|  |  |
| --- | --- |
| 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 Чыонг Нгуен Вьет Уи

*фамилия, имя, отчество*

Группа ИУ7И-42Б

Тип практики производственная

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

Студент

Руководитель практики

Чыонг Н.В.У

*подпись, дата фамилия, и.о.*

Куров А.В.

*подпись, дата фамилия, и.о.*

Оценка

*Москва, 2021 г.*

**Индивидуальное задание:**

Разработать программу для трехмерной визуализации городской среды. Реализация анимации движения машины по дороге с повышением быстродействия генерации изображения.

/технологическая/производственная

Оглавление

[Введение 4](#_Toc87397873)

[1. Аналитическая часть 5](#_Toc87397874)

[1.1. Анализ предметной области 5](#_Toc87397875)

[1.1.1. Разработка трехмерной модели 5](#_Toc87397876)

[1.1.2. Преобразование модели, синтезируемой в пространстве, к координатам наблюдателя 8](#_Toc87397877)

[1.1.3. Отсечение объектов визуального пространства в пределах пирамиды видимости. 9](#_Toc87397878)

[1.1.4. Вычисление двумерных перспективных проекций синтезируемых объектов видимости на картинную плоскость. 11](#_Toc87397879)

[1.2. Анализ алгоритмов удаления невидимых линий и поверхностей 12](#_Toc87397880)

[1.2.1. Алгоритм обратной трассировки лучей 12](#_Toc87397881)

[1.2.2. Алгоритм, использующий Z буфер 13](#_Toc87397882)

[1.2.3 Алгоритм, использующий Z буфер 14](#_Toc87397883)

[1.3. Анализ методов закрашивания 17](#_Toc87397884)

[1.3.1. Простая закраска 17](#_Toc87397885)

[1.3.2. Закраска по Гуро 18](#_Toc87397886)

[1.3.3. Закраска по Фонгу 20](#_Toc87397887)

[1.4. Анализ алгоритмов построения теней 21](#_Toc87397888)

[1.5. Выводы из аналитического раздел 22](#_Toc87397889)

[2. Конструкторская часть 23](#_Toc87397890)

[2.1. Алгоритм Z-буфера 23](#_Toc87397891)

[2.2. Простой метод освещения 24](#_Toc87397892)

[2.3. Анимация движения автомобиля 24](#_Toc87397893)

[2.4. Выбор используемых типов и структур данных 24](#_Toc87397894)

[3. Технологическая часть 25](#_Toc87397895)

[3.1. Выбор и обоснование языка программирования и среды разработки 25](#_Toc87397896)

[3.2. Структура и состав классов 25](#_Toc87397897)

[3.3. Сведения о модулях программы 26](#_Toc87397898)

[3.4. Интерфейс программы 26](#_Toc87397899)

[Заключение 27](#_Toc87397900)

[Список использованной литературы 28](#_Toc87397901)

[Приложения 29](#_Toc87397902)

# 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.

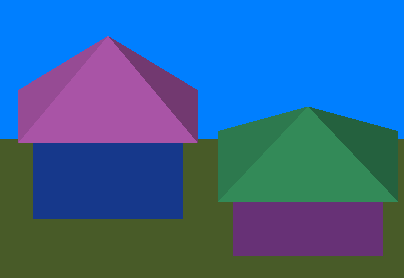
Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

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

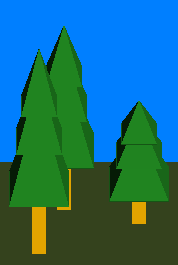
# Аналитическая часть

* 1. **Анализ предметной области**
     1. **Разработка трехмерной модели**

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.

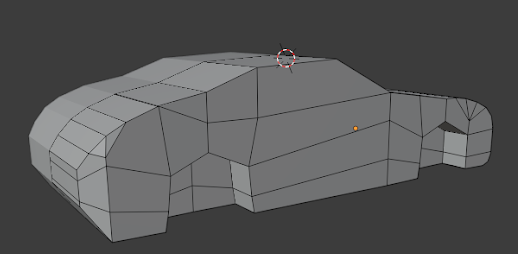


*Рисунок 1. Здание*

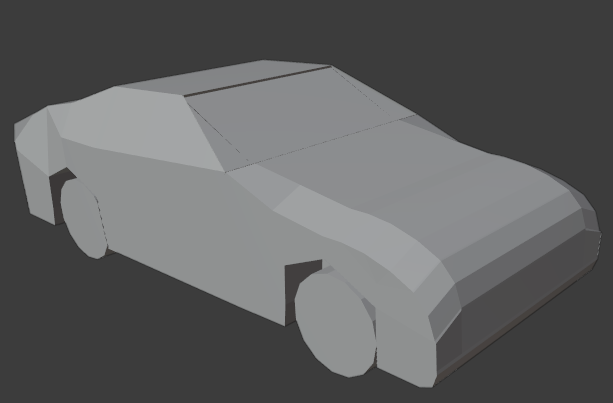
****Дерево состоит из цилиндрa внизу и сфер вверху. Дерево задается координатами (x, y), высотой цилиндрa h, радиусом сферы R.

*Рисунок 2. Дерево*

Автомобиль состоит из двух прямоугольных ящиков (большого и малого) с вырезом для колес.



*Рисунок 3. Автомобиль*



*Рисунок 4. Автомобиль*

Существует большое число способов описания поверхностей объектов визуализации моделируемого пространства. Из них самый подходящий способ для модели представления объектов – полигональное моделирование.

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).

Сферная (cложная) поверхность объекта может быть аппроксимирована многогранником, каждая грань которого является простейшим плоским многоугольником (треугольник, четырехугольник).

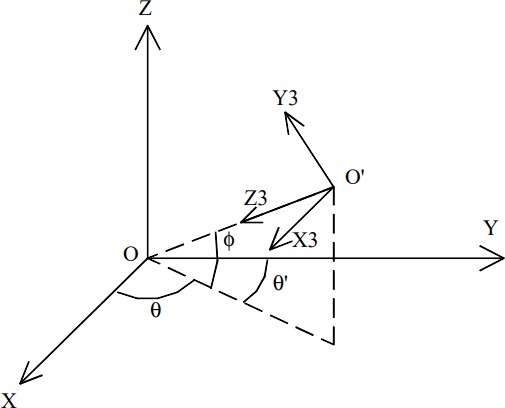
* + достоинствo - простотa математических методов обработки плоскостей;
  + недостатoк – прямая зависимость достоверности вида поверхностей объекта от количества граней, используемых для аппроксимации каждой из поверхностей модели.

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

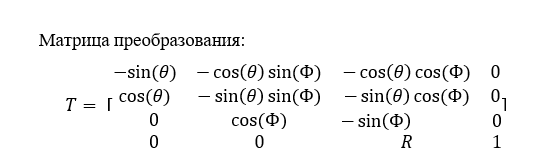
* + 1. **Преобразование модели, синтезируемой в пространстве, к координатам наблюдателя**

Объект находится в систему (O, X, Y, Z).

Глаз находится в началo координат системы (O', X0, Y0, Z0).

* + - 1. Перевод точки O в O', система (X, Y, Z) переходит в систему (X1, Y1, Z1).
      2. Повернуть систему (X1, Y1, Z1) на угол -θ' (θ' = 90 - θ) вокруг оси Z1 по часовой стрелке, система (X1, Y1, Z1) переходит в систему (X2, Y2, Z2).
      3. Повернуть систему (X2, Y2, Z2) на угол 90 + Φ вокруг оси X2. Это преобразование будет заставляет ось Z2 указывать на начало координат O.

*Рисунок 5. Преобразование координат системы*



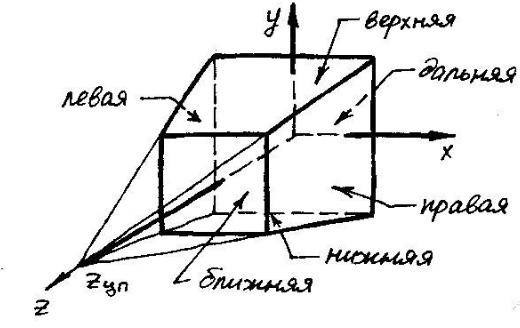
* + 1. **Отсечение объектов визуального пространства в пределах пирамиды видимости.**

Как и в двумерном отсечении, здесь также можно использовать обобщенные коды концевых точек, предложенные Коэном и Сазерлендом.

Единица в соответствующий бит заносится в следующих случаях:

* в первый бит, если точка находится левее отсекателя (левее левой грани);
* во второй бит, если точка находится правее отсекателя (правее правой грани);
* в третий бит, если точка находится ниже отсекателя (ниже нижней грани);
* в четвертый бит, если точка находится выше отсекателя (выше верхней грани);
* в пятый бит, если точка находится ближе отсекателя (перед передней гранью);
* в шестой бит, если точка находится дальше отсекателя (за дальней гранью).

Для составления кода используются пробные функции:



*Рисунок 6. Отсечение объектов*

Уравнение прямой на плоскости xz, несущей проекцию правой грани отсекателя, имеет вид:

𝑥 = (𝑧 − 𝑧цп)𝑥п = 𝛼

𝑧 + 𝛼

, 𝛼

= 𝑥п

, 𝛼

= −𝛼 𝑧

𝑧д − 𝑧цп

1 2 1

𝑧д − 𝑧цп 2

1 цп

Подстановка координат x и z точки P в пробную функцию правой грани fп даст следующий результат:

* 0, если Р справа от плоскости

𝑓п = 𝑥 − 𝛼1𝑧 − 𝛼2 { = 0, если Р на плоскости

< 0, если Р слева от плоскости

* 0, если Р слева от плоскости

𝑓л = 𝑥 − 𝛽1𝑧 − 𝛽2 {

= 0, если Р на плоскости

< 0, если Р справа от плоскости

* + 0, если Р выше плоскости

𝑓в = 𝑦 − 𝛾1𝑧 − 𝛾2 {= 0, если Р на плоскости

< 0, если Р ниже плоскости

* + 0, если Р ниже плоскости

𝑓н = 𝑥 − 𝛼1𝑧 − 𝛼2 {= 0, если Р на плоскости

< 0, если Р выше плоскости

* 0, если Р ближе плоскости

𝑓б = 𝑧 − 𝑧б {

= 0, если Р на плоскости

< 0, если Р дальше плоскости

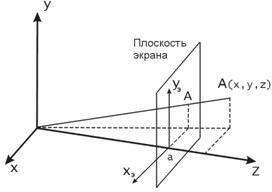
* 0, если Р дальше плоскости

𝑓д = 𝑧 − 𝑧д { = 0, если Р на плоскости

< 0, если Р ближе плоскости

* + 1. **Вычисление двумерных перспективных проекций синтезируемых объектов видимости на картинную плоскость.**

*Параллельные проекции -* это такие проекции, у которой все линии проецирования параллельны друг другу.

 aaaa*Перспективные проекции*

*-* это такие проекции, у которых все линии проецирования сходятся в одной точке - центре проецирования (или точки наблюдения). Эти проекции наиболее полно отражают специфику человеческого зрения.

*Рисунок 7. Перспективные проекции*

Точка A проецируется на экран как A’.

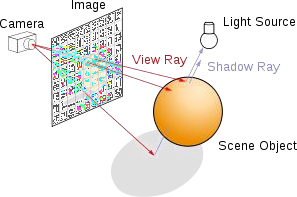
Если точку наблюдения поместить в начало координат, а проекционную плоскость на расстояние a:

*x’ =*

*y’ =*

* 1. **Анализ алгоритмов удаления невидимых линий и поверхностей**
     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 8. 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*

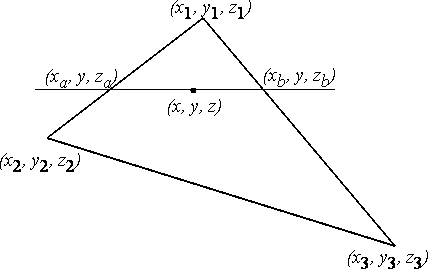
Преимущество: Высокая степень параллельности вычислений

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 *х* и *у* наибольшего значения функции *z (х, у).*

Для нахождения необходимых значений Z, используется линейная интерполяция**:**

*Рисунок 9. Линейная интерполяция*

𝑧 = 𝑧𝑎

+ (𝑧𝑏

− 𝑧𝑎

) 𝑥 − 𝑥𝑎

𝑥𝑏 − 𝑥𝑎

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, использующий Z буфер**

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:

**aх + by + cz + d = 0**

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.

𝑎1 𝑎2 [𝑉] = [𝑏1 𝑏2

𝑐 𝑐

1 2

𝑑1 𝑑2

… 𝑎𝑛

… 𝑏𝑛

]

… 𝑐𝑛

… 𝑑𝑛

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]

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 [*Е*]·[*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 преграде рассматриваеммое тело

Уравнение отрезка:

𝑝(𝑡) = 𝑃1 − (𝑃2 − 𝑃1)𝑡, 0 ≤ 𝑡 ≤ 1

Уравнение плоскости проходящий через отрезок и луч:

Q(t, al) = P(t) + al ∗ g, al ≥ 0

Невидимым точкам ребра Р1 Р2 соответствуют такие значения параметров t, al при которых выполняется все неравенства

𝐻 = 𝑄 ∗ 𝑉, ℎ𝑗 > 0 𝑗 = 1. . 𝑛

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.

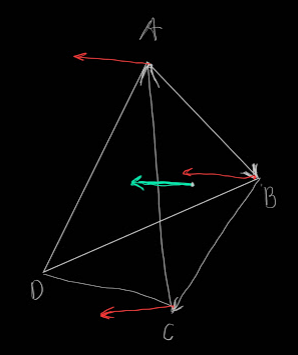
Некоторые из оптимизаций крайне сложны, что затрудняет реализацию этого алгоритма.

**1.3. Анализ методов закрашивания**

* + 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.



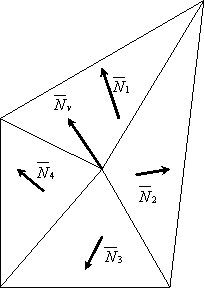
*Рисунок 10. Bычисление вектора нормали к плоскости*

* + 1. **Закраска по Гуро**

Метод Гуро основан на интерполяции интенсивности, позволяет устранить дискретность изменения интенсивности и создается иллюзия гладкой криволинейной поверхности.

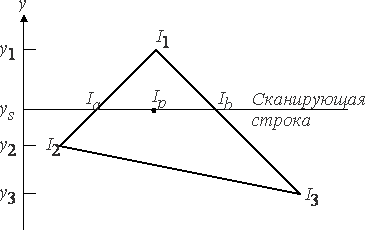
Процесс закраски по методу Гуро осуществляется в четыре этапа:

1. Вычисляются нормали ко всем полигонам.

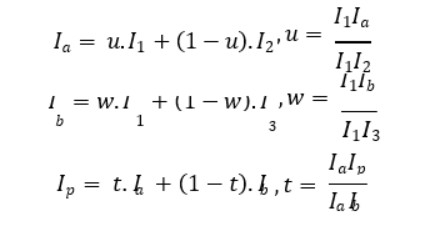


*Рисунок 11. Нормали к вершинам*

1. Определяются нормали в вершинах путем усреднения нормалей по всем полигональным граням, которым принадлежит вершина.
2. Используя нормали в вершинах и применяя произвольный метод закраски, вычисляются значения интенсивности в вершинах.
3. Каждый многоугольник закрашивается путем линейной интерполяции значений интенсивностей в вершинах сначала вдоль каждого ребра, а затем и между ребрами вдоль каждой сканирующей строки.



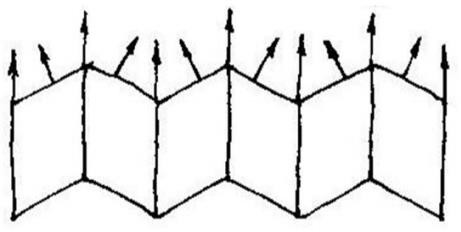
*Рисунок 12. Интерполяция интенсивностей*



Закраска по Гуро хорошо сочетается с диффузным отражением. Данный метод интерполяции обеспечивает лишь непрерывность значений интенсивности вдоль границ многоугольников, но не обесиечивает

непрерывноси изменения интенсивности. Значит, возможно проявление полос Маха.

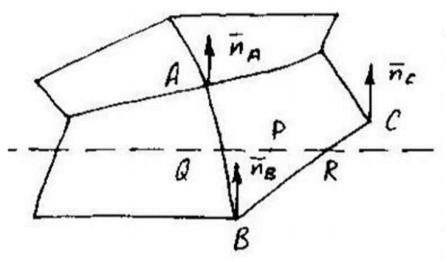
Недостаток: усреднение нормалей. Поверхность закрашивается с одной интенсивностью. Будет выглядеть плоской.



*Рисунок 13. Поверхность закрашивается с одной интенсивностью*

* + 1. **Закраска по Фонгу**

В методе закраски по Фонгу, используется интерполяция вектора нормали к поверхности вдоль видимого интервала на сканирующей строке внутри многоугольника, а не интерполяция интенсивности.

𝑛𝑄

= 𝑢. 𝑛𝐴

+ (1 − 𝑢)𝑛𝐵

, 𝑢 = 𝐴𝑄

𝐴𝐵

𝐶𝑅

*Рисунок 14. Закраска по Фонгу*

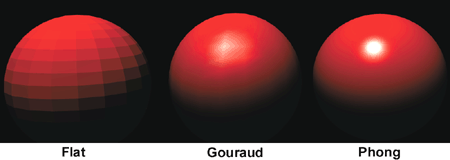
𝑛𝑅 = 𝑤. 𝑛𝐶 + (1 − 𝑤)𝑛𝐵, 𝑢 = 𝐶𝐵

𝑄𝑃

𝑛𝑃 = 𝑡. 𝑛𝑄 + (1 − 𝑡)𝑛𝑅, 𝑡 = 𝑄𝑅

Преимущества: Достигается лучшая локальная аппроксимация кривизны поверхности. Изображение выходит более реалистичным, зеркальные блики выглядят правдоподобнее, чем в методе закраски по Гуро.

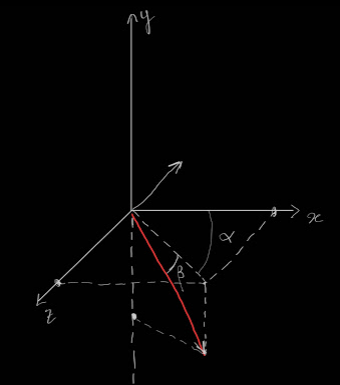
Недостаток: закраска по Фонгу требует больших вычислительных затрат



*Рисунок 1.2.3-2. Сравнение методов закрашивания*

## 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.



## 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.

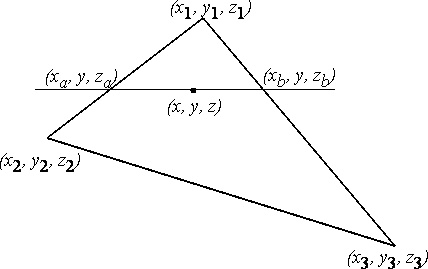
Так как здания состоят из плоскостей, закраска по Фонгу и Гуро будет скорее мешать: ребра зданий будут сглажены. Поэтому простая закраска хорошо подходит.

# Конструкторская часть

* 1. **Алгоритм Z-буфера**

Главное преимущество алгоритма – простота. Oписание 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:
   1. Для каждого пикселя (x, y), связанного с многоугольником вычислить его глубину z(x, y)
   2. 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

𝑥𝑎

𝑥𝑏

= 𝑥1

= 𝑥1

+ (𝑥2

+ (𝑥3

− 𝑥1

− 𝑥1

) 𝑦 − 𝑦1

𝑦2 − 𝑦1

) 𝑦 − 𝑦1

𝑦3 − 𝑦1

𝑧𝑎

𝑧𝑏

= 𝑧1

= 𝑧1

+ (𝑧2

+ (𝑧3

− 𝑧1

− 𝑧1

) 𝑦 − 𝑦1

𝑦2 − 𝑦1

) 𝑦 − 𝑦1

𝑦3 − 𝑦1

*Рисунок 15. Сканирующая строка по грани*

𝑧 = 𝑧𝑎

+ (𝑧𝑏

− 𝑧𝑎

) 𝑥 − 𝑥𝑎

𝑥𝑏 − 𝑥𝑎

* 1. **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 = I0\* cos(α)

Где I – результирующая интенсивность света в точке I0 – интенсивность источника

α – угол между нормалью к поверхности и вектором направления света

* 1. **Анимация движения автомобиля**

Автомобиль едет прямо с повторением

Для повышения достоверности необходимо создать эффект вращения колеса. Скорость вращения колеса зависит от скорости автомобиля.

Где

𝜔 =

2𝜋

𝑇 =

4𝜋2𝑅

𝑣

ω – cкорость вращения колеса R – радиус колеса

v – cкорость автомобиля

Во время движения автомобиля сцена не меняется.

* 1. **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
  + Многоугольник – хранит вершины, нормаль, цвет
* Интерфейс – используются библиотечные классы для предоставления доступа к интерфейсу.

# Технологическая часть

* 1. **Выбор и обоснование языка программирования и среды разработки**

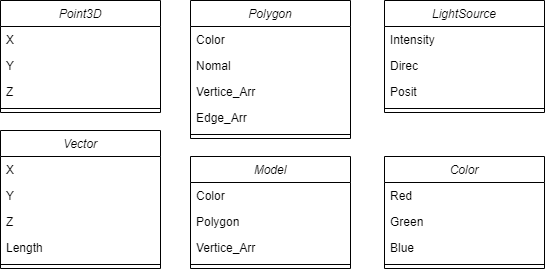
В качестве языка программирования был выбран Python т.к.:

* + - 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. oнa имеет множество удобств, которые облегчают процесс написания и отладки кода.

* 1. **Структура и состав классов**

В этом разделе будут рассмотрена структура и состав классов



*Рисунок 16. Cтруктура классов*

Point3D – хранить координат точки.

Vector – хранит направление вектора и его длину.

Polygon – класс многоугольника, хранит цвет, вершины, нормали.

Model – хранит информацию о модели: ее цвет, вершины, многоугольники. LightSource – класс источники света, хранить его интенсивность, цвет,

направление и положение. Color – класс цветa.

* 1. **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;

Zbuffer.py – алгоритм Z буфера; Colors.py – взаимодействие цветов;

Transformation.py – функции преобразования координат;

* 1. **Интерфейс программы**

Пользователь может выбрать направление источника света (из заданного набора).

Программа позволяет выполнить операции поворота сцены в нужных направлениях.

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

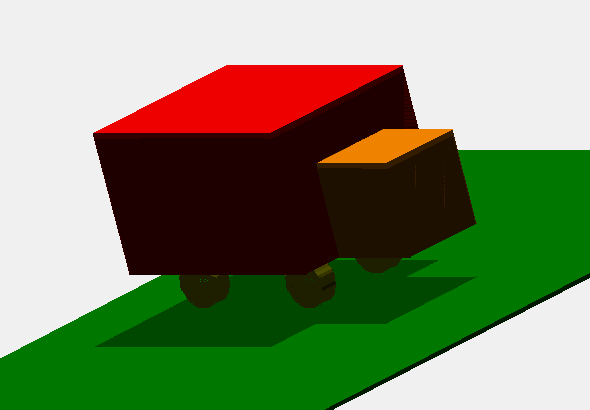
# Заключение

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.

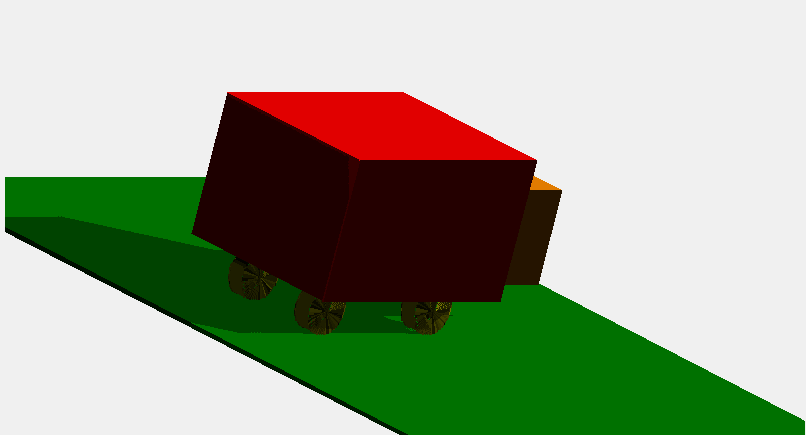
# Список использованной литературы

1. Компьютерная Графика - А. Ю. Дёмин, А. В. Кудинов. [[h](http://compgraph.tpu.ru/index.html)t[tp://compgraph.tpu.ru/index.html](http://compgraph.tpu.ru/index.html)]
2. Computer Graphics [https://[www.javatpoint.com/computer-graphics-tutorial](http://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]

# Приложения



*Рисунок 17. Визуализация сцены*



*Рисунок 18. Визуализация сцены*