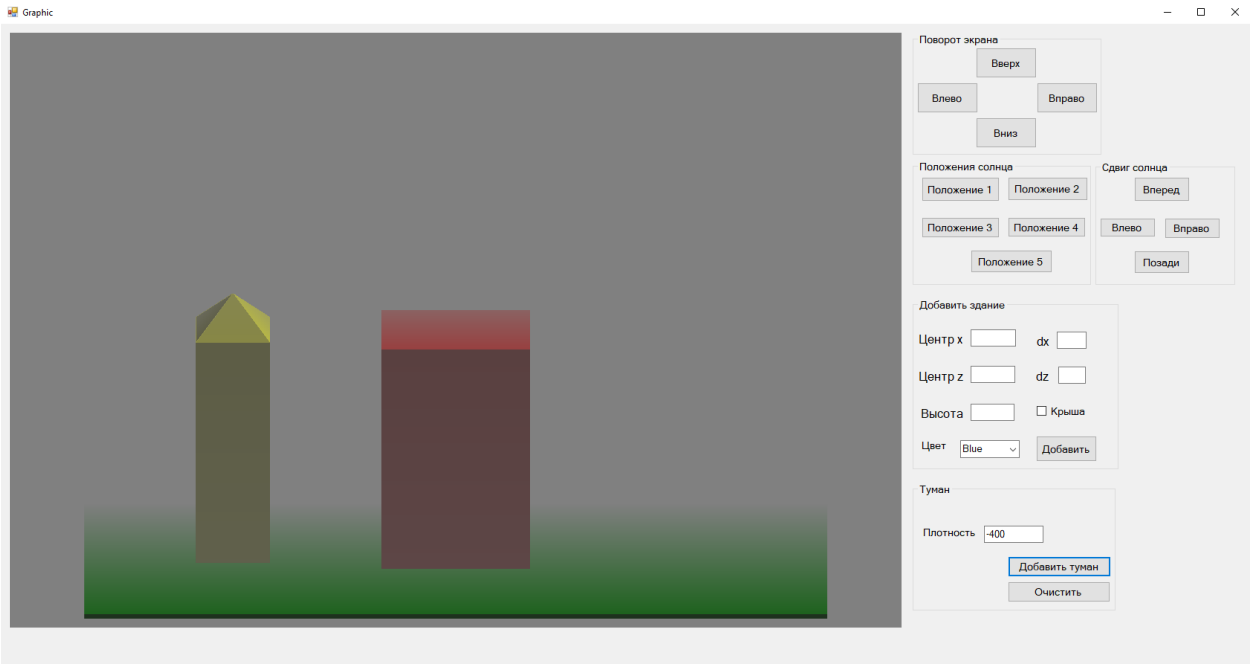


Рис. 4.4 Эффект тумана

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™ i5-1135G7
- 4 ядра
- 8 логических процессоров
- 8 гб оперативной памяти

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

- 5271535 тиков – обычное нахождение теней;
- 15370099 тиков – параллельное нахождение теней.

Из предоставленных данных видно, что параллельное нахождение теней, даже с оптимизацией разбиения, работает медленнее в 2.9 раза чем обычное.

Заключение

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

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

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

Список использованных источников

1. Геометрическое моделирование [Электронный ресурс]. – Режим доступа: http://bourabai.ru/graphics/0209.htm (дата обращения 27.11.22).
2. Bruce Baumgart, Winged-Edge Polyhedron Representation for Computer Vision. National Computer Conference, May 1975.
3. Е. А. Снижко. Компьютерная геометрия и графика, 2005. - 17 с.
4. Алгоритм, использующий z-буфер [Электронный ресурс]. – Режим доступа: http://stratum.ac.ru/education/textbooks/kgrafic/additional/addit21.html (дата обращения 27.11.22).
5. Удаление скрытых линий и поверхностей [Электронный ресурс]. – Режим доступа: http://algolist.ru/graphics/delinvis.php (дата обращения 27.11.22).
6. Создание объемного тумана [Электронный ресурс]. – Режим доступа: http://algolist.ru/graphics/effect/fog.php (дата обращена 27.11.22).

ПРИЛОЖЕНИЕ А

Трехмерная визуализация городской среды в разных погодных условиях

Студент: Динь Вьет Ань ИУ7И-54Б
Научный руководитель: Филиппов М. В

Рисунок 1 – Титульный слайд (слайд 1)

Цель работы

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

1. Описание структуры трехмерной сцены.
2. Выбор и/или модифицирование существующих алгоритмов трехмерной графики, которые позволят визуализировать трехмерную сцену.
3. Реализация данных алгоритмов для создания трехмерной сцены.
4. Разработка программного обеспечения, которое позволит отобразить трехмерную сцену и визуализировать погодные эффекты в городском ландшафте.

Рисунок 2 – Цельработы (слайд 2)

Формализация сцены

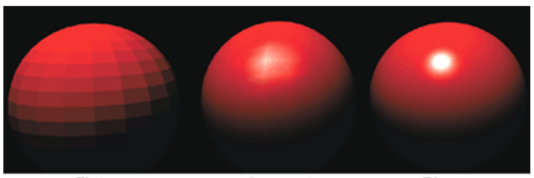
- Плоскость земли
- Здания
- Источник света (солнце)
- Туман

Рисунок 3 – Формализация сцены (слайд 3)

Выбор алгоритмов

Алгоритмы удаления невидимых линий, построения теней

- Обратная трассировка лучей
- Алгоритм Робертса
- Алгоритм Варнока
- Z буфер



Методы закрашивания:

- Простая закраска - один уровень интенсивности на грань
- Закраска по Гуро - билинейная интерполяция интенсивностей
- Закраска по Фонгу - билинейная интерполяция векторов нормалей

Рисунок 4 Выбор алгоритмов (слайд 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

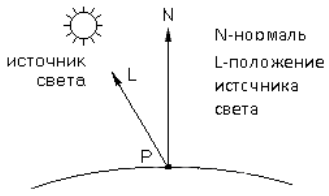


Рисунок 5 – Простой метод освещения (слайд 5)

Общий алгоритм визуализации сцены

1. Получение информации о сцене
2. Выполнение преобразований и расчетов
3. Алгоритм Z-буфера для наблюдателя и источника света
4. Поиск теней
5. Добавление эффекта тумана

Рисунок 6 – Общий алгоритм визуализации сцены (слайд 6)

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

Рисунок 7 – Алгоритм Z – буфера (слайд 7)

Туман

Для того, чтобы создать эффект дымки или плотного тумана нужно знать удаленность от наблюдателя видимых пикселей.

Если пиксель дальше последнего видимого z интенсивность тумана будет равна 1, иначе

$$k = \frac{(z\text{пикс} - z\text{дальнее})}{(z\text{наблюдателя} - z\text{дальнее})},$$

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

Рисунок 8 – Туман (слайд 8)

Выбор языка программирования и среды разработки

В качестве языка программирования был выбран C#:

- ознакомился с этим языком во время занятий по компьютерной графике
- ООП

В качестве среды разработки был выбран Visual Studio 2022:

- она бесплатна для студентов
- она удобства отладки и написания кода

Для создания графического интерфейса был использован Windows Forms.

Рисунок 9 – Выбор языка программирования и среды разработки (слайд 9)

Структура и состав классов

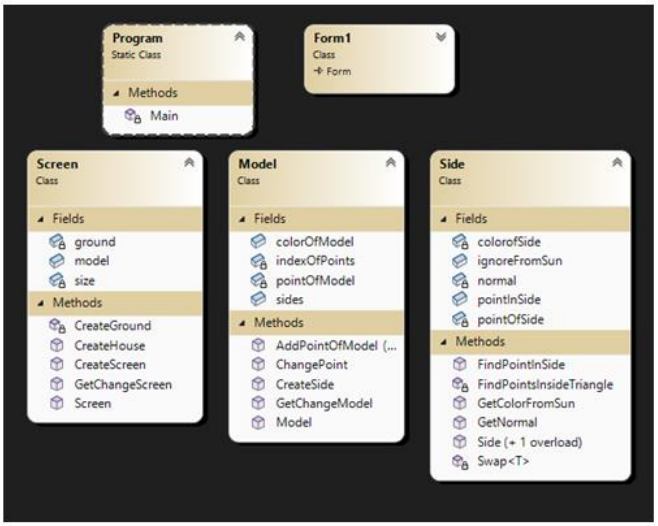


Рисунок 10 – Структура и состав классов, часть 1 (слайд 10)

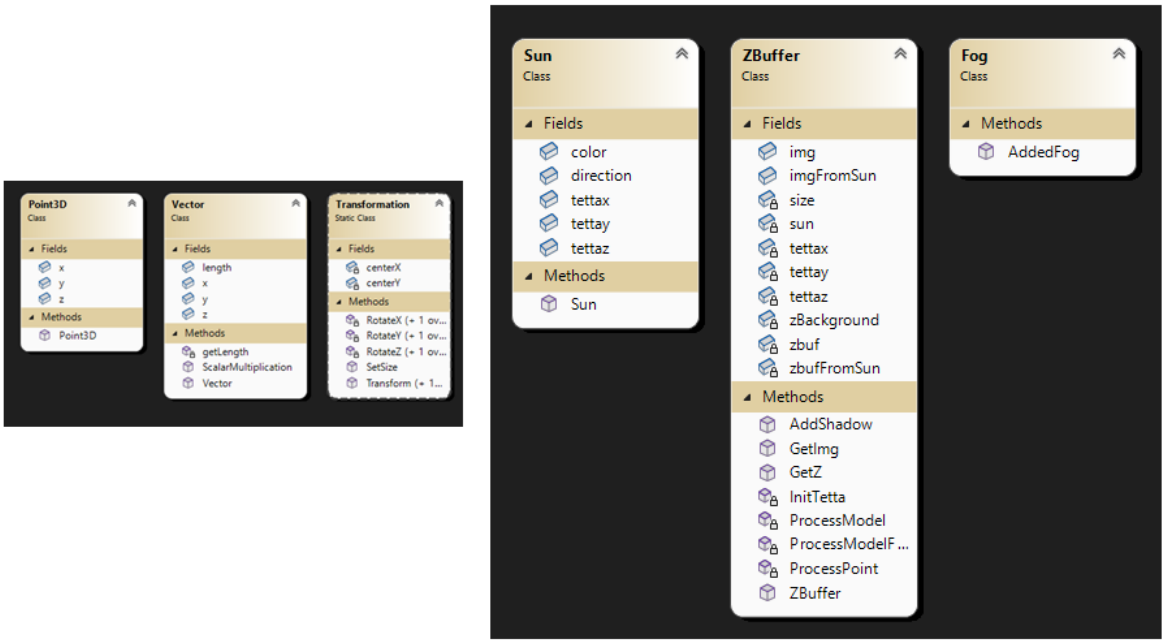


Рисунок 11 – Структура и состав классов, часть 2 (слайд 11)

Интерфейс программы

Интерфейс состоит из групп:

- Поворот экрана - поворот сцены
- Положения солнца – выбрать быстро положение солнца
- Сдвиг солнца – изменить положение солнца
- Добавить здание - добавление зданий
- Туман – добавить или очистить эффект тумана

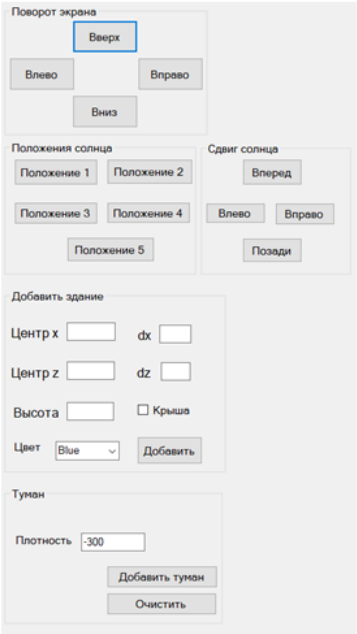


Рисунок 12 – Интерфейс программы (слайд 12)

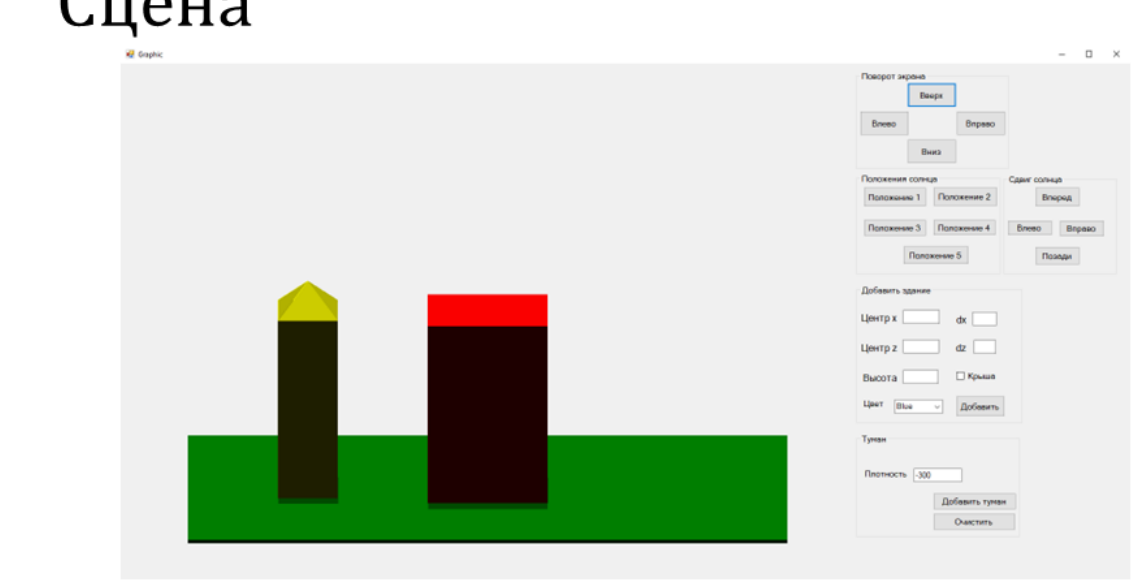


Рисунок 13 – Сцена (слайд 13)

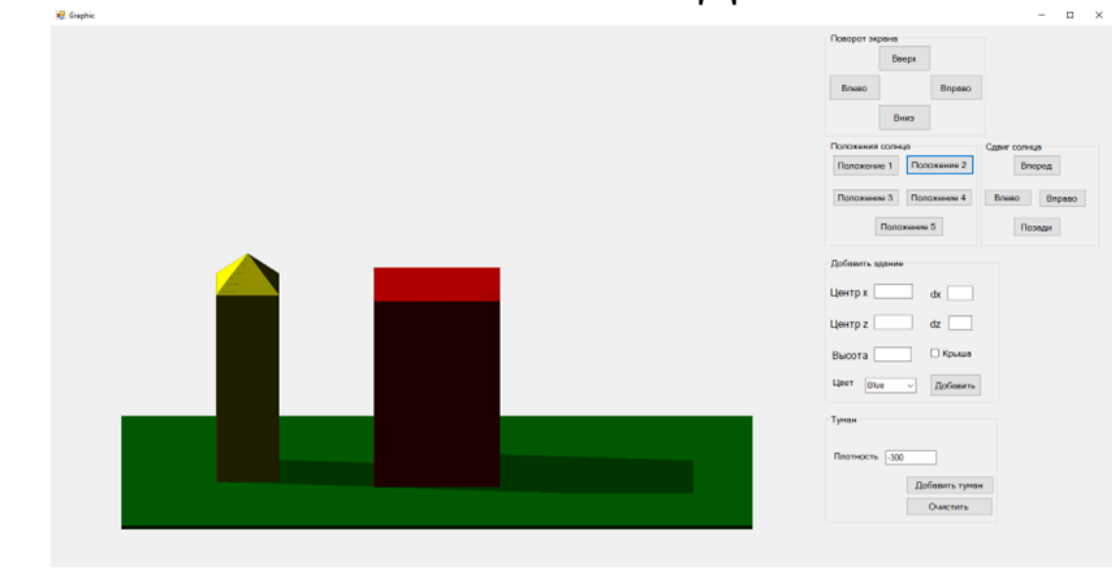


Рисунок 14 – Тени в обычном погоде (слайд 14)

Город в разных точек наблюдения с разными положениями солнца

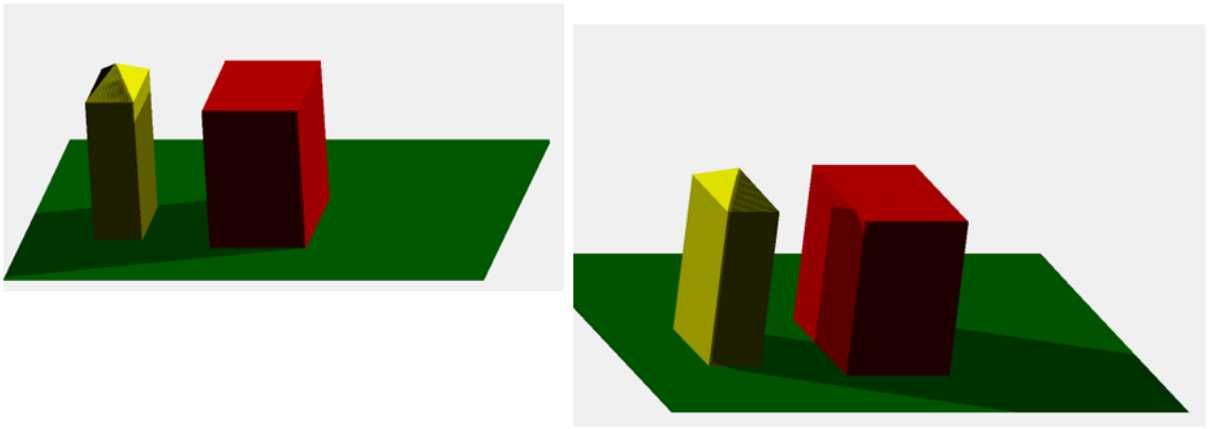


Рисунок 15 – Город в разных точек наблюдения с разными положениями солнца (слайд 15)

Город в тумане

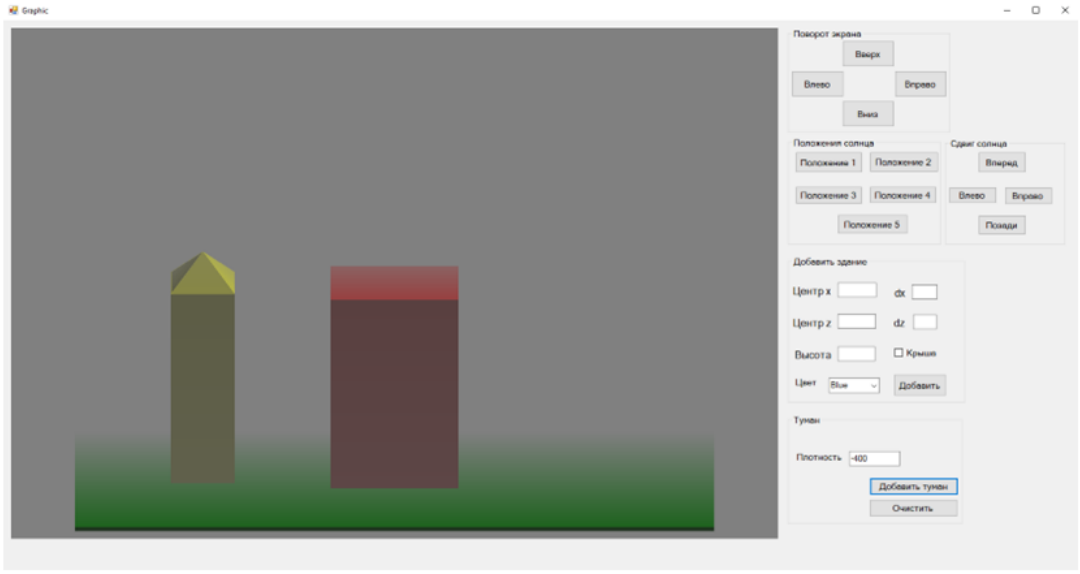


Рисунок 16 –Город в тумане (слайд 16)

Эксперимент

Параллельное нахождение теней с помощью функции Parallel.For, которая производит итерации цикла параллельно, показало увеличение времени работы в 2.9 раза.

- 5271535 тиков – обычное нахождение теней;
- 15370099 тиков – параллельное нахождение теней.

Рисунок 17 – Эксперимент (слайд 17)

Заключение

- Была описана структура трехмерной сцены;
- Были рассмотрены основные алгоритмы удаления невидимых линий, построения теней, методы закрашивания ;
- Выбраны и реализованы наиболее подходящие для решения поставленной задачи;
- Было разработано программное обеспечение для визуализации сцены и эффекта тумана.

Рисунок 18 – Заключение (слайд 18)