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 Санг Фыок_____
фамилия, имя, отчество

Группа ____ИУ7И-46Б_____

Тип практики _____стационарная_____

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

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

Руководитель практики _____ ___Куров А.В._____
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2020 г.

Индивидуальное задание:

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

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 алгоритмов удаления невидимых линий и поверхностей

1.1.1. 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.

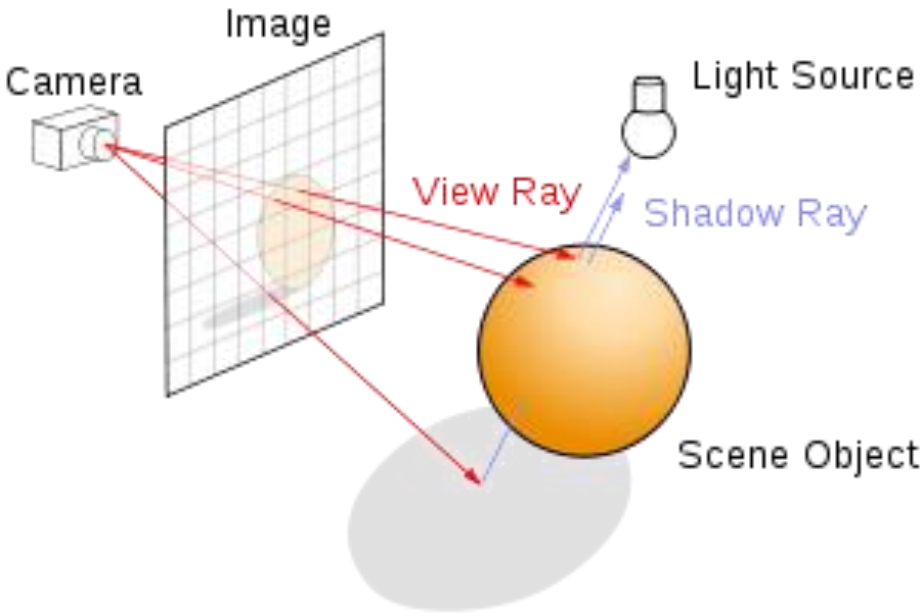


Рисунок 1.1.1. Алгоритм трассировки лучей.

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
```

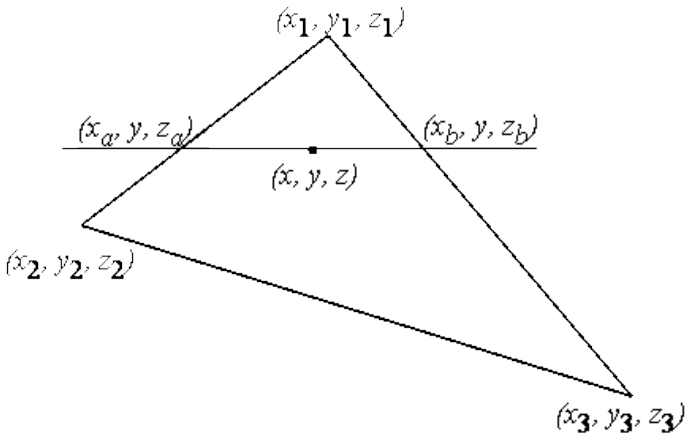
Преимущество: Высокая степень параллельности вычислений

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

1.1.2. 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)*.



Для нахождения необходимых значений Z, используется линейная интерполяция:

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

Рисунок 1.1.1. Линейная интерполяция

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

1.1.3. 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:

$$\mathbf{ax + by + cz + d = 0}$$

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.

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

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 $[E]\cdot[V] < 0$ 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 преграде рассматриваеммое тело

Уравнение отрезка:

$$p(t) = P1 - (P2 - P1)t, \qquad 0 \le t \le 1$$

Уравнение плоскости проходящий через отрезок и луч:

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

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

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

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.

1.2.Анализ методов закрашивания

1.2.1. 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.

1.2.2. 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.

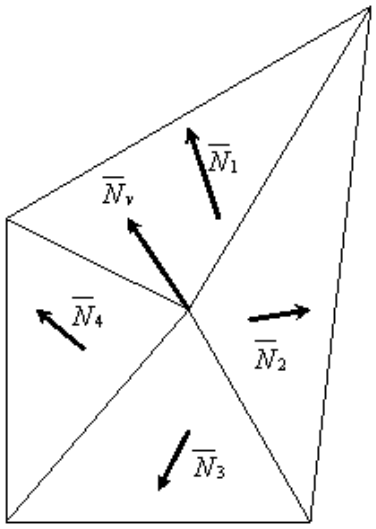


Рисунок 1.2.2-1. Нормали к вершинам

2) Определяются нормали в вершинах путем усреднения нормалей по всем полигональным граням, которым принадлежит вершина.

3) Используя нормали в вершинах и применяя произвольный метод закраски, вычисляются значения интенсивности в вершинах.
4) Каждый многоугольник закрашивается путем линейной интерполяции значений интенсивностей в вершинах сначала вдоль каждого ребра, а затем и между ребрами вдоль каждой сканирующей строки.

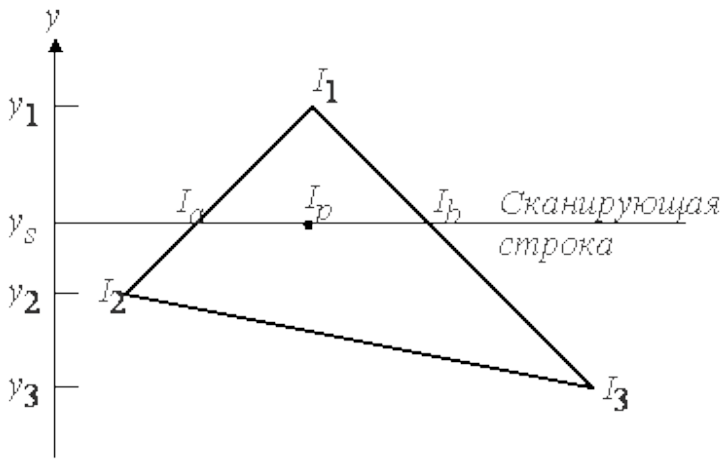


Рисунок 1.2.2-2. Интерполяция интенсивностей

$$I_a = u.I_1 + (1-u).I_2, u = \frac{I_1 I_a}{I_1 I_2}$$

$$I_b = w.I_1 + (1-w).I_3, w = \frac{I_1 I_b}{I_1 I_3}$$

$$I_p = t.I_a + (1-t).I_b, t = \frac{I_a I_p}{I_a I_b}$$

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.

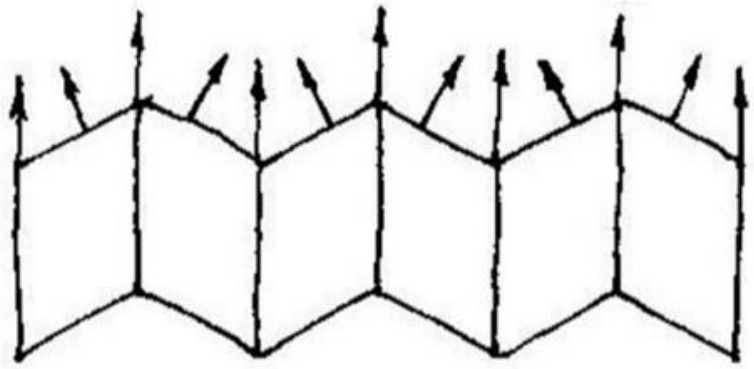
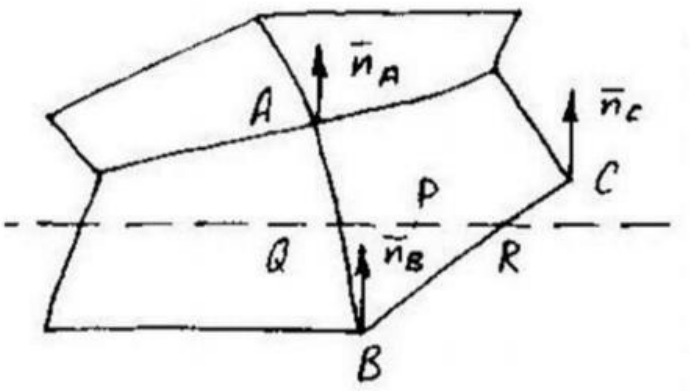


Рисунок 1.2.2-3. Поверхность закрашивается с одной интенсивностью

1.2.3. 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.



$$n_Q = u.n_A + (1-u)n_B, u = \frac{AQ}{AB}$$

$$n_R = w.n_C + (1-w)n_B, u = \frac{CR}{CB}$$

$$n_P = t.n_Q + (1-t)n_R, t = \frac{QP}{QR}$$

Рисунок 1.2.3-1. Закраска по Фонгу

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

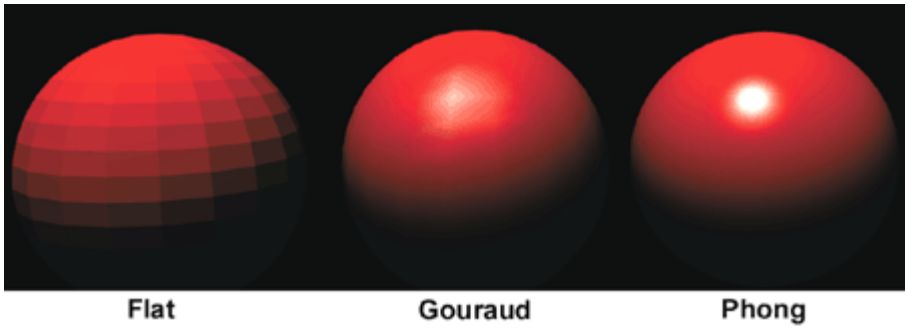


Рисунок 1.2.3-2. Сравнение методов закрашивания

1.3. 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.

1.4. 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.

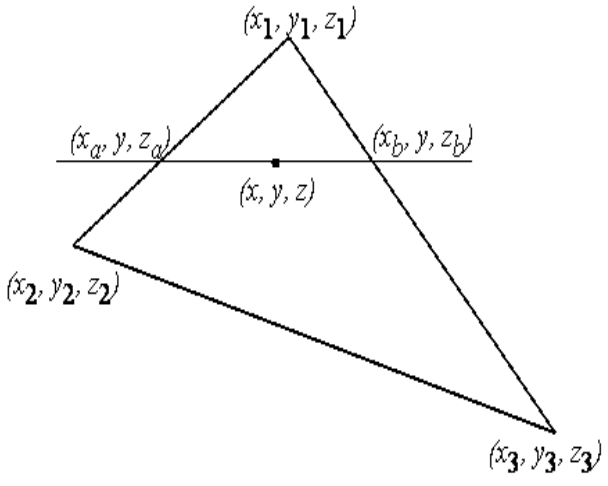
Из них самый подходящий способ для модели представления объектов – полигональное моделирование.

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 2.2. 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:

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

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

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

$$\omega = \frac{2\pi}{T} = \frac{4\pi^2 R}{v}$$

Где

ω – скорость вращения колеса

R – радиус колеса

v – скорость автомобиля

Во время движения автомобиля сцена не меняется.

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

3. Технологическая часть

3.1.Выбор и обоснование языка программирования и среды разработки

В качестве языка программирования был выбран 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.

3.2.Структура и состав классов

В этом разделе будут рассмотрена структура и состав классов

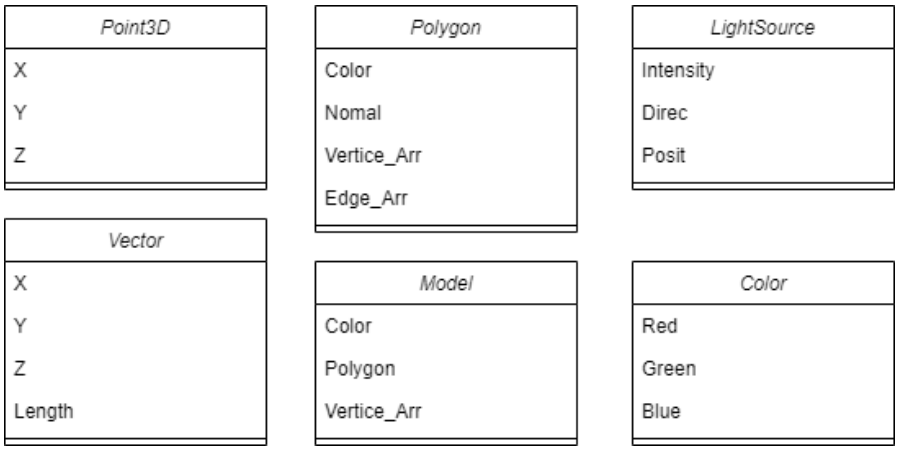


Рисунок 3.2. Структура классов

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 цвета.

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;

Zbuffer.py – алгоритм Z буфера;

Colors.py – взаимодействие цветов;

Transformation.py – функции преобразования координат;

3.4.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. [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]

Приложения

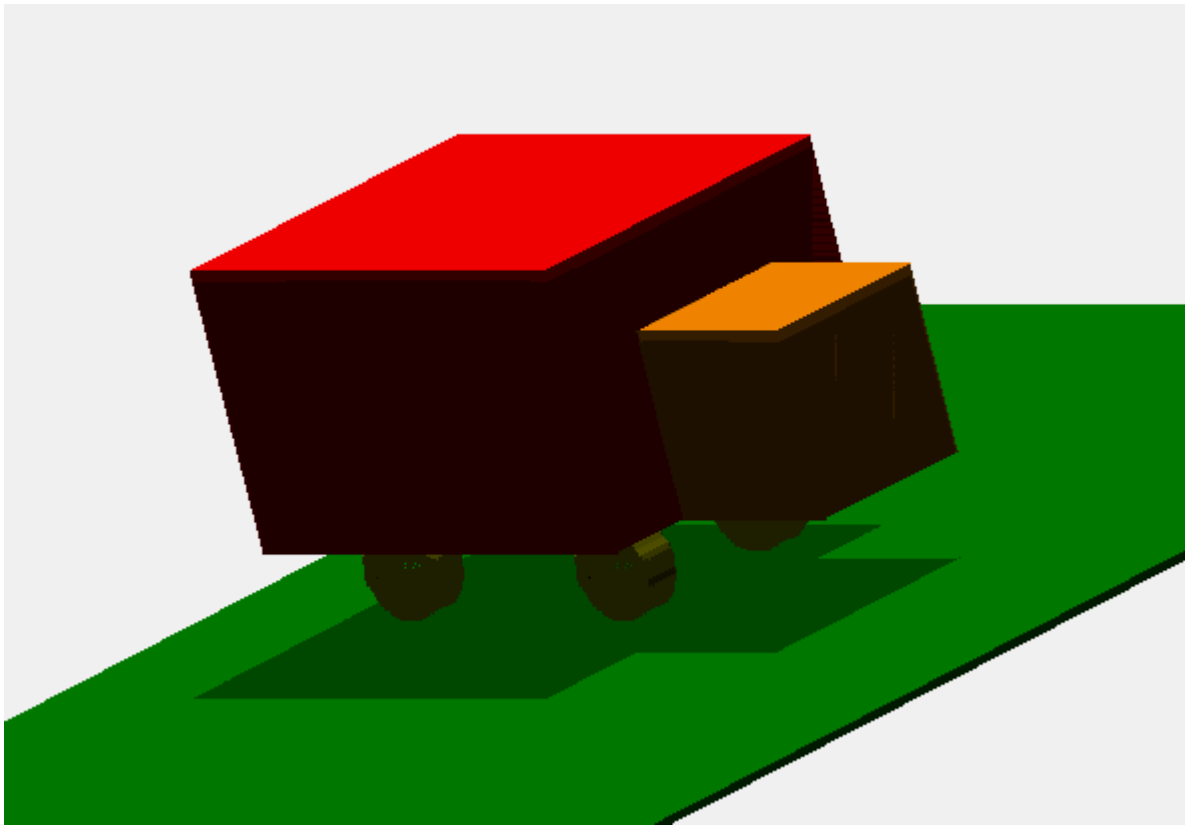


Рисунок 4-1. Визуализация сцены

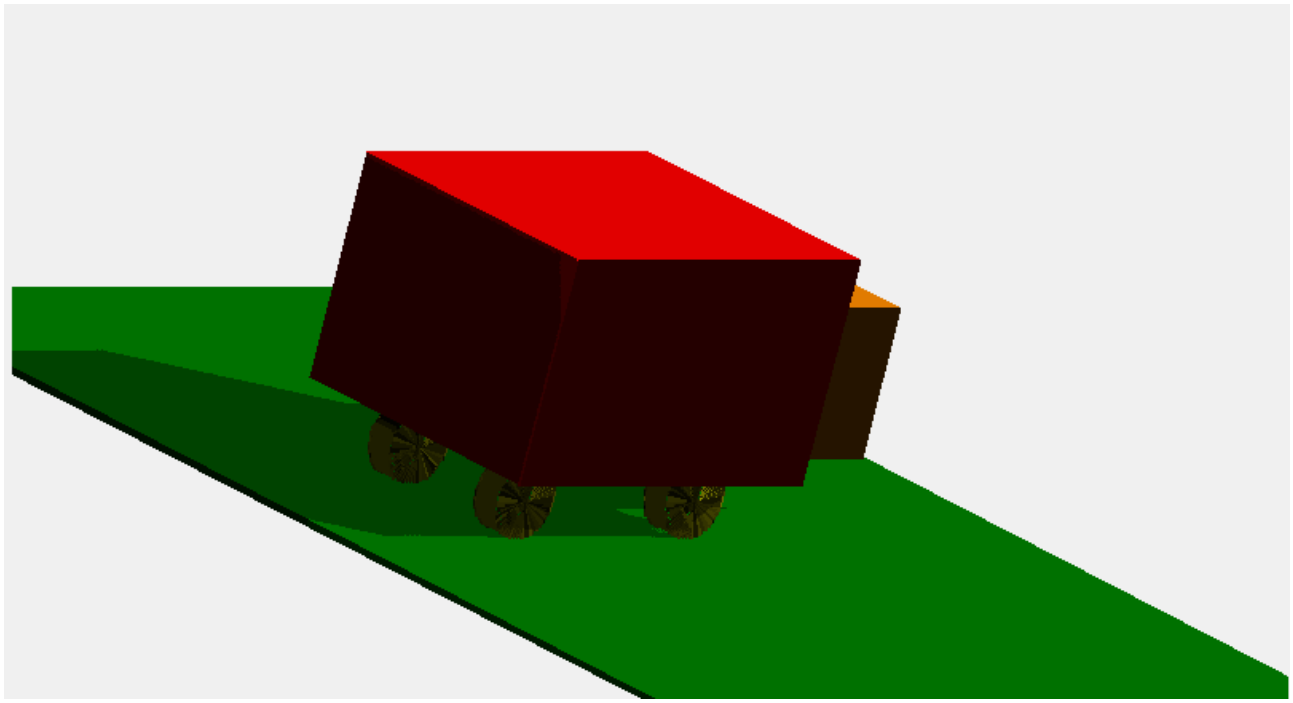


Рисунок 4-2. Визуализация сцены