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

Студент_____Нгуен Санг Фыок_____
фамилия, имя, отчество

Группа ____ИУ7И-56Б_____

Название предприятия _____МГТУ им. Н. Э. Баумана, каф. ИУ7_____

Студент _____ ___ Нгуен С. Ф._____
подпись, дата *фамилия, и.о.*

Руководитель курсового проекта _____Филиппов М. В._____
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 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)**

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7
(Индекс)
_____ И.В.Рудаков
(И.О.Фамилия)
« ___ » _____ 2020 г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине_____Компьютерная графика_____

Трехмерная визуализация городской среды
(Тема курсового проекта)

Студент_____ Нгуен Ф. С. гр. ИУ7-56Б_____
(Фамилия, инициалы, индекс группы)

График выполнения проекта: 25% к _4 нед., 50% к _7_ нед., 75% к _11нед., 100% к 14 нед.

1. Техническое задание

Разработать программу моделирования трехмерной визуализации городской среды, состоит из зданий, деревьев и машины, движущейся по дороге. Анализ существующих методов и алгоритмов компьютерной графики. Источник света – солнце (т.е. бесконечно удалённый, в каждой точке сцены поток света имеет одинаковую интенсивность). Камера выберается по определенным положеням.

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 _____ Филиппов М. В.
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____ Нгуен Ф. С.
(Подпись,дата) (И.О.Фамилия)

Оглавление

Введение

Компьютерное моделирование требуется во многих областях жизнедеятельности человека. Создание разных моделей, строительство, дизайн, телевидение, кино, тренажеры для подготовки кадров, компьютерные игры - во всех этих сферах компьютерное моделирования стало необходимом атрибутом.

Трехмерное моделирование и анимации постоянно развивается и совершенствуется и предоставляет нам все большие возможности, чтобы реализовать нужные нам замыслы.

Целью проекта является реализовать построение трехмерной городской сцены.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

1. выбор и/или модифицирование существующих алгоритмов трехмерной графики, которые позволят визуализировать трехмерную сцену;
2. реализация данных алгоритмов для создания трехмерной сцены;
3. реализация анимации движения автомобиля.

1. Аналитическая часть

1.1.Анализ предметной области

1.1.1. Разработка трехмерной модели

Существует большое число способов описания поверхностей объектов визуализации моделируемого пространства.

В полигональных моделях используются данные следующих типов:

- геометрические (координаты вершин, уравнения ребер, плоскостей, поверхностей);
- топологические (данные, определяющие связи между вершинами и ребрами, ребрами и гранями, гранями и телами);
- вспомогательные (данные, передающие цвет, фактуру, освещенность объекта, прозрачность граней и т.п., которые используются для визуализации модели).

При использовании полигональной модели возможно применение операции удаления невидимых линий и поверхностей, а также визуальная проверка пересечений поверхностей.

Здание является правильной призмой с основанием, параллельным плоскости земли, и боковыми ребрами, перпендикулярными плоскости земли. Здание задается координатами положения центра основания на плоскости земли (x, y), высотой h.

Рисунок 1 Здание

Дерево состоит из цилиндра внизу и 3 четырехугольные пирамиды вверху. Дерево задается координатами (x, y), высотой цилиндра h, ширина (width) и длина (height)

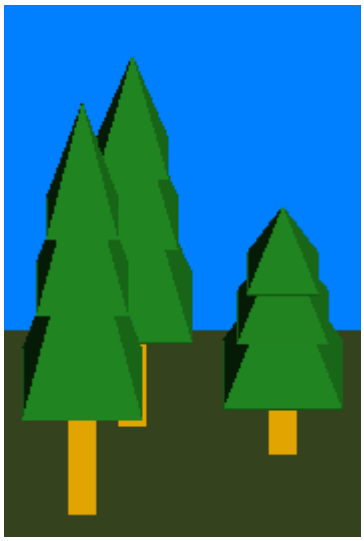


Рисунок 2 Дерево

Автомобиль: (разработано программой Blender) - сохраняется методом: вершины задаются координатами, а ребра задаются в порядке вершин

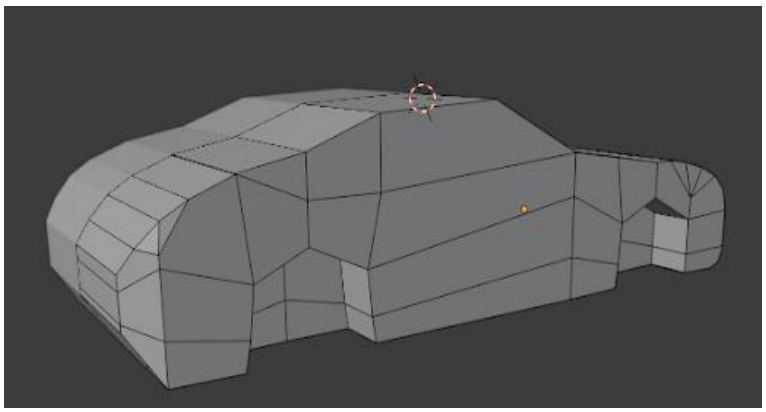


Рисунок 3 Автомобиль

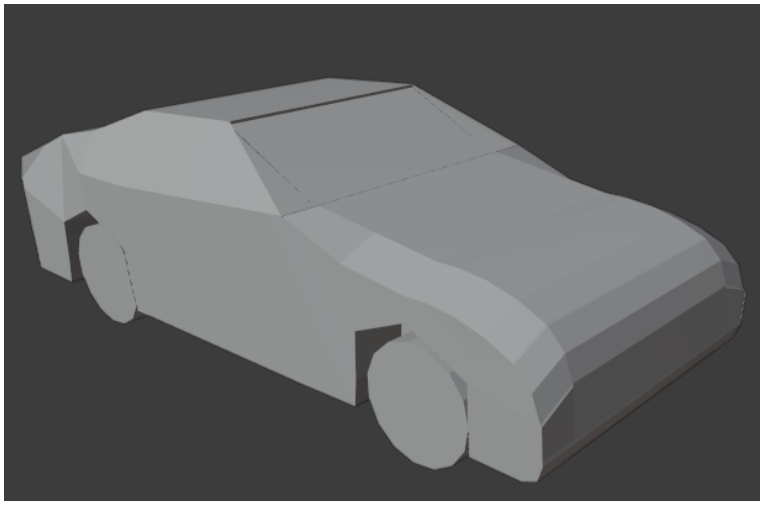


Рисунок 4 Автомобиль

1.2.Анализ алгоритмов удаления невидимых линий и поверхностей

1.2.1. Алгоритм обратной трассировки лучей

Алгоритм выглядит следующим образом: из виртуального глаза через каждый пиксел изображения испускается луч и находится точка его пересечения с поверхностью сцены.

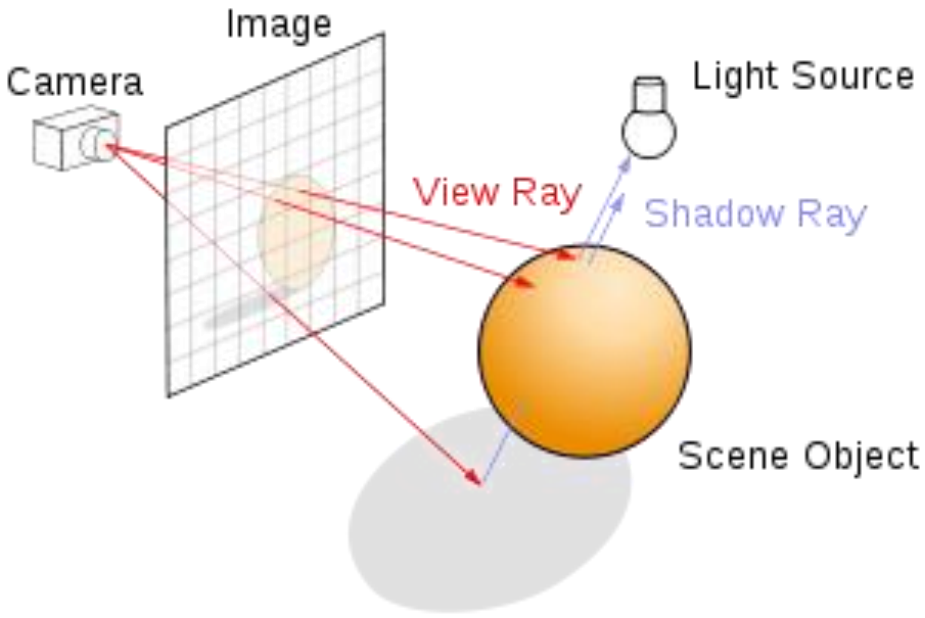


Рисунок 5 Алгоритм трассировки лучей.

Далее необходимо определить для каждого источника освещения, видна ли из него эта точка.

Алгоритм на псевдокоде можно кратко записать так:

```
for all pixels
    for all objects
        compare z
```

Преимущество: Высокая степень параллельности вычислений

Недостаток: высокая вычислительная стоимость расчетов; резкие границы цветовых переходов и “зазубренность” линий.

1.2.2. Алгоритм, использующий Z буфер

Алгоритм работает в пространстве изображения.

Идея *z*-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пиксела в пространстве изображения, z-буфер - это отдельный буфер глубины, используемый для запоминания координаты *z* или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение *z* каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен

в *z*-буфер. Если это сравнение показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер и, кроме того, производится корректировка *z*-буфера новым значением *z*. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по *x* и *y* наибольшего значения функции *z (x, y)*.

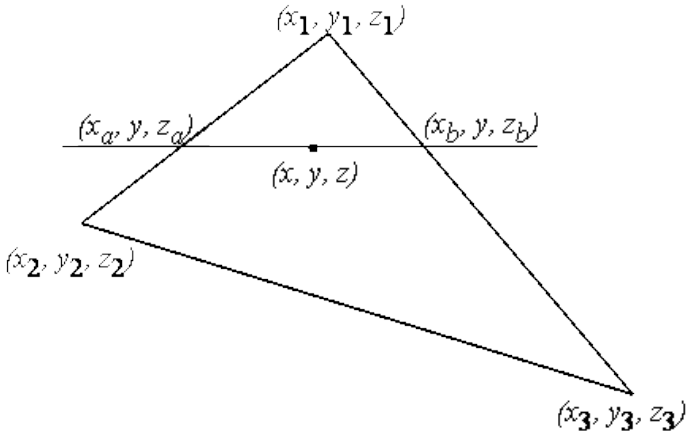


Рисунок 6 Линейная интерполяция

Для нахождения необходимых значений Z, используется линейная интерполяция:

$$z = z_a + (z_b - z_a)\frac{x - x_a}{x_b - x_a}$$

Главное преимущество - простота. Сцены могут быть любой сложности

Основной недостаток - большой объем требуемой памяти. Другой недостаток состоит в трудоемкости и высокой стоимости устранения лестничного эффекта, а также реализации эффектов прозрачности и просвечивания

1.2.3. Алгоритм Робертса

Алгоритм работает в объектном пространстве.

Алгоритм прежде всего удаляет из каждого тела те ребра или грани, которые экранируются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, экранируются этими телами.

В алгоритме Робертса требуется, чтобы все изображаемые тела или объекты были выпуклыми

Уравнение произвольной плоскости в трехмерном пространстве имеет вид:

$$ax + by + cz = 0 \qquad (1)$$

В матричной форме [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.

$$[V] = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{bmatrix} \qquad (2)$$

Матрицы тел для объектов преобразованной сцены можно получить или преобразованием исходных матриц тел, или вычислением новых матриц тел, используя преобразованные вершины или точки:

$$[VT] = [T]^{-1}[V] \qquad (3)$$

Матрица тела должно быть сформировано корректно то есть любая точка расположена внутри тела должна располагаться в положительную сторону от каждой грани тела. Если это не так, матрица корректируется умножением соответствующего столбца на -1.

Условие $[E]\cdot[V] < 0$ определяет, что плоскости — нелицевые, [E] = точка наблюдения.

Для определения видимых точек ребра надо построить луч соединяющии точку наблюдения с точкой на ребре. Точка невидима если луч на своем пути встречает в качестве прегради рассматриваеммое тело

Уравнение отрезка:

$$p(t) = P1 - (P2 - P1)t, \qquad 0 \le t \le 1 \tag{4}$$

Уравнение плоскости проходящий через отрезок и луч:

$$\mathrm{Q(t, al)} = \mathrm{P(t)} + \mathrm{al} * \mathrm{g}, \quad \mathrm{al} \ge 0 \tag{5}$$

Невидимым точкам ребра P1 P2 соответствуют такие значения параметров t, al при которых выполняется все неравенства

$$H = Q * V, \qquad hj > 0 \; j = 1..n \tag{6}$$

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.

Вывод:

Поскольку моя сцена статична (если машина не движется), большинство пикселей не меняются во время движения машины. Используя заархивированное изображение, можно сократить количество ненужных вычислений. Это означает, что при сохранении изображения со статическими объектами (зданиями и деревьями) по мере движения автомобиля будет отображаться только изображение автомобиля без дополнительных вычислений для статических объектов. Итак, наиболее подходящий - **алгоритм *z*-буфера**

1.3.Анализ методов закрашивания

1.3.1. Простая закраска

При однотонной закраске вычисляют один уровень интенсивности, который используется для закраски всего многоугольника.

Влияние состоит в том, что каждая из видимых полигональных граней аппроксимированной поверхности хорошо отличима от других, поскольку интенсивность каждой из этих граней отличается от интенсивности соседних граней. Различие в окраске соседних граней хорошо заметно вследствие эффекта полос Маха.

Для каждой плоскости вычислить вектор нормали так, чтобы: если плоскость сравнивается с солнцем, вектор нормали этой плоскости совпадает с направлением луча на угол менее 90 градусов.То есть вектор нормали направляющей плоскости не наружу, а внутрь.

Чтобы вычислить вектор нормали к плоскости, просматривайте вершины по часовой стрелке, вычисляйте средний вектор произведения следующих друг за другом векторов.

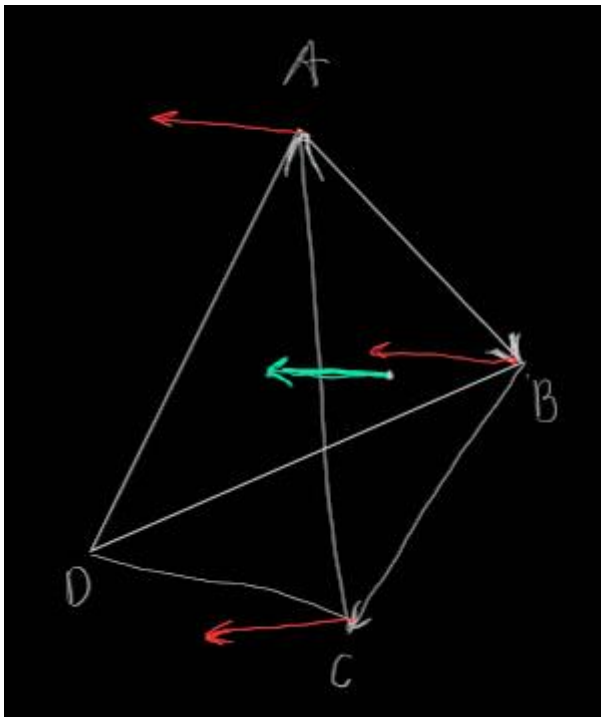


Рисунок 7 Вычисление вектора нормали к плоскости

Для каждой плоскости вычислить угол между вектором луча и нормали, если угол меньше 90 градусов, т.е. плоскость освещена, затем выделить с

соответствующей интенсивностью, наоборот, если угол больше 90 градусов, означает, что плоскость не освещена солнцем, выделите его более темной интенсивностью (не ноль из-за света из окружающей среды)

1.4.Анализ алгоритмов построения теней

Поскольку алгоритмы затенения и удаления скрытых поверхностей одинаковы, представляется возможным обрабатывать описание объекта, используя лишь один из этих алгоритмов, последовательно применяя его к точке зрения и к каждому из точечных источников света.

Для этого выполнить вращения последовательно так, чтобы вектор наблюдений (v = [0, 0, -1]) был в том же направлении, что и направление солнечных лучей.

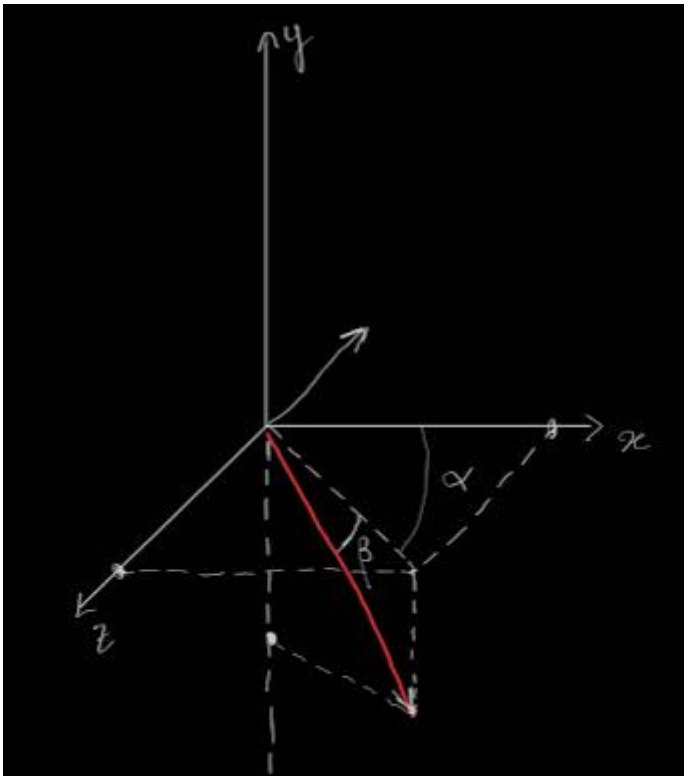


Рисунок 8 Направление луча света

Повернуть по часовой стрелке к оси Oy на угол (90 - альфа) градусов, затем поверните O на бета-градус. Теперь вектор наблюдения совпадает с направлением луча.

Применяем алгоритм Z-буфера, получаем видимую карту от солнца (bufferSun)

Затем для каждого пикселя найти соответствующую точку на карте, видимую со стороны солнца P (x, y, z). Сравните с z1 = bufferSun [x] [y], если z <z1, эта точка видна с солнца, она закрыта другим, поэтому этот пиксель является тенью. Напротив, z = z1, пиксель, видимый с солнца.

Если пиксель теневой, затемнить пиксель

1.5.Описание трехмерных преобразований сцены:

Перенос:

$$\begin{cases} x = x + dx \\ y = y + dy \\ z = z + dz \end{cases} \quad (7)$$

$$M_{\text{Перенос}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ dx & dy & dz & 1 \end{bmatrix} \quad (8)$$

Масштабирование:

KX, KY, KZ - коэффициенты масштабирования вдоль координатных осей

$$M_{\text{Масштаб}} = \begin{bmatrix} KX & 0 & 0 & 0 \\ 0 & KY & 0 & 0 \\ 0 & 0 & KZ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$\begin{cases} X1 = X * KX + (1 - KX) * X_M \\ Y1 = Y * KY + (1 - KY) * Y_M \\ Z1 = Z * KZ + (1 - KZ) * Z_M \end{cases} \quad (10)$$

Где (X,Y,Z) - координаты исходной точки,

(X1,Y1,Z1) - координаты промасштабированной точки,

(X_M, Y_M, Z_M) - центра масштабирования

Поворот (наблюдатель смотрит с конца положительной полуоси)

$$M_{\text{ПовZ}} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad (11)$$

$$M_{\text{ПовX}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

$$M_{\text{ПовY}} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

Если совершается произвольная последовательность поворотов вокруг осей X,Y,Z, то матрица преобразования будет иметь следующий вид:

$$M_{\text{Перенос}} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{24} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

1.6.Выводы из аналитического раздел:

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.

Так как здания состоят из плоскостей, закраска по Фонгу и Гуро будет скорее мешать: ребра зданий будут сглажены. Поэтому простая закраска хорошо подходит.

2. Конструкторская часть

2.1.Полигональное моделирование

Существует несколько способов 3Д моделирования, которые использует 3Д моделлер: полигональное, сплайновое и NURBS моделирование.

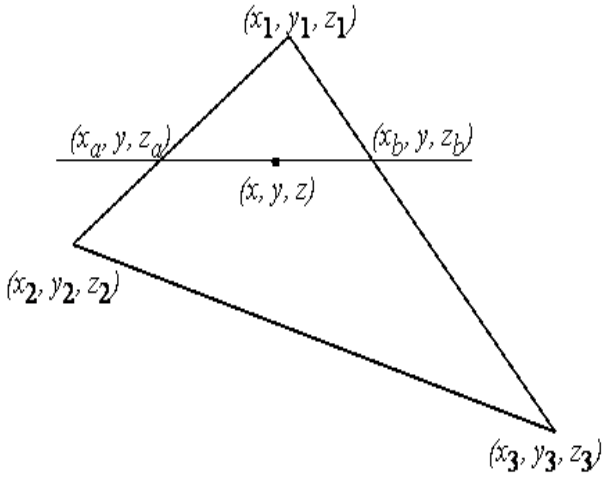
Из них самый подходящий способ для модели представления объектов – полигональное моделирование.

2.2.Алгоритм Z-буфера

Главное преимущество алгоритма – простота.

Описание алгоритма z-буфера:

1. Заполнить буфер кадра фоновым значением интенсивности или цвета.
2. Заполнить *z*-буфер минимальным значением *z*.
3. Выполнить растровую развертку каждого многоугольника сцены:
   i. Для каждого пикселя (x, y), связанного с многоугольником вычислить его глубину z(x, y)
   ii. Сравнить глубину z(x, y) со значением, хранимым в Z буфере. Если z(x, y) > zбуф(x, y), то записать атрибут этого пикселя в буфер кадра и заменить zбуф(x,y) =z(x,y).
4. Отобразить результат



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

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

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

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

$$z = z_a + (z_b - z_a)\frac{x - x_a}{x_b - x_a}$$

Рисунок 9 Сканирующая строка по грани

2.3.Простой метод освещения

Так как здания состоят из плоскостей, закраска по Фонгу и Гуро будет скорее мешать: ребра зданий будут сглажены. Поэтому простая закраска хорошо подходит.

В простом методе освещения интенсивность рассчитывается по закону Ламберта:

$$I = I_0 * \cos(\alpha) \qquad (15)$$

Где I – результирующая интенсивность света в точке

I_0 – интенсивность источника

α – угол между нормалью к поверхности и вектором направления света

2.4.Анимация движения автомобиля

Автомобиль едет прямо с повторением

Во время движения автомобиля сцена не меняется.

Для этого передаем автомобилю матрицу соответствующего преобразования

2.5.Выбор используемых типов и структур данных

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

- Источник света – направленностью света.
- Сцена – задается объектами сцены
- Объекты сцены – задаются вершинами и гранями.
- Математические абстракции
    - Точка – хранит координаты x, y, z
    - Вектор – хранит направление по x, y, z
- 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 Python т.к.:

- Это язык Python четкой формой, четкой структурой и кратким синтаксисом.
- Данный язык программирования объектно-ориентирован.

В качестве среды разработки была выбрана «Visual Studio 2017» т.к. она имеет множество удобств, которые облегчают процесс написания и отладки кода.s

3.2.Структура и состав классов

В этом разделе будут рассмотрена структура и состав классов

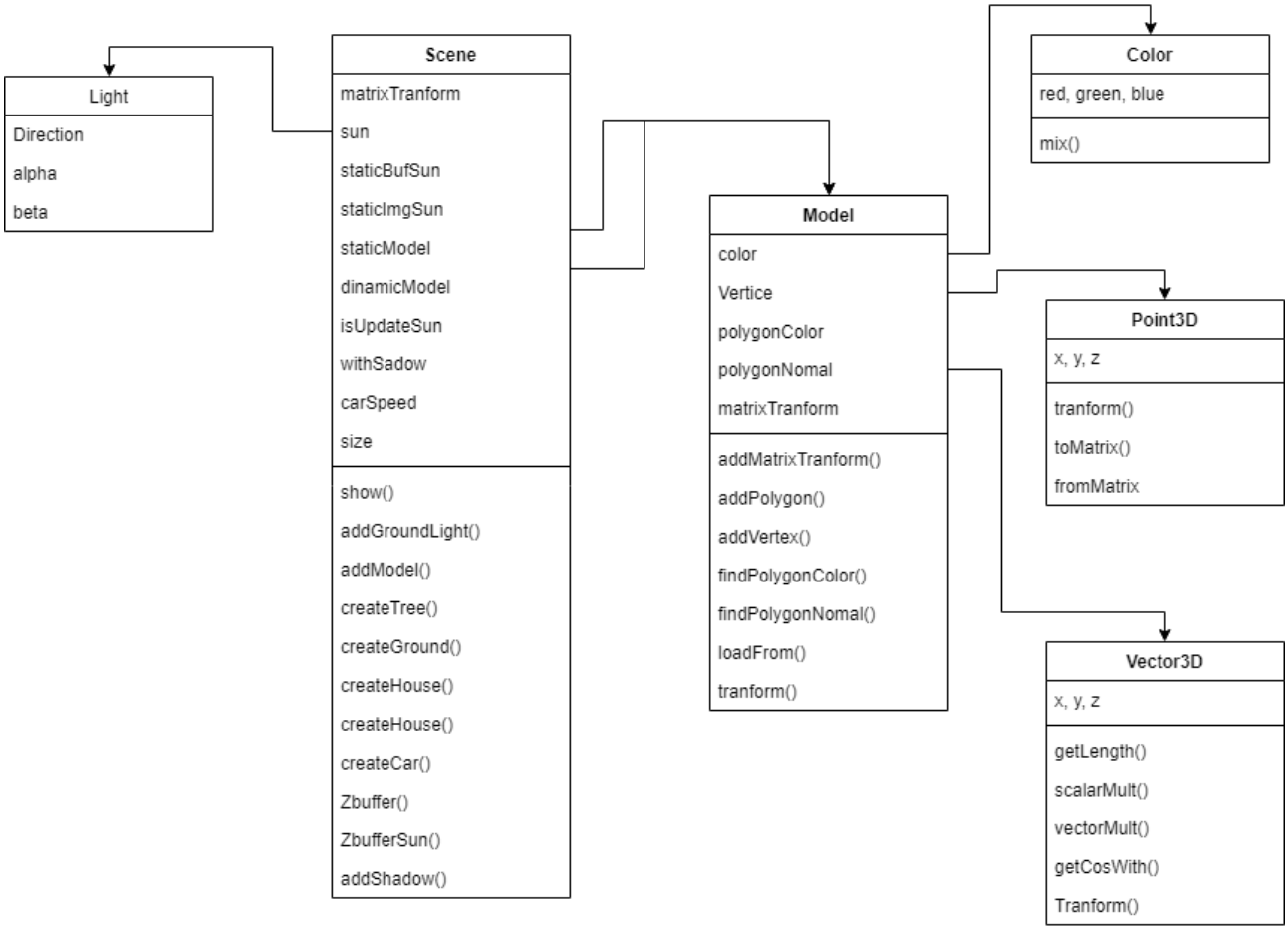


Рисунок 10 Структура классов

Point3D – хранить координат точки.

Vector – хранит направление вектора и его длину.

Model – хранит информацию о модели: ее цвет, вершины, многоугольники.

Light – класс источники света, хранить его интенсивность, цвет, направление.

Color – класс цвета.

Scene - класс сцена

3.3. Сведения о модулях программы

Main.py – главная точка входа в приложение;

Windown.ui – интерфейс;

Light.py – описание источников света;

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

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

Colors.py – взаимодействие цветов;

Transformation.py – функции преобразования координат;

Const.py - хранить другие константы

3.4.Интерфейс программы

Пользователь может выбрать направление источника света.

Программа позволяет выполнить операции поворота сцены в нужных направлениях.

Возможность изменять скорость и направление движения автомобиля (вперед и назад)

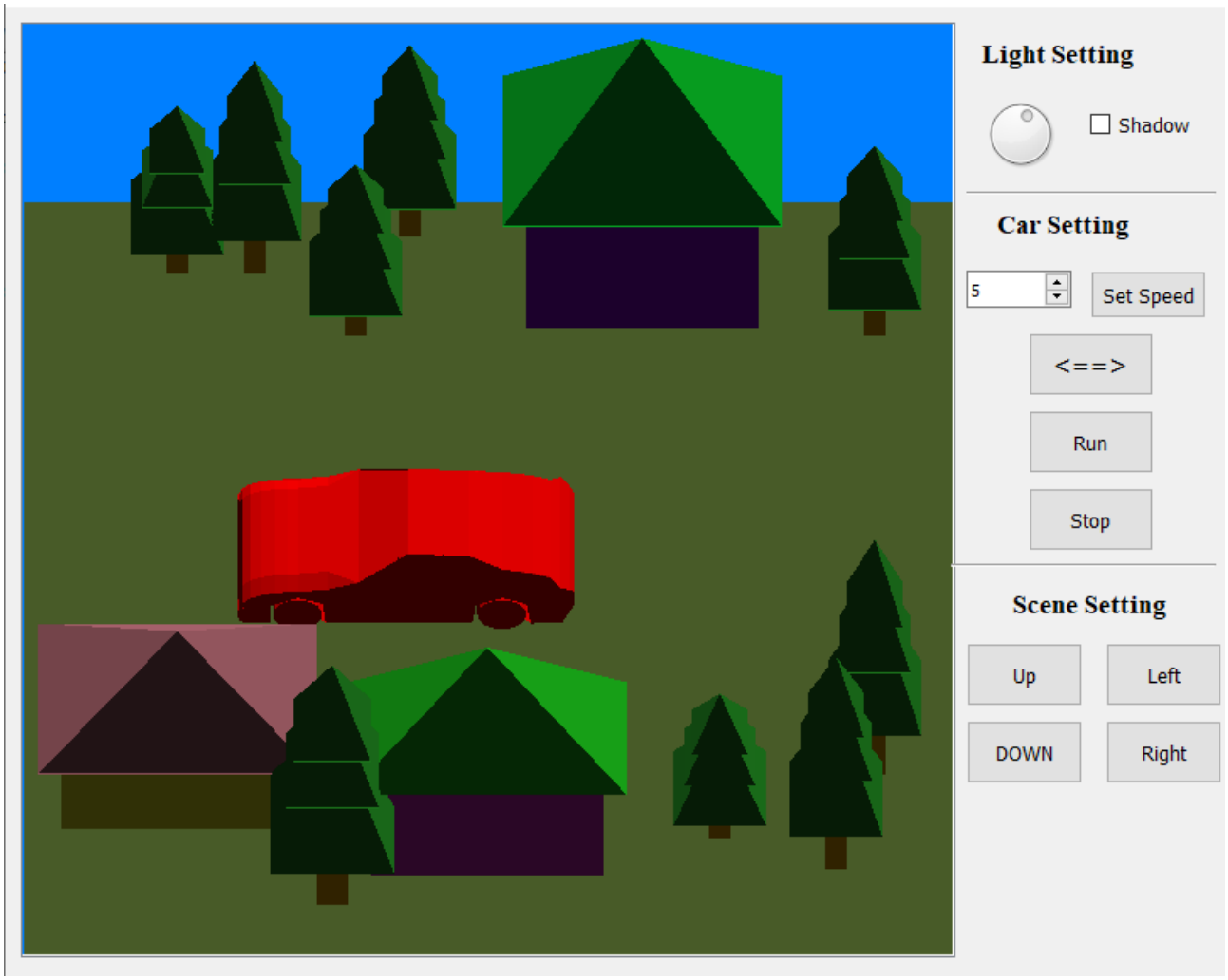


Рисунок 11Интерфейс программы

4. Экспериментальная часть

4.1.Цель эксперимента:

Целью эксперимента является проверка правильности выполнения поставленной задачи.

4.2. Апробация:

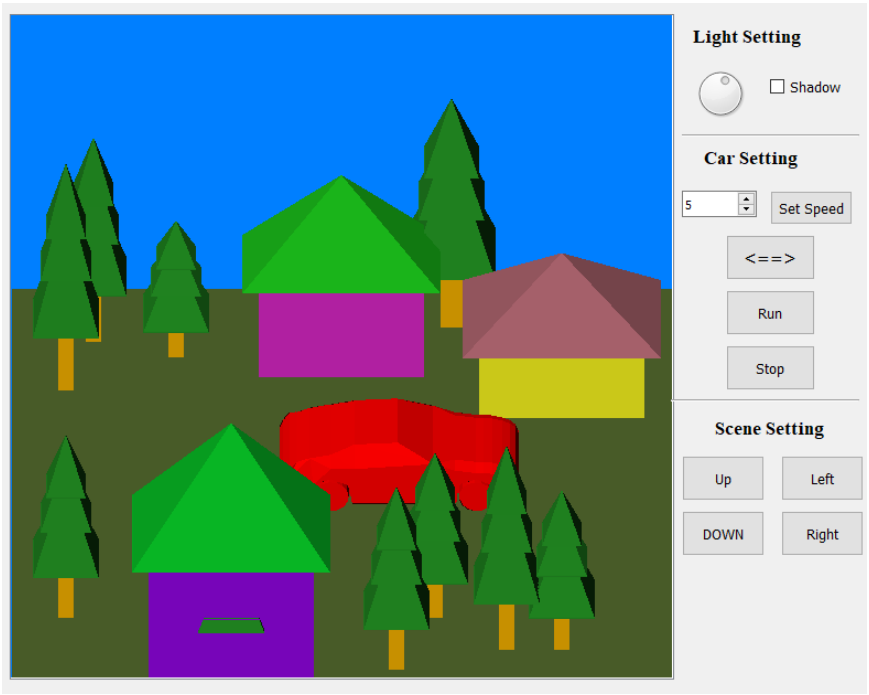


Рисунок 12 Сцена с разными источниками света

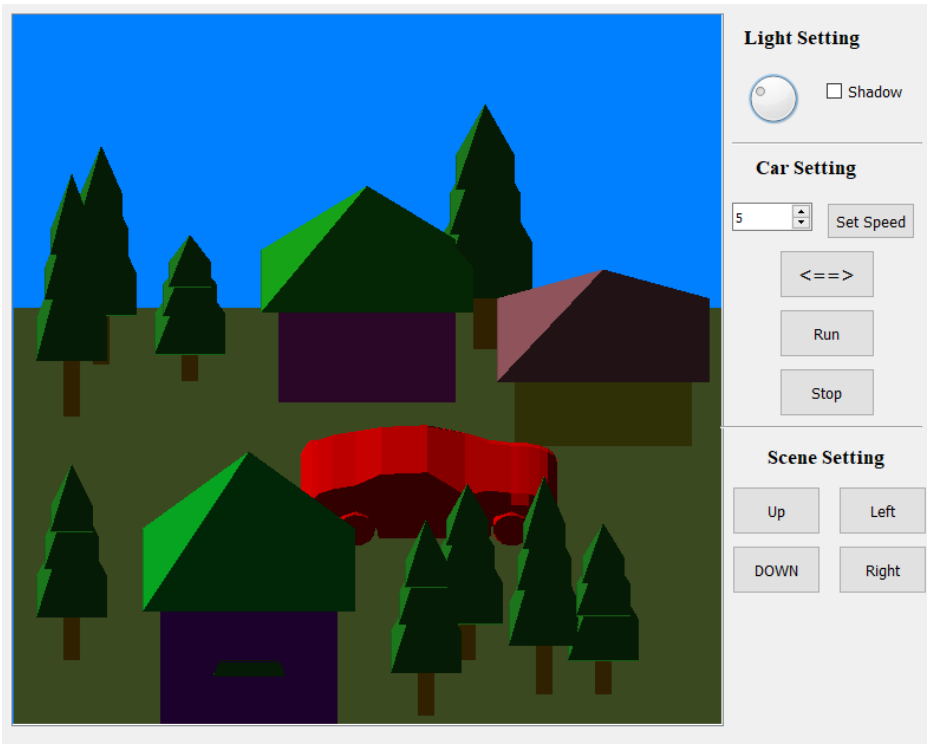


Рисунок 13 Сцена с разными источниками света

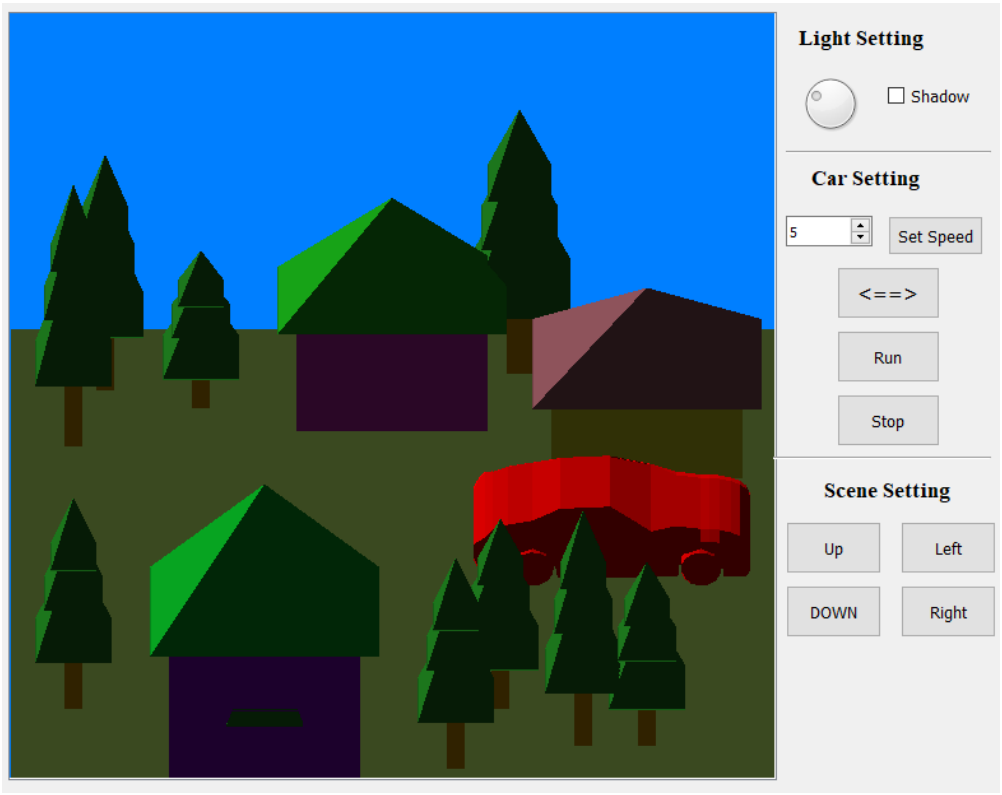


Рисунок 14 Машина движется

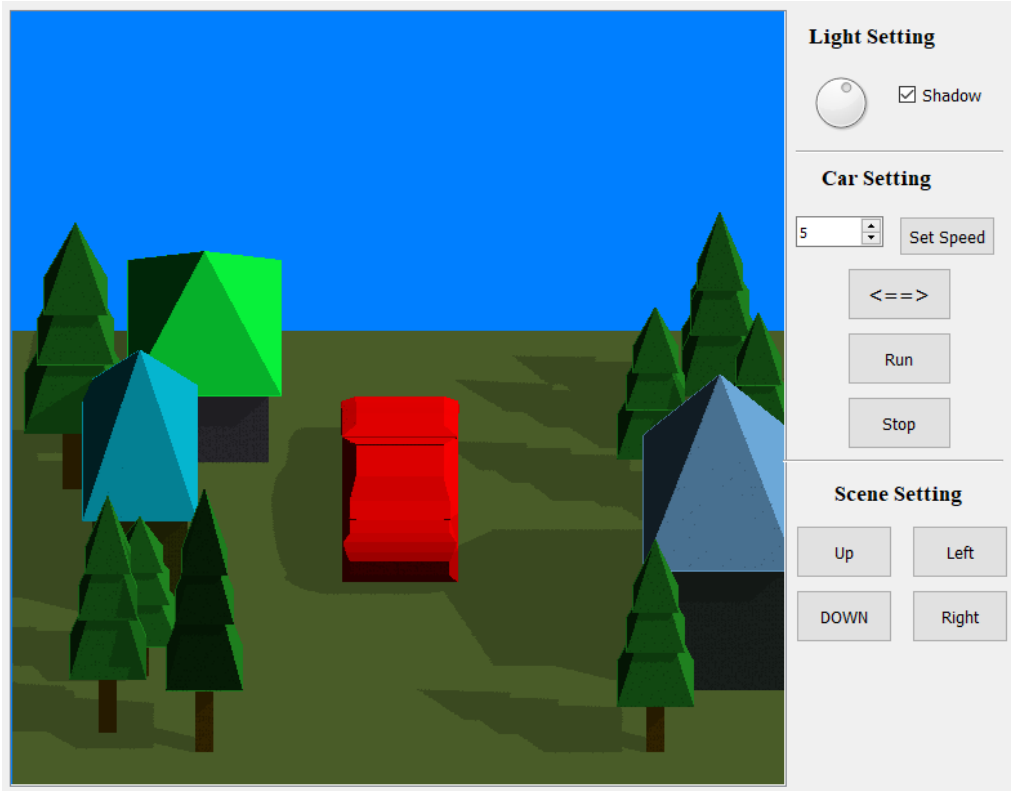


Рисунок 15 Сцена с тенью

Заключение

В ходе практики разработаны основные алгоритмы удаления невидимых линий, построения теней, методы закрашивания. Были проанализированы их достоинства и недостатки, выбраны наиболее подходящие для решения поставленной задачи.

Список использованной литературы

1. Компьютерная Графика - А. Ю. Дёмин, А. В. Кудинов. [http://compgraph.tpu.ru/index.html]
2. Computer Graphics [https://www.javatpoint.com/computer-graphics-tutorial]
3. Методы закрашивания в компьютерной графика (в Вьетнамском языке) [https://tailieu.vn/doc/cac-mo-hinh-to-mau-bong-trong-do-hoa-may-tinh-198113.html]