



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 \_\_\_\_\_ Нгуен Ань Тхы \_\_\_\_\_  
*фамилия, имя, отчество*

Группа \_\_\_\_\_ ИУ7-И46Б \_\_\_\_\_

Тип практики \_\_\_\_\_ Стационарная \_\_\_\_\_

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

Студент \_\_\_\_\_ Нгуен Ань Тхы \_\_\_\_\_  
*подпись, дата* *фамилия, и.о.*

Руководитель практики \_\_\_\_\_ Куров А.В. \_\_\_\_\_  
*подпись, дата* *фамилия, и.о.*

Оценка \_\_\_\_\_

2020 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)  
)  
« \_\_\_\_ » \_\_\_\_\_ 2020 г.

## **ЗАДАНИЕ на выполнение курсового проекта**

по дисциплине \_\_\_\_\_ Компьютерная графика \_\_\_\_\_

Студент группы ИУ7-И46Б \_\_\_\_\_

\_\_\_\_\_ Нгуен Ань Тхы \_\_\_\_\_  
(Фамилия, имя, отчество)

Тема курсового проекта \_\_\_\_\_  
\_\_\_\_\_ Моделирование визуализации городской среды и погодных условий. \_\_\_\_\_

Направленность КП (учебный, исследовательский, практический, производственный, др.) \_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

### **1. Задание**

Разработать программу для трехмерной визуализации городской среды и погодных условий. Сцена включает здания, лежащие на земле, источник света – солнце, рассматривается как материальную точку расположен в бесконечности. Визуализировать основные погодные условия: дождь, солнце и туман. Программа позволяет пользователю изменять положения обзора и источника света. Анализ существующих алгоритмов и выбрать из них подходящие для наиболее эффективного построения проекта.

### **2. Оформление курсового проекта:**

2.1. Расчетно-пояснительная записка на 25-30 листах формата А4. Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.). На защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.

Дата выдачи задания « \_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

**Руководитель курсового проекта**

\_\_\_\_\_ Силантьева.А.В \_\_\_\_\_  
(Подпись, дата) (И.О.Фамилия)

**Студент**

\_\_\_\_\_ Нгуен Ань Тхы \_\_\_\_\_  
(Подпись, дата) (И.О.Фамилия)

## Оглавление:

Введение.....	4
1. Аналитическая часть.....	5
1.1 Анализ алгоритмов удаления невидимых линий и поверхностей .....	5
1.1.1 Алгоритм Робертса.....	5
1.1.2 Алгоритм трассировки лучей .....	6
1.1.3 Алгоритм Z-буфера.....	7
1.2 Анализ алгоритмов закрашивания .....	9
1.2.1 Однотонная закрашка.....	9
1.2.2 Метод Гуро.....	10
1.2.3 Метод Фонга .....	11
1.3 Анализ алгоритм построения тени.....	12
1.3.1 Построение теней с использованием алгоритма z-буфера.....	12
2. Конструкторская часть .....	13
2.1 Общий алгоритм решения задачи.....	13
2.2 Формализация объектов синтезируемой сцены.....	13
2.3 Алгоритм удаления невидимых граней с использованием z – буфера....	13
3. Технологическая часть .....	16
3.1 Выбор и обоснование языка программирования и среды разработки.....	18
3.2 Описание структуры программы.....	18
3.3 Интерфейс.....	18
4. Заключение.....	16
5. Список использованных источников .....	17

## **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.

Цель проекта - разработка программу создания трехмерной графической сцены для визуализации городской среды и погодных условий.

Для достижения поставленной цели необходимо решить следующие задачи: провести анализ существующих алгоритмов удаления невидимых линий и поверхностей, закраски, текстурирования, а также моделей освещения и выбрать из них подходящие для наиболее эффективного выполнения проекта.

## 1. Аналитическая часть

### 1.1 Анализ алгоритмов удаления невидимых линий и поверхностей

#### 1.1.1 Алгоритм Робертса

Алгоритм Робертса представляет собой первое известное решение задачи об удалении невидимых линий. Это математически элегантный метод, работающий в объектном пространстве.

Работа Алгоритм Робертса проходит в два этапа:

- Определение нелицевых граней для каждого тела отдельно.
- Определение и удаление невидимых ребер.

Уравнение произвольной плоскости в трехмерном пространстве:

$$ax + by + cz + d = 0$$

В матричной форме этот результат выглядит так:

$$[x \ y \ z \ 1][P]^T = 0$$

Где  $[P]^T = [a \ b \ c \ d]$  представляет собой плоскость. Поэтому любое выпуклое твердое тело можно выразить матрицей тела, состоящей из коэффициентов уравнений плоскостей, т. е.

$$[V] = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{bmatrix}$$

Подстановка координат трех неколлинеарных точек  $(x_1, y_1, z_1)$ ,  $(x_2, y_2, z_2)$ ,  $(x_3, y_3, z_3)$  в нормированное уравнение дает:

$$ax_1 + by_1 + cz_1 = -1;$$

$$ax_2 + by_2 + cz_2 = -1;$$

$$ax_3 + by_2 + cz_3 = -1.$$

В матричной форме это выглядит так:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

$$\text{Или: } [X][C] = [D] \Rightarrow [C] = [X]^{-1}[D].$$

Другой способ используется, если известен вектор нормали к плоскости:

$$n = ai + bj + ck,$$

где  $i, j, k$  – единичные векторы осей  $x, y, z$  соответственно.

В частности, если компоненты этой точки на плоскости  $(x_l, y_l, z_l)$ , то

$$d = -(ax_l + by_l + cz_l)$$

- Удаление нелицевых граней тела:

$E = [0, 0, -1, 0]$  – вектор наблюдателя.

Для определения невидимых граней:  $[E][V]$  отрицательные компоненты полученного вектора будут соответствовать невидимым граням.

$[E].n > 0$  – грань видима.

$[E].n < 0$  – грань невидима.

$[E].n = 0$  – грань на границе видимости.

- Удаление отрезков, экранируемых другими телами

$P_1P_2$  – исследуемый отрезок.  $P(t) = P_1 + (P_2 - P_1)t$  – параметрическое уравнение. Нужно найти  $t$ , при котором изменяется видимость отрезка. Затем определяют минимальное значение параметра  $t$ . Отрезок невидим при  $t_{\min} < t < t_{\max}$ .

После определения частично видимых или полностью невидимых отрезков определяют пары объектов, связанных отношением протыкания (в случае протыкания объектов сцены ищутся решения на границе  $\alpha = 0$ ), и вычисляют отрезки, которые образуются при протыкании объектами друг друга.

### 1.1.2 Алгоритм трассировки лучей

Трассировка лучей — один из методов геометрической оптики — исследование оптических систем путём отслеживания взаимодействия отдельных лучей с поверхностями. В узком смысле — технология построения изображения трёхмерных моделей в компьютерных программах, при которых отслеживается обратная траектория распространения луча (от экрана к источнику).

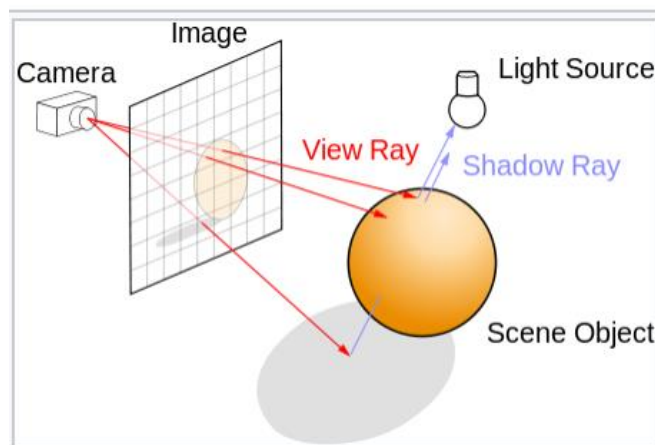


Рисунок 1. Алгоритм трассировки лучей.

#### Достоинства

Возможность рендеринга гладких объектов без аппроксимации их полигональными поверхностями (например, треугольниками).

Вычислительная сложность метода слабо зависит от сложности сцены.

Высокая алгоритмическая распараллеливаемость вычислений — можно параллельно и независимо трассировать два и более лучей, разделять участки (зоны экрана) для трассирования на разных узлах кластера и т.д.

Отсечение невидимых поверхностей, перспектива и корректное изменение поля зрения являются логическим следствием алгоритма.

#### Недостатки

Серьёзным недостатком метода обратного трассирования является производительность. Метод растеризации и сканирования строк использует

когерентность данных, чтобы распределить вычисления между пикселями. В то время как метод трассирования лучей каждый раз начинает процесс определения цвета пикселя заново, рассматривая каждый луч наблюдения в отдельности. Впрочем, это разделение влечёт появление некоторых других преимуществ, таких как возможность трассировать больше лучей, чем предполагалось для устранения контурных неровностей в определённых местах модели. Также это регулирует отражение лучей и эффекты преломления, и в целом — степень фотореалистичности изображения.

### 1.1.3 Алгоритм Z-буфера

Это один из простейших алгоритмов удаления невидимых поверхностей.

Главное преимущество алгоритма — его простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоемкости алгоритма не более чем линейна. Поскольку элементы сцены или картинки можно заносить в буфер кадра или в z-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Поэтому экономится вычислительное время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма — большой объем требуемой памяти. Если сцена подвергается видовому преобразованию и отсекается до фиксированного диапазона координат  $z$  значений, то можно использовать z-буфер с фиксированной точностью.

Напомним, что уравнение плоскости имеет вид:

$$Ax + By + Cz + D = 0$$
$$z = -\frac{Ax + By + D}{C}, \quad C \neq 0$$



При  $C = 0$  плоскость многоугольника параллельна оси  $Z$ .

У сканирующей строки  $y = \text{const}$  и глубина пиксела на этой строке, у которого  $x_1 = x + \Delta x$ , равна:

$$z_1 - z = -\frac{ax_1 + d}{c} + \frac{ax + d}{c} = \frac{a(x - x_1)}{c}$$

Так как  $\Delta x = 1$ , то  $z_1 = z - \frac{A}{c}$

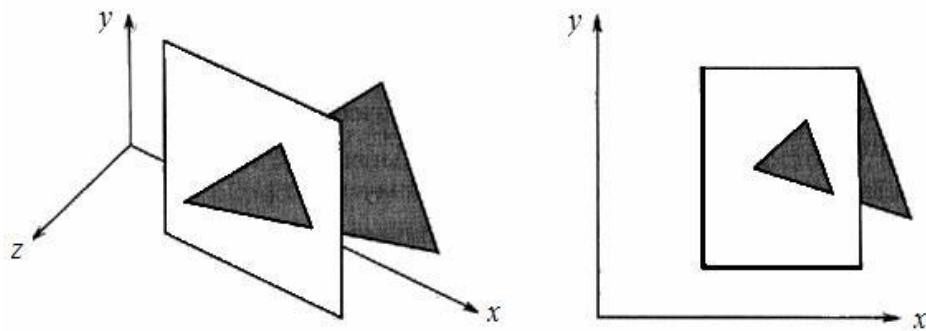


Рисунок 2. Пример алгоритма с использованием Z-буфера

Алгоритм, использующий  $z$ -буфер, можно также применить для построения сечений поверхностей. Изменится только оператор сравнения:

$$z(x, y) > Z_{\text{буфер}}(x, y) \text{ and } z(x, y) \leq Z_{\text{сечения}}.$$

где  $Z_{\text{сечения}}$  — глубина искомого сечения. Эффект заключается в том, что остаются только такие элементы поверхности, которые лежат на самом сечении или позади него.

### Вывод:

Поскольку размеры пространства изображения фиксированный, поэтому для экономия вычислительное время выбрать алгоритм  $z$ -буфера. Этот алгоритм просто решает проблему пересечения сложных поверхностей.

## 1.2 Анализ алгоритмов закрашивания

### 1.2.1 Однотонная закрашка

Это наиболее простой и требующий наименьших вычислительных ресурсов метод закрашки поверхности. Цвет всей поверхности рассчитывается согласно закону Ламберта. Он формулируется так: плоская поверхность, имеющая одинаковую яркость по всем направлениям, отражает свет, интенсивность которого изменяется по закону косинуса:

$$I = I_0 \cos \theta$$

Где:

$I_0$  – интенсивность отражается в направлении нормали к поверхности.

$\theta$  – угол между направлением на наблюдателя и нормалью к поверхности.

Метод гранения позволяет получать изображения, сравнимые по качеству с реальными объектами, лишь при выполнении следующих условий:

- источник света находится на большом расстоянии от объекта.
- наблюдатель находится на большом расстоянии от объекта.
- каждая грань тела является гранью многогранника, а не аппроксимирующей поверхностью.
- поверхность аппроксимирована большим числом небольших плоских граней.

### 1.2.2 Метод Гуро

Метод закрашки Гуро основан на интерполяции интенсивности. Он позволяет устранить дискретность изменения интенсивности и создать иллюзию гладкой криволинейной поверхности.

Процесс закрашки по методу Гуро осуществляется в четыре этапа:

- i. Вычисляются нормали ко всем полигонам.
- ii. Определяются нормали в вершинах путем усреднения нормалей по всем полигональным граням, которым принадлежит рассматриваемая вершина,
- iii. Используя нормали в вершинах и, применяя определенную модель освещения, вычисляют значения интенсивностей в вершинах многоугольника.
- iv. Каждый многоугольник закрашивается путем линейной интерполяции значений интенсивностей в вершинах сначала вдоль каждого ребра, а затем и между ребрами вдоль каждой сканирующей строки

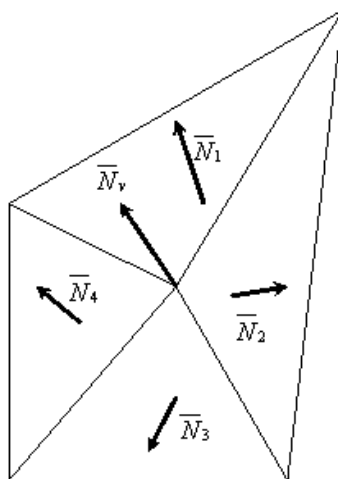


Рисунок 3. Определение нормалей.

Метод Гуро применим только для небольших граней, расположенных на значительном расстоянии от источника света. Если же размер грани достаточно велик, то расстояние от источника света до ее центра будет значительно меньше, чем до ее вершин, и, согласно закону освещенности, центр грани должен быть освещен сильнее ребер. Однако модель изменения освещенности, принятая в методе Гуро, предполагает линейное изменение яркости в пределах грани и не позволяет сделать середину грани ярче, чем ее края. В итоге на изображении появляются участки с неестественной освещенностью.

### **1.2.3 Метод Фонга**

Метод закрашки Фонга основан на интерполяции вектора нормали, который затем используется в модели освещения для вычисления интенсивности пиксела.

Процесс закрашки по методу Фонга осуществляется в четыре этапа:

- i. Определяются нормали к граням.
- ii. По нормальям к граням определяются нормали в вершинах.
- iii. В каждой точке закрашиваемой грани определяется интерполированный вектор нормали.
- iv. По направлению векторов нормали определяется цвет точек грани.

Закраска Фонга требует больших вычислительных затрат, однако при этом достигается лучшая локальная аппроксимация кривизны поверхности, получается более реалистичное изображение, правдоподобнее выглядят зеркальные блики.

#### **Вывод:**

Поскольку сцена состоит из плоскостей, поэтому закрашка по Фонгу и Гуро будет сложно и мешать ребра зданий будут сглажены. Поэтому лучше всего использовать тот же метод однотонной закрашки.

## **1.3 Анализ алгоритм построения тени**

### **1.3.1 Построение теней с использованием алгоритма z-буфера**

Алгоритм удаления невидимых поверхностей с использованием z-буфера легко может быть модифицирован и для того, чтобы учесть эффект отбрасывания теней.

При построении теней с использованием алгоритма z-буфера выполняется два прохода: один - относительно источника света, другой - относительно наблюдателя. Для этого выделяется отдельный "теневой" z-буфер. Первый проход необходим для того, чтобы определить, какие точки видны со стороны источника света. При втором проходе сцена визуализируется из положения наблюдателя с учетом того, что точки, которые оказались невидимыми со стороны источника света, находятся в тени.

Таким образом, алгоритм работает в два этапа. При первом проходе сцена рассчитывается при совмещении точки наблюдения с положением источника света. Значения глубины пикселей для данного вида заносятся в "теневой" z-буфер, а значения интенсивности не учитываются.

При втором проходе сцена строится из точки, в которой на самом деле находится наблюдатель. При сканировании каждой поверхности значение ее глубины каждого ее пиксела сравнивается со значением глубины в z-буфере.

Если поверхность видима, то необходимо проверить, видима ли данная точка со стороны источника света. Для этого координаты точки  $x, y, z$  из вида наблюдателя линейно преобразуются в координаты  $x', y', z'$  на виде из источника света. Проверка на видимость осуществляется сравнением значения, которое хранится в "теновом" z-буфере для координат  $x', y'$ , и значения  $z'$ . Если точка невидима для источника света (значение в "теновом" z-буфере больше значения  $z'$ ), значит она находится в тени и ее свечение вычисляется с учетом затенения. Если же точка видима из положения источника света, то она изображается без изменений.

## **2. Конструкторская часть**

### **2.1 Общий алгоритм решения задачи**

- Задать объекты сцены
- Задать положения источника цвета и наблюдателя
- Использовать алгоритм  $z$  – буфера для получения изображения сцены с тени
- Отображение дожди

### **2.2 Формализация объектов синтезируемой сцены**

Сцена состоит из:

- Плоскость земли (Параметры задаются шириной, длиной и цветом в виде RGB): Ограничивающая плоскость расположена под объектов.
- Здания (Параметры задаются координатом центра основания на земле, высотой и цветом в виде RGB): Прямоугольный параллелепипед с основанием, параллельным плоскости земли, и боковыми ребрами, перпендикулярными плоскости земли.
- Источник цвета (Параметры задаются трехмерными координатами, цветом в виде RGB)
- Дождь, туман, солнце (Параметры задаются интенсивностью и направлением движения)

### **2.3 Алгоритм удаления невидимых граней с использованием $z$ – буфера**

- i. Заполнить  $z$ -буфер элементами с фоновым значением цвета и минимальным значением  $z$ ;
- ii. все точки системы частиц занести в  $z$ -буфер;
- iii. декомпозировать каждый многоугольник на треугольники:

- a. запомнить первую вершину;
  - b. запомнить треугольник, который первая вершина образует с каждой следующей парой вершин;
- iv. Для каждого треугольника:
- a. найти  $umin$  и  $umax$ ;
  - b. пустить сканирующую строку от  $umin$  до  $umax$ ;
  - c. для каждой сканирующей строки найти глубину  $z$  каждой точки  $(x, y)$ ;

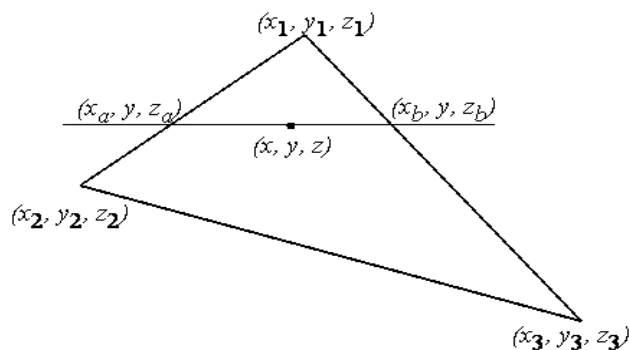


Рисунок 4. Сканирующая строка по грани

Для рисунка  $y$  меняется от  $y_1$  до  $y_2$  и далее до  $y_3$ , при этом для каждой строки определяется  $x_a, z_a, x_b, z_b$ :

$$x_a = x_1 + (x_2 - x_1) * \frac{(y - y_1)}{(y_2 - y_1)};$$

$$x_b = x_1 + (x_3 - x_1) * \frac{(y - y_1)}{(y_3 - y_1)};$$

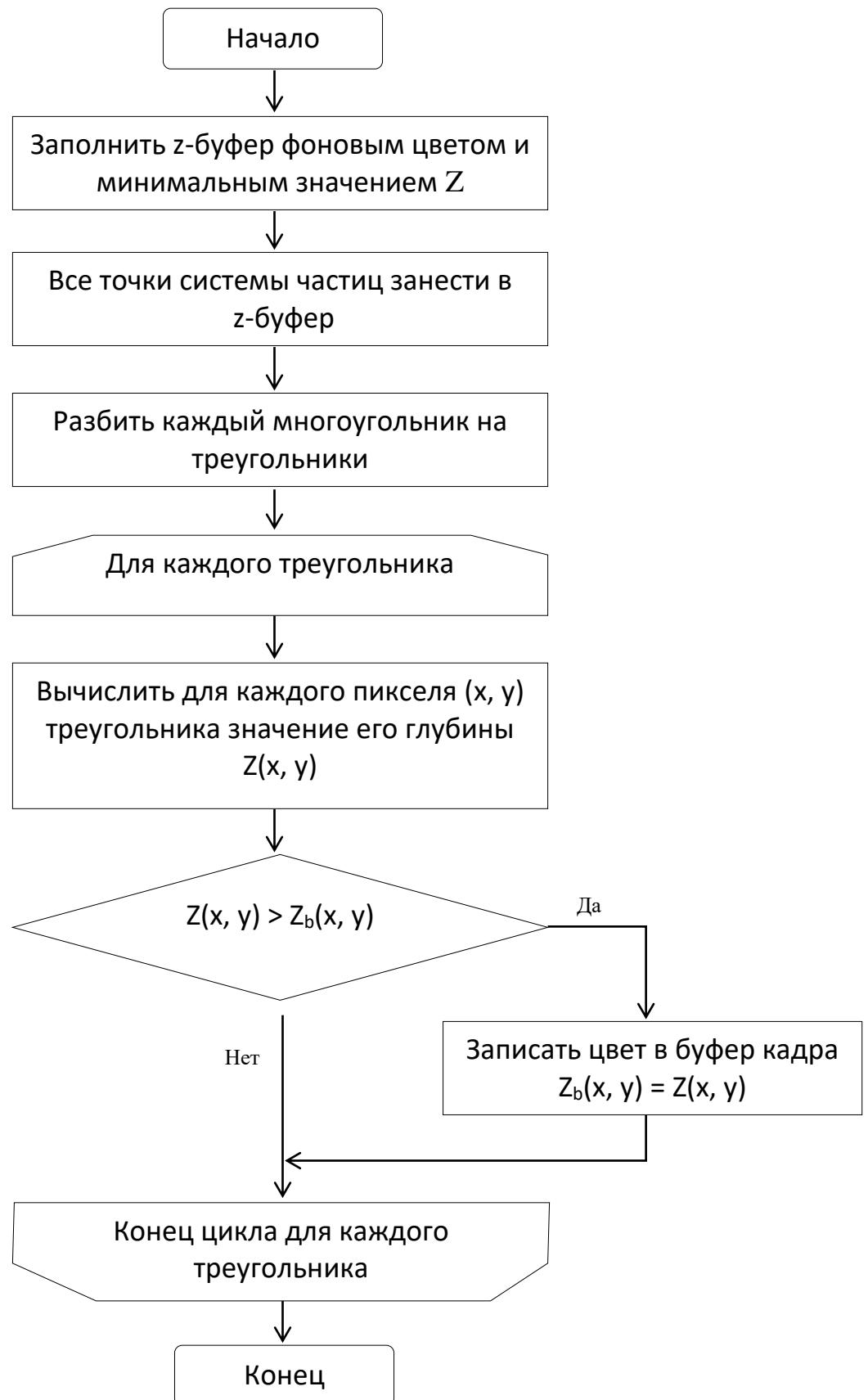
$$z_a = z_1 + (z_2 - z_1) * \frac{(y - y_1)}{(y_2 - y_1)};$$

$$z_b = z_1 + (z_3 - z_1) * \frac{(y - y_1)}{(y_3 - y_1)};$$

- d. получить  $z$  для каждого  $x$  из  $(x_a, x_b)$  сравнить со значением в  $z$ -буфере:
  - если посчитанное значение больше – занести его и соответствующий пикселю цвет в  $z$ -буфер;

➤ иначе никаких действий не производить.

### Схема алгоритма z – буфера:





### **3. Технологическая часть**

#### **3.1 Выбор и обоснование языка программирования и среды разработки**

В качестве языка программирования, Python был выбран для выполнения этого проекта по причинам: я работала с этим языком во время выполнения лабораторных работ по Компьютерной графике. Поэтому можно сократить время написания коды программы.

В качестве среды разработки была выбрана «Visual Studio Code» потому что ее легко использовать и бесплатна в использовании студентами, и предварительно установлен в свой машине.

#### **3.2 Описание структуры программы**

Программа состоит из 4 модулей:

1. main: Главная точка входа в приложение, создать интерфейс.
2. structures: Содержит описание структура сцены и состав классов.
- Объекты (Задания и земли):

class object:

```
def __init__(self, polygons, location, color):
```

```
self.polygons = polygons
```

```
self.location = location
```

```
self.color = color
```

- Вектор источника света:

class light:

```
def __init__(self, direction, intensity):
```

```
self.direction = direction
```

```
self.intensity = intensity
```

3. Zbuffer: алгоритм, использующий z буфер

- z-буффер:

```
class z_buffer:
```

```
    def __init__(self, w, h, value):  
        self.buf = init_buf(w, h, value)
```

4. weather: изображает дождь, солнце и туман

### **3.3 Интерфейс**

Левая часть: Результат программы.

Правая часть: Панель управления программой, позволяющий изменять параметры:

- Источник света: возможность изменять направление и интенсивность.
- Сцена: возможность добавить и удалить объекты в сцене.
- Дождь, солнце и туман: возможность изменять интенсивность и направление движения

## **Заключение**

Во время выполнения проекта, были рассмотрены основные алгоритмы удаления невидимых линий, алгоритмы закрашивания, методы построения и проанализированы их достоинства, недостатки, сделаны выводы, выбаны наиболее подходящие алгоритмов для решения задачи.

Разработанный программный продукт синтезирует трехмерное изображение с помощью алгоритмов компьютерной графики. Программа реализована таким образом, что пользователь может легко использовать и видеть результат.

#### **4. Список использованных источников**

1. <http://stratum.ac.ru/education/textbooks/kgrafic/contents.html>
2. <https://studfile.net/preview/1751396/page:5/>
3. АЛГОРИТМЫ ПОСТРОЕНИЯ ТЕНЕЙ, А.Н.Романюк, М.В.Куринный  
<https://core.ac.uk/download/pdf/52159835.pdf>
4. Лекционные записки по “Компьютерной Графике” МГТУ им. Баумана –  
А.В.Куров.
5. [http://compgraph.tpu.ru/Del\\_hide\\_line.htm](http://compgraph.tpu.ru/Del_hide_line.htm)