|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Динь Вьет Ань\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа \_\_\_\_ИУ7-44Б\_\_\_\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_технологическая\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия \_\_\_\_\_\_МГТУ им. Н. Э. Баумана, каф. ИУ7\_\_\_\_\_\_\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Динь Вьет Ань**\_\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Куров А.В.**\_\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2022 г.*

**Индивидуальное задание:**

Разработать программу для трехмерной визуализации городской среды в различных погодных условиях: сонлце и туман. Источник света – солнце (т.е бесконечно удаленный, в каждой точке сцены поток света имеет одинаковую интенсивность). Выбрать методы построения реалистичных изображений и визуализации погодных условий.

Оглавление

[Введение 4](#_Toc14169990)

[1. Аналитическая часть 5](#_Toc14169991)

[1.1 Формализация объектов синтезируемой сцены 5](#_Toc14169992)

[1.2 Анализ алгоритмов удаления невидимых линий и поверхностей 6](#_Toc14169993)

[1.3 Анализ методов закрашивания 8](#_Toc14169994)

[1.4 Анализ алгоритмов построения теней 9](#_Toc14169995)

[2. Конструкторская часть](#_Toc14169997) 10

[2.1 Общий алгоритм решения задачи 10](#_Toc14169998)

[2.2 Алгоритм Z-буфера 1](#_Toc14169999)0

[2.3 Простой метод освещения 1](#_Toc14170000)0

[2.4 Выбор используемых типов и структур данных 1](#_Toc14170002)1

[3. Технологическая часть 1](#_Toc14170003)2

[3.1 Выбор и обоснование языка программирования и среды разработки 1](#_Toc14170004)2

[3.2 Сведения о модулях программы 1](#_Toc14170006)2

[Заключение 1](#_Toc14170008)3

[Список использованной литературы 14](#_Toc14170009)

# Введение

В современном мире компьютерная графика является неотъемлемой частью жизни человека. Он повсеместно используется для визуального представления данных и создания эффектов в компьютерных играх и фильмах.

В следствие этого перед людьми, создающими трехмерные сцены, встает задача создания реалистичных изображений, которые будут учитывать такие оптические явления как преломление, отражение и рассеивание света. Дифракция света, интерференция и вторичные отражения учитываются для создания более реалистичного изображения.

Существует множество алгоритмов компьютерной графики, решающих эту задачу. Эти алгоритмы часто являются ресурсоемкими. Чем лучше будет полученное изображение, тем больше времени и памяти вы потратите на его композицию. Это становится проблемой при создании динамических сцен, которые необходимо пересчитывать каждый временной интервал.

Моей целью во время практики - выбрать или модифицировать существующие алгоритмы и реализовать эти алгоритмы для создания 3D-сцен.

# 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 света расположен в бесконечности, имеет направленность). Положение задается трехмерными координатами, а цвет свечения описывается RGB параметрами.
* Солнце – 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 Оxz. Размеры задаются шириной и длиной и цветом RGB.
* Здания – правильная призма 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.

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. Список граней - наиболее удобным способом хранения моей сцены 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

**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.

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

## 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.

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

**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.

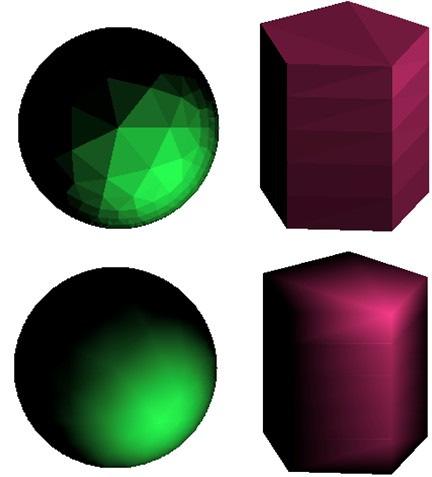


Рисунок 1 Сравнение методов закрашивания

## 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.

# 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. 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:
   1. Для каждого пикселя, связанного с многоугольником вычислить его глубину z(x, y)
   2. 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.

1. Отобразить результат

## 2.3 Простой метод освещения

В простом методе освещения интенсивность рассчитывается по закону Ламберта:

I = I0\*cos(α), где

I – результирующая интенсивность света в точке

I0 – интенсивность источника

α – угол между нормалью к поверхности и вектором направления света

## 2.4 Выбор используемых типов и структур данных

* Источник света – задается расположением и направленностью света.
* 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 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 Сведения о модулях программы

Program – главная точка входа в приложение;

Form1 – интерфейс;

Light – описание источников света;

Fog – описание тумана;

Model – Описание объектов сцены;

Zbuffer – Алгоритм Z буфера.

# 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>

1. 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>
2. Е. А. Снижко. Компьютерная геометрия и графика [Текст], 2005. - 17 с.
3. Проблемы трассировки лучей – из будущего в реальное время. [Электронный ресурс]. – Режим доступа: <https://nvworld.ru/articles/ray-tracing/3/>
4. RayTracing – царь света и теней, Лев Дымченко [Электронный ресурс]. – Режим доступа: <https://old.computerra.ru/206167/>
5. Реалистичная визуализация атмосферных осадков в приложениях реального времени, В. В. Карабчевский, А.А. Лунтовская, Донецк ДонНТУ [Текст], 2015
6. K. Garg, S. K. Nayar. Photorealistic rendering of rain streaks. In ACM SIGGRAPH 2006 Papers. SIGGRAPH '06. ACM, New York, NY, 996‐ 1002 с.