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:

Моделирование визуализации городской среды и погодных условий.

Студент __ИУ7-56Б___ _____ **__Нгуен Ань Тхы__**__
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ **__Силантьева.А.В ___**
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

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)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине _____ Компьютерная графика _____

Студент группы ____ ИУ7-И56Б ___

_____ 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 визуализции 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 ползователю изменять положения обзора и источника света.

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:

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 подходяшие для наиболее эффективного выполнения проекта.

1. Аналитическая часть:

1.1 Анализ алгоритмов трехмерных преобразований:

1.1.1 Способы хранения и обработки декартовых координат:

Координаты можно хранить в форме вектор-столбца [x, y, z]. Однако в этом случае неудобно применять преобразования поворота, так такой вектор нельзя умножить на соответствующие матрицы трансформации размерности четыре. Целесообразнее использовать вектор-столбцы размерности четыре - [x,y,z,w], где w для точки равно одному. Преобразования координат выполняются умножением слева преобразуемого вектора-столбца на соответствующую матрицу линейного оператора.

1.1.2 Матрицы аффинных преобразований декартовых координат:

1) сдвиг точки на dx, dy, dz по координатным осям:

$$\begin{cases} X = x + dx \\ Y = y + dy \\ Z = z + dz \end{cases} \quad \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (1)$$

2) масштабирование относительно начала координат с коэффициентами k_x, k_y, k_z:

$$\begin{cases} X = x \cdot k_x \\ Y = y \cdot k_y \\ Z = z \cdot k_z \end{cases} \quad \begin{pmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2)$$

3) поворот относительно осей x, y, z на угол φ:

- ось x:

$$\begin{cases} X = x \\ Y = y \cdot \cos\phi + z \cdot \sin\phi \\ Z = -y \cdot \sin\phi + z \cdot \cos\phi \end{cases} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3)$$

- ось y:

$$\begin{cases} X = x \cdot \cos\phi - z \cdot \sin\phi \\ Y = y \\ Z = x\sin\phi + z\cos\phi \end{cases} \qquad \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (4)$$

- ось z:

$$\begin{cases} X = x \cdot \cos\phi + y \cdot \sin\phi \\ Y = -x \cdot \sin\phi + y \cdot \cos\phi \\ Z = z \end{cases} \qquad \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (5)$$

1.2 Анализ алгоритмов удаления невидимых линий и поверхностей:

1.2.1 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:

$$ax + by + cz + d = 0 \qquad (6)$$

В матричной форме этот результат выглядит так:

$$[x\ y\ z\ 1][P]^T = 0 \quad (7)$$

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.

$$[V] = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{bmatrix} \quad (8)$$

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:

$$\begin{cases} ax1 + by1 + cz1 = -1 \\ ax2 + by2 + cz3 = -1 \\ ax3 + by2 + cz3 = -1 \end{cases} \quad (9)$$

В матричной форме это выглядит так:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \quad (10)$$

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

Другой способ используется, если известен вектор нормали к плоскости:

$$n = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}, \quad (11)$$

где $\mathbf{i, j, k}$ – единичные векторы осей x, y, z соответственно.

В частности, если компоненты этой точки на плоскости (x_1, y_1, z_1), то

$$d = -(ax_1 + by_1 + cz_1)d \quad (12)$$

- Удаление нелицевых граней тела:

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

Для определения невидимых граней: [E][V] отрицательные компонеты полученого 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

P1P2 – исследуемый отрезок. P(t) = P1 + (P2 - P1)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.

1.2.2 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).

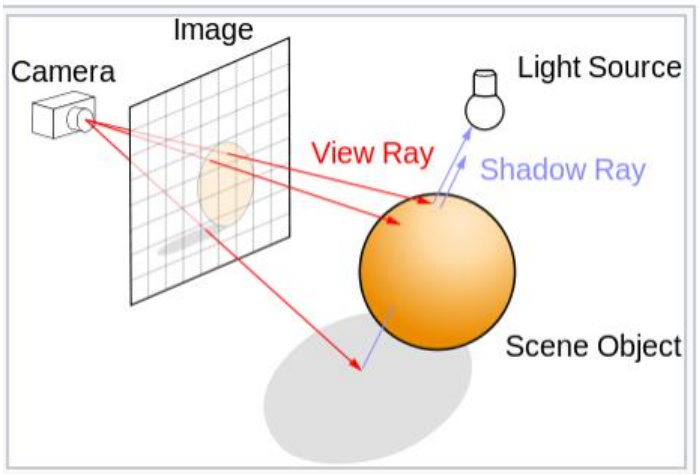


Рисунок 1.1. Алгоритм трассировки лучей.

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 изменения 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.

1.2.3 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:

$$Ax + By + Cz + D = 0 \qquad (13)$$

$$z = -\frac{Ax + By + D}{C}, \quad C \neq 0 \qquad (14)$$

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

У сканирующей строки $y = const$ и глубина пиксела на этой строке, у которого $x1 = x + \Delta x$, равна:

$$z1 - z = -\frac{ax1 + d}{c} + \frac{ax + d}{c} = \frac{a(x - x1)}{c} \qquad (15)$$

Так как $\Delta x = 1$, то $z1 = z - \frac{A}{C}$

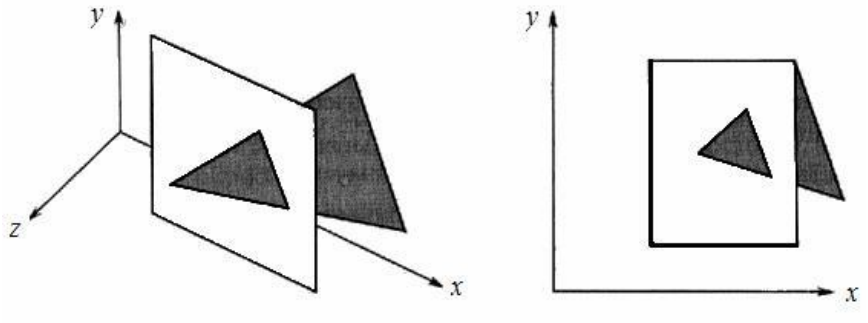


Рисунок 1.2. Пример алгоритма с использованием Z-буфера.

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

z(x, y) > Zбуфер(x, y) and z(x, y) <= Zсечения.

где Zсечения — 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-буффера. Этот алгоритм просто решает проблему пересечения сложных поверхностей.

1.3 Анализ алгоритмов закрашивания:

1.3.1 Однотонная закраска:

Это наиболее простой и требующий наименьших вычислительных ресурсов метод закраски 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:

$$I = I_0 cos\theta \qquad (16)$$

Где:

I_0 – интенсивность отражается в направлении нормали к поверхности.

θ – 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.

1.3.2 Метод Гуро:

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

Процесс закраски по методу Гуро осуществляется в четыре этапа:

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

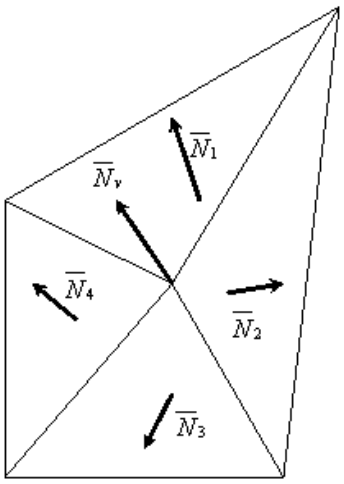


Рисунок 1.3. Определение нормалей.

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.

1.3.3 Метод Фонга:

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

Процесс закраски по методу Фонга осуществляется в четыре этапа:

1) Определяются нормали к граням.
2) По нормалям к граням определяются нормали в вершинах.
3) В каждой точке закрашиваемой грани определяется интерполированный вектор нормали.
4) 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 закраска по Фонгу и Гуро будет сложно и мешать ребра зданий будут сглажены. Поэтому лучше всего использовать тот же метод однотонной закраски.

1.4 Анализ алгоритм построения тени:

1.4.1 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" zбуфер, 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 "теневом" z-буфере для координат x', y', и значения z'. Если точка невидима для источника света (значение в "теневом" 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.

1.5 Анализ алгоритм визуализации туман:

Для того, чтобы создать эффект плотного тумана нужно знать глубину (расстояние от наблюдателя) видимого пикселя. Используя это значение можно вычислить интенсивность тумана для этого пикселя. Если z >= $z_{дальнее}$, то интенсивность тумана будет равна 1, иначе вычислить интенсивность по формуле:

$$k = 1 - \frac{z_{пикс} - z_{дальнее}}{z_{наблюдателя} - z_{дальнее}} \quad (17)$$

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

1.6 Анализ алгоритм визуализации дожди:

Система частиц

В системе частиц частицы (капли дождя) рассматриваются как материальные точки с разными атрибутами. Сама же система частиц — это совокупность всех частиц явления. Обычно все частицы в системе меняют скорость или размер по общему закону. Различная интенсивность осадков достигается за счет изменения общего количества частиц.

Для ускорения программы предполагают, что частицы не отбрасывают тени и не поглощаются свет.

1) Инициализация начальных данных (направления, интенсивность)
2) Пока не получена команда прекращения осадков:

- Обновление положения частиц по заданному закону
- Инициализация новых частиц
- Отображение частиц на дисплее

3) Пока система частиц не пуста:

- Обновление положения частиц по заданному закону
- Отображение частиц на дисплее

2. Конструкторская часть:

2.1 Требования к программе:

Программа должна предоставлять следующие возможности:

- Визуализация сцены объекта
- Добавление и удаление объекта в сцене
- Поворот сцены
- Изменение направление света
- Отображение погодных явлений (дождь и туман)

2.2 Общий алгоритм решения задачи:

1) Задать входные данные: объекты сцены (полигона), положения (векторы направления) источника цвета и наблюдателя
2) Для каждого полигона вычислить вектор нормали, интесивность цвета, найти внутренние пиксели.
3) Использовать алгоритм z – буфера для получения изображения сцены с тени:
   a) Вычислить и заполнить буфер глубины всех полигонов и изображение исходнной сцены
   b) Вычислить и заполнить другой буфер глубины для источника света
   c) Из двух буферов мы найдем затененные участки и запольнить его в окончательном изображении
4) Отобразить результат.
5) Отображение дожди и туман.

2.3 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. Част здания – крыша (она может отсутствовать, параметры задаются координатом центра основания и высотой) – это четырехугольная пирамида с основанием совпадает с верхним основанием здания.

- Источник света (Параметры задаются трехмерными координатами, цветом в виде RGB)
- Дождь, туман, солнце (Параметры задаются интенсивностью и направлением движения)

2.4 Простой метод освещения

В простом методе освещения интенсивность рассчитывается по закону Ламберта:

$$I = I_0 * \cos\alpha \qquad (18)$$

Где:

I_0: интенсивность источника света.

α: угол между нормалью к поверхности и вектором направления света.

2.5 Алгоритм удаления невидимых граней с использованием z – буффера:

1) Заполнить z-буфер элементами с фоновым значением цвета и минимальным значением z;
2) все точки системы частиц занести в z-буфер;
3) декомпозировать каждый многоугольник на треугольники:
   a. запомнить первую вершину;
   b. запомнить треугольник, который первая вершина образует с каждой следующей парой вершин;
4) Для каждого треугольника:
   a. найти *ymin* и *ymax*;
   b. пустить сканирующую строку от *ymin* до *ymax*;

c. для каждой сканирующей строки найти глубину *z* каждой точки *(x, y);*

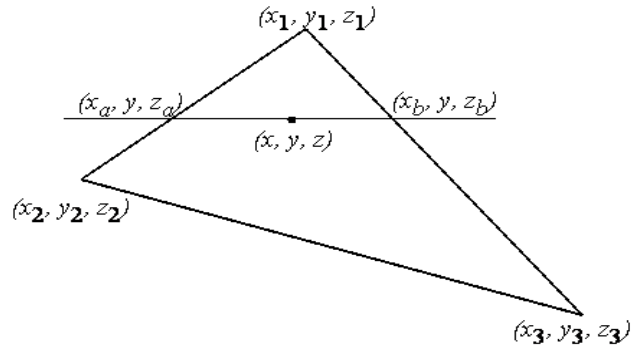


Рисунок 2.1. Сканирующая строка по грани.

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:

$$x_a = x_1 + (x_2 - x_1) * \frac{(y - y_1)}{(y_2 - y_1)}; \quad (19)$$

$$x_b = x_1 + (x_3 - x_1) * \frac{(y - y_1)}{(y_3 - y_1)}; \quad (20$$

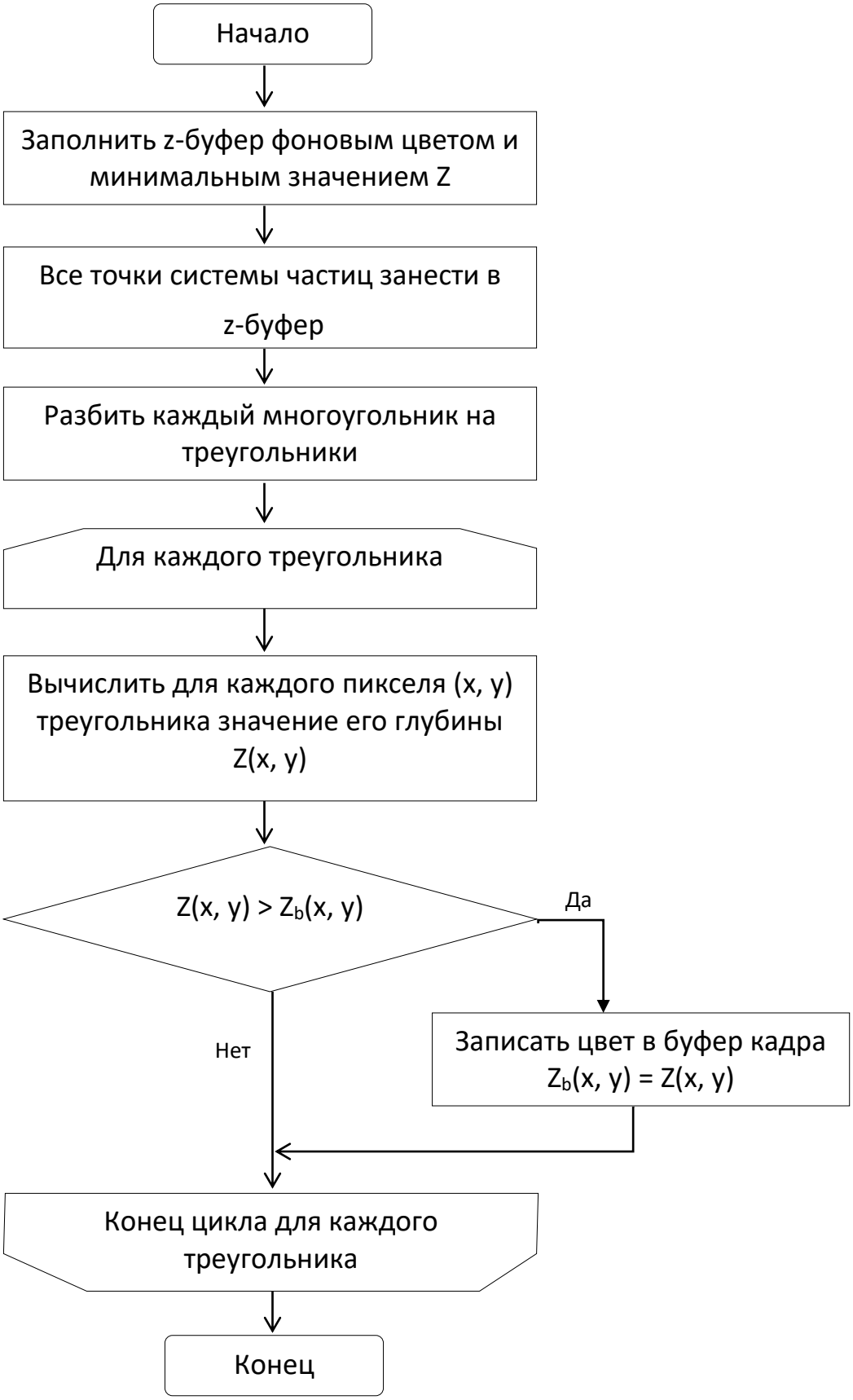
$$z_a = z_1 + (z_2 - z_1) * \frac{(y - y_1)}{(y_2 - y_1)}; \quad (21)$$

$$z_b = z_1 + (z_3 - z_1) * \frac{(y - y_1)}{(y_3 - y_1)}; \quad (22)$$

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 – буффера:



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 выполненя лабораторных работах 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 пользовании студентами, и предварительно установлен в свой машине.

3.2 Описание структуры программы:

Программа состоит из 6 модулей:

1. main: Главная точка входа в приложение, создать пользовательский интерфейс.

2. structures: Содержить описание структура сцены и состав классов.

- Вершины (составлять полигоны) – хранить координаты в трехмерном пространстве:

```
class vertex:
    def __init__(self, x = 0, y = 0, z = 0):
        self.x = x
        self.y = y
        self.z = z
```

- Полигон (составлять объекты) хранить информацию о вершинах, положении, цвет:

```
class polygon:
    def __init__(self, vertexs, locaiton, color):
```

self.vertexs = vertexs

self.location = location

self.color = color

- Обекты (Задания и земли) – хранить информации о объекте: его поверхности (полигоны), положение и цвет:

class object:

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

self.polygons = polygons

self.location = location

self.color = color

- Вектор источника света (направление и интенсивность):

class light:

def __init__(self, direction, intensity):

self.direction = direction

self.intensity = intensity

3. Action: действия: перенос, вопорот обекты

4. Scene: создать сцены

5. Zbuffer: алгоритм, использующий z буфер для удаления невидимых линий, поверхностей и построение тени.

- z-буффер:

class z_buffer:

def __init__(self, w, h, value):

self.buf = init_buf(w, h, value)

6. weather: изображает дождь и туман

3.3 Интерфейс:

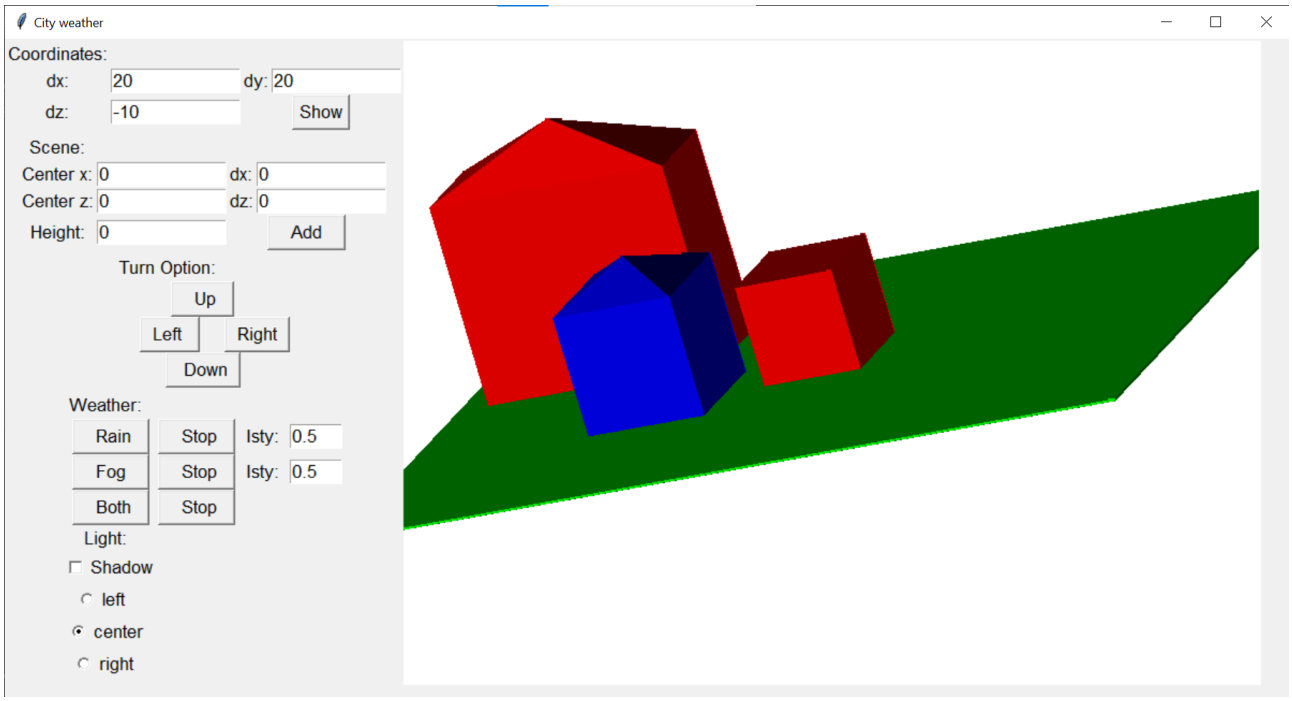


Рисунок 3.1. Интерфейс программы

Правая часть: Результат программы.

Левая часть: Панель управления программой, позволяющий изменять параметры:

- Scene (Сцена): возможность добавить и удалить объекты в сцене, поворот сцены в различных направлениях.
- Weather (Дождь, туман): позволяется визуализации дожди, тумана или обе на текущую сцену (возможность изменять направления и интенсивности).
- Light (Источник света): возможность изменять направление источника света.

4. Экспериментальная часть:

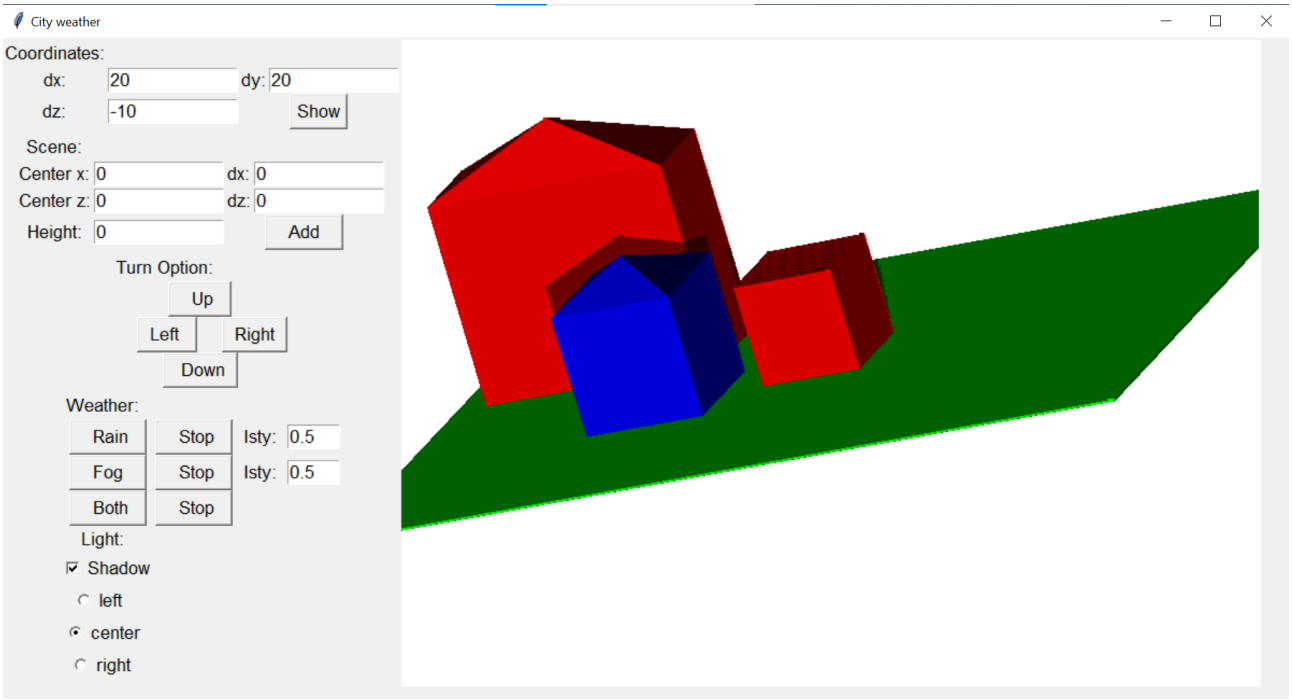


Рисунок 4.1. a) Визуализация вращающейся сцены с тенью при источник света в центре..

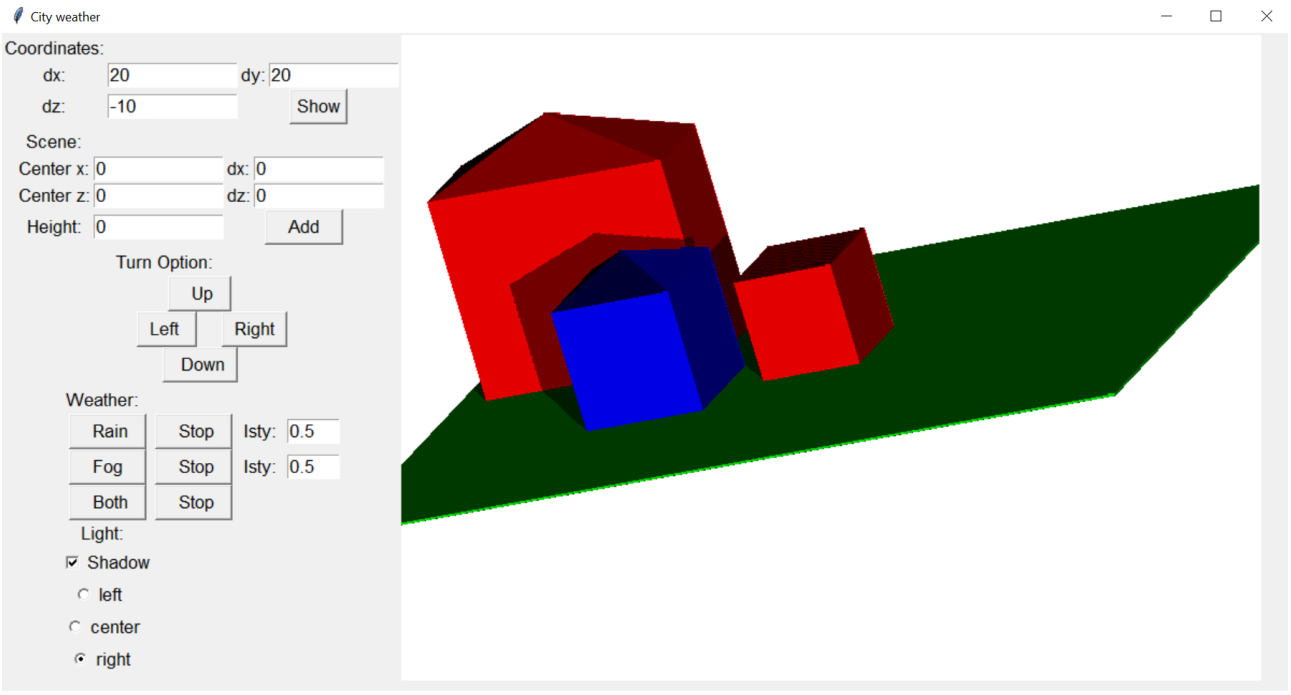


Рисунок 4.1. b) Визуализация сцены другого направления (при источник света в права).

В рисуке 4.1 a) и b) показан один и тот же городской ландшафт при разном положении источника. Очевидно что при смене источника света возникает изменение теней синего здания на стене большего красного здания, она наклоняется влево, потому что источник света меняется вправо.

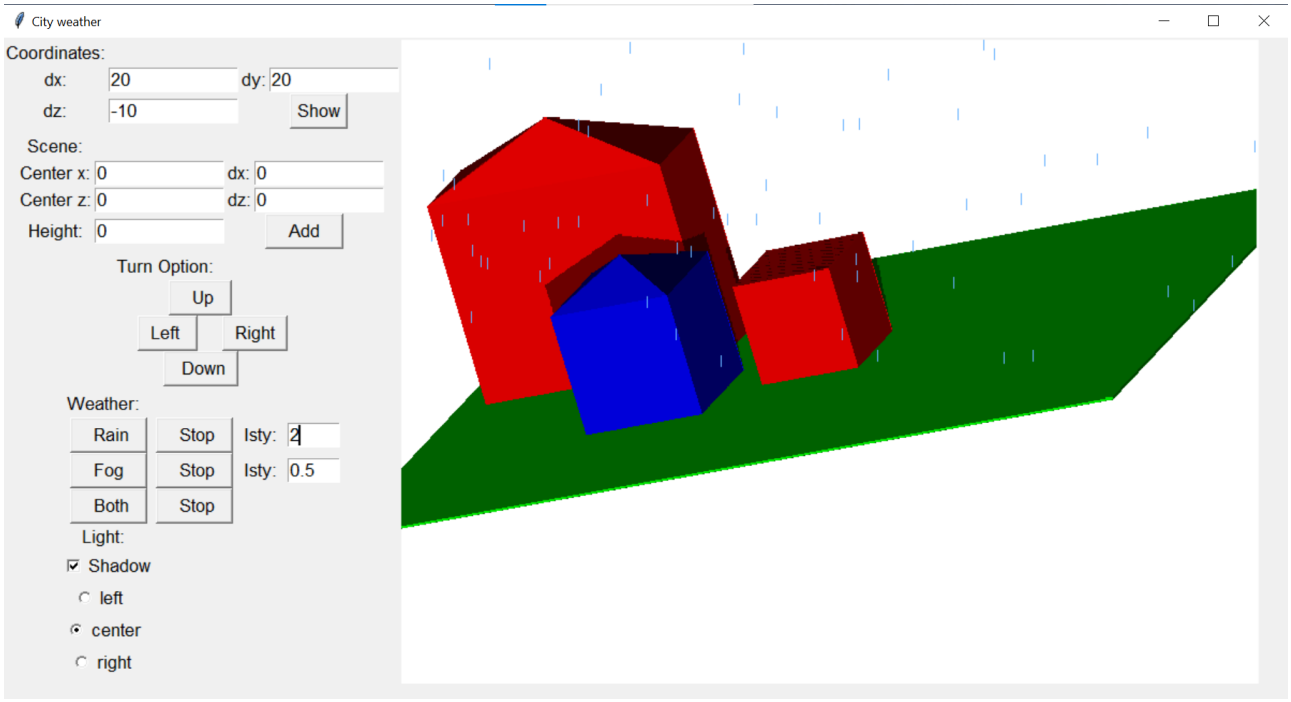


Рисунок 4.2. Визуализация сцены с эффектом дожди.

В рисуке 4.2 показан эффект дождя. Из рисунка видно, что эффект дождя работает корректно.

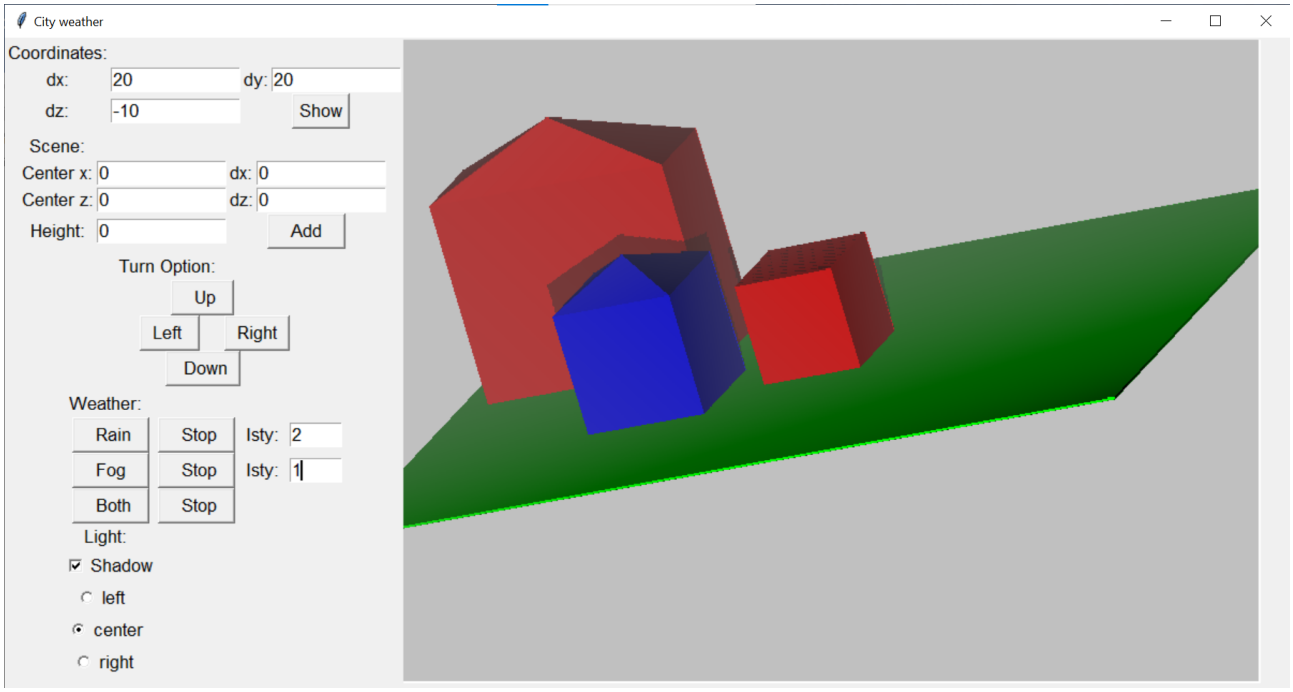


Рис. 4.3. Визуализация сцены с эффектом тумана.

В рисуке 4.3 показан эффект тумана. Из рисунке видим, что большое красное здание более размыты чем маленькое здание потому что оно дальше от точки зрения. Т.к чем ближе объект к точке наблюдателя, тем отчетливее он виден.

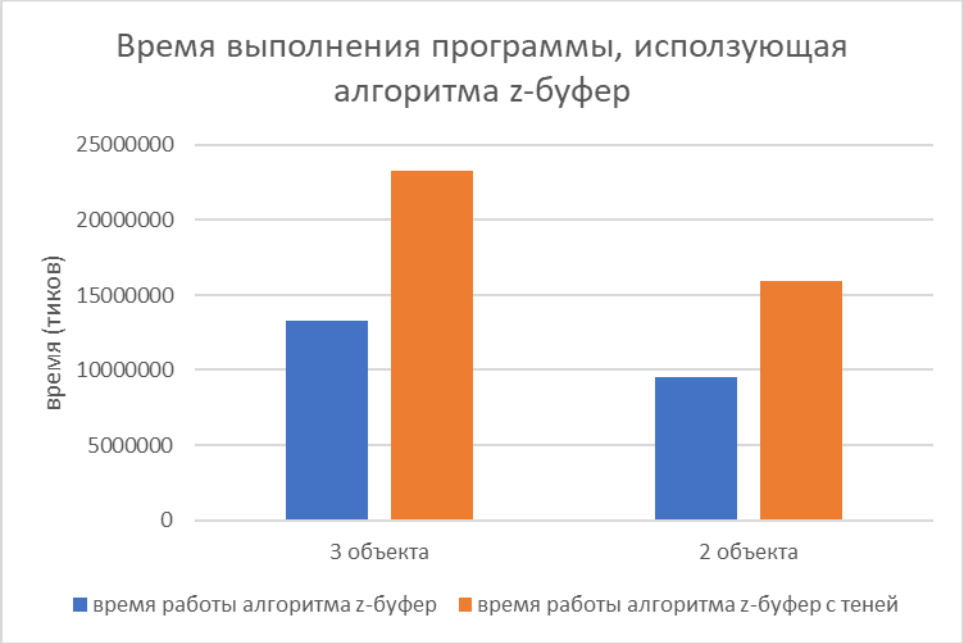


Рисунок 4.4 Время выполнения алгоритм z-буфер

В рисунке 4.4 показан время выполнения алгоритм z-буфер и алгоритм z-буфер с построением теней для сцены имеет 2 и 3 объекта. Мы видим, что время выполнения программы увеличивает на 2 раза при использовании алгоритм с построением теней. При увеличении количества объектов в сцене также значительно увеличивает время выполнения программы.

Заключение:

Во время выполнения проекта, были рассмотренны 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.

Список использованных источников

[1] Компьютерная графика – Мухин О.И. [Электронный ресурс]

Режим доступа: http://stratum.ac.ru/education/textbooks/kgrafic/contents.html

(Дата обращения 03.10.2020)

(использовано методы удаления невидимых линий: метод z-буфер)

[2] Основы компьютерная графика – [Электронный ресурс]

Режим доступа: https://studfile.net/preview/1751396/page:5/

(Дата обращения 10.10.2020)

(использовано алгоритм закрашивания: метод Гуро и метод Фонга)

[3] АЛГОРИТМЫ ПОСТРОЕНИЯ ТЕНЕЙ, А.Н.Романюк, М.В.Куринный [Электронный ресурс]

Режим доступа: https://core.ac.uk/download/pdf/52159835.pdf

(Дата обращения 10.10.2020)

(использовано алгоритм построения теней с использования алгоритма z-буфер)

[4] Лекционные записки по “Компьютерной Графике” МГТУ им. Баумана – А.В.Куров. [Текст], 2020

(использовано алгоритмы трехмерной преобразований)

[5] Компьютерная графика - А. Ю. Дёмин, А. В. Кудинов [Электронный ресурс]

Режим доступа: http://compgraph.tpu.ru/Del_hide_line.htm

(Дата обращения 10.10.2020)

(использовано алгоритм удаления невидимых линий и поверхностей: метод Робертса, метод трассировки лучей)