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:

Реализация модели физического эксперимента «Маятник Фуко»

Студент _____ИУ7-55Б_____ _____ _____А.А. Лаврова_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ ____А.В. Силантьева_____
(Подпись, дата) (И.О.Фамилия)

2019 г.

Содержание

Введение

В современном мире компьютерная графика окружает человека практически везде. Прежде всего стал популярен синтез изображений, так как с помощью него разработчики могут создавать компьютерные игры, спецэффекты в кино, виртуальную реальность. Компьютерная графика используется в науке и промышленности для визуализации и моделирования различных процессов.

Для различных областей применения перед компьютерной графикой ставятся различные задачи. Например, создать наиболее реалистичное изображение или наоборот, выбрать такой алгоритм, который сможет удовлетворять требованиям производительности.

Целью проекта является разработка программы для создания трехмерных сцен из трехмерных геометрических примитивов для визуализации физической модели маятника Фуко.

Для достижения поставленной цели, необходимо решить следующие задачи:

- Выбрать оптимальные по времени работы алгоритмы
- Произвести основные математические расчеты для реализации модели физического эксперимента
- Выбрать подходящий язык программирования, спроектировать архитектуру программы
- Реализовать пользовательский интерфейс

1. Аналитический раздел

Для того чтобы выбрать подходящий алгоритм построения изображения, необходимо провести обзор известных алгоритмов и осуществить выбор наиболее подходящего для решения поставленной задачи.

В этом разделе проводится анализ существующих алгоритмов построения трехмерных изображений и выбираются наиболее подходящие алгоритмы для решения поставленных задач.

1.1 Формализация объектов синтезируемой сцены

Объекты сцены:

1) Маятник Фуко

   Сложный объект, состоит из 3 частей: шарика, нити и подвеса. Нить и подвес анимированы, подвес остается статичным. Данные о подвесе хранятся в текстовом файле в виде списка ребер и вершин. Радиус шарика и длина нити являются константами.

   Объект видимый.

2) Земная поверхность

   Равномерная поверхность.

   Объект видимый.

3) Камера

   Камера является точечным объектом, с помощью которой мы получаем перспективное отображение объектов сцены. Возможно создание нескольких камер.

   Объект невидимый.

4) Источник света

   Источник света испускает параллельные лучи, от угла падения которых зависит интенсивность цвета, а также тени объектов. Представлен точкой и углом наклона.

   Объект невидимый.

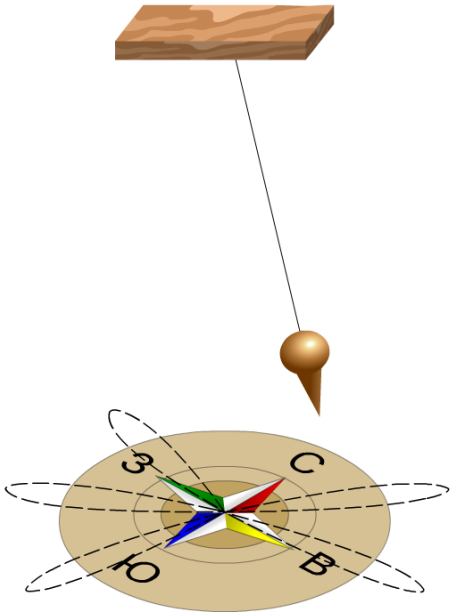

Рисунок 1 – модель маятника Фуко

Анимация:

Маятник совершает колебательные движения в зависимости от того, какая широта Земли задана пользователем. Эффект вращения плоскости колебаний максимален на полюсах и отсутствует на экваторе Земли.

Управление:

Реализация изменения эффекта вращения с помощью изменения широты, на которой находится маятник.

1.2 Алгоритмы построения трёхмерных изображений

Алгоритм Робертса:

Алгоритм Робертса является алгоритмом отсечения невидимых граней и ребер. Алгоритм работает в объектном пространстве. В нем используются уравнения плоскостей, пересечением которых образовано тело. Алгоритм Робертса может быть применен для изображения множества выпуклых

многогранников на одной сцене в виде проволочной модели с удаленными невидимыми линиями

Этапы работы:

1) Подготовка исходных данных
2) Удаление линий, экранируемых самим телом (для одного тела на этом работа заканчивается)
3) Удалений линий, экранируемых другими телами
4) Удаление линий пересечения тел, экранируемых самими телами, связанными отношением протыкания

Преимущество алгоритма заключается в том, что используются точные и мощные математические методы. К недостаткам данного метода можно отнести то, что его нельзя назвать эффективным из-за вычислительной трудоёмкости алгоритма.

Алгоритм Варнока:

Алгоритм Варнока работает в пространстве изображений. Единственной версии алгоритма не существует. Алгоритм Варнока и его варианты используют то, что большие области изображения когерентны.

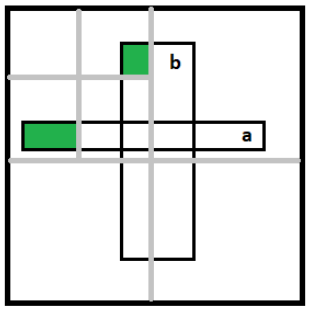

Рисунок 2 – Пример алгоритма Варнока

Рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если содержимое не оказывается таковым, то требуется дальнейшее разбиение на окна до тех пор, пока остаются области, содержащие не один многоугольник, или размер

области не станет совпадать с одним пикселом. Тогда для полученного пикселя необходимо вычислить значение координаты Z каждого многоугольника и визуализировать пиксел с максимальным значением этой координаты.

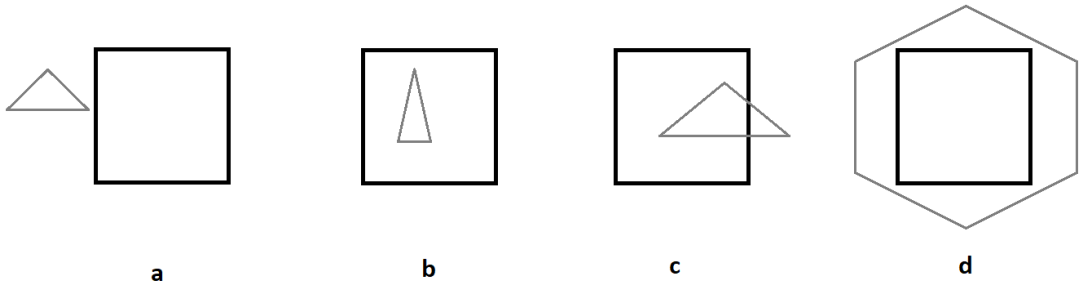

Рисунок 3 - Типы многоугольников: внешний (a), внутренний (b), пересекающий (c), охватывающий (d)

Достоинством данного алгоритма является то, что алгоритм достаточно прост с точки зрения понимания, однако может потребоваться большое количество разбиений, соответственно потребуется много времени на анализ и отображение содержимого всего окна.

Алгоритм трассировки лучей:

В данном алгоритме через каждый пиксел пространства изображения выпускается луч, с помощью которого определяется ближайшая к нему грань, а также находятся все его пересечения с гранями и среди них выбирается ближайшая.

К достоинству данного метода относится высокая реалистичность получаемого изображения за счёт создания тени и отраженного света, отсутствия аппроксимации гладких объектов примитивами.

Сложность использования алгоритма заключается в большом количестве вычислений, как следствие это отражается на производительности и скорости создания изображения. Однако вычислительная сложность метода линейно зависит от сложности сцены.

Алгоритм, использующий Z-буфер:

Алгоритм работает в пространстве изображений. Этот метод использует буфер кадра для заполнения интенсивности каждого пикселя, наряду с этим вводится Z-буфер – буфер глубины каждого пикселя.

Значение координаты Z (глубина) каждого нового пиксела, который надо занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в Z-буфер. Если это сравнение показывает, что новый пиксел расположен ближе к наблюдателю, чем пиксел, уже находящийся в буфере кадра, то новое значение координаты Z заносится в Z-буфер, корректируется значение интенсивности в буфере кадра.

Алгоритм, использующий Z-буфер очень прост в реализации, также не тратится время на сортировку элементов сцены.

Однако следует учесть, что увеличиваются затраты памяти при использовании этого метода, так как приходится запоминать информацию о глубине и цвете каждого пиксела изображения.

Вывод:

Не смотря на то, что изначально мной был выбран алгоритм трассировки лучей из-за высокой реалистичности получаемого изображения, в ходе анализа алгоритмов было выяснено, что целесообразно выбрать алгоритм, использующий Z-буфер, так как для динамических сцен важна скорость работы алгоритма.

1.3 Алгоритмы закраски

Метод закраски Гуро:

Данный вид закраски основан на билинейной интерполяции интенсивности, что позволяет получить сглаженное изображение. Метод основан на идее закрашивания грани не одним цветом, а плавно изменяющимися оттенками. Для вычисления оттенка, интерполируются цвета примыкающих граней.

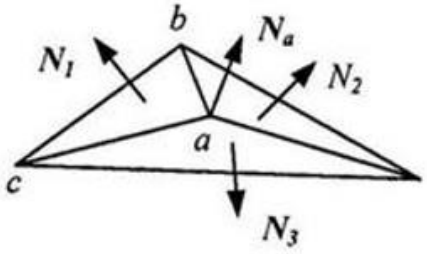

Рисунок 4 – Определение нормалей

Закраска по Гуро хорошо сочетается с диффузным отражением. Данный метод интерполяции обеспечивает непрерывность значений интенсивности только вдоль границ многоугольников, но не обеспечивает непрерывность изменения интенсивности, что приводит к появлению полос Маха.

Метод закраски Фонга:

Данный вид закраски основан на билинейной интерполяции вектора нормали. Используя закраску по Фонгу, можно достичь лучшей локальной аппроксимации кривизны поверхности. Это позволяет уменьшить количество полос Маха.

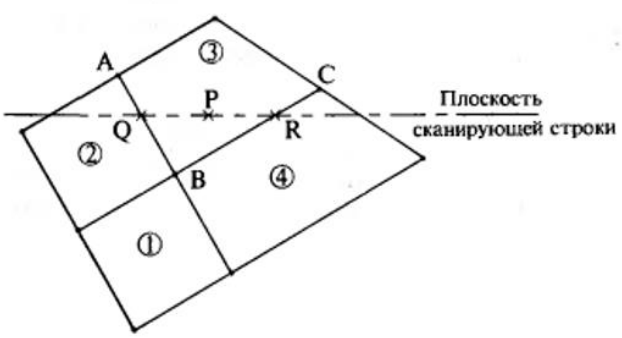

Рисунок 5 – Пример работы метода закраски Фонга

Закраска Фонга требует больших вычислительных затрат, но при этом достигается лучшая локальная аппроксимация кривизны поверхности, более реалистично выглядят зеркальные блики.

Простой метод освещения:

Вся грань закрашивается одним уровнем интенсивности, который высчитывается по закону Ламберта.

Этот метод крайне прост в реализации и совершенно не требователен к ресурсам. Однако плохо подходит для тел вращения и почти не учитывает отраженный свет. В моём случае этот метод очень хорошо подходит для отображения плоскости и куба.

Вывод:

Исходя из поставленной задачи наиболее оптимальным решением станет комбинация закраски по методу Гуро, так как в сцене предусматривается появление зеркального объекта (металлический шарик), и плоской закраски для отображения плоскости и куба (подвеса).

1.4 Модель освещения

Существует две модели освещенности: локальная и глобальная. Локальная модель учитывает освещение точечных источников, а в глобальной модели освещения отслеживается весь путь луча до тех пор, пока он не покинет сцену или переносимая им энергия не станет достаточно малой, чтобы не учитывать ее.

Глобальная модель освещения позволяет создать более реалистичное изображение. Локальная модель может быть достаточной для работы с матовыми объектами.

Вывод:

Будет использоваться локальная модель освещения, так как большинство объектов сцены будут матовыми. Также использование глобальной модели освещения увеличит количество вычислений и сложность программы.

2. Конструкторский раздел

2.1 Алгоритм, использующий Z-буфер

Данный алгоритм работает в пространстве изображения.

Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пиксела в пространстве изображения. Z-буфер – это отдельный буфер глубины, используемый для запоминания координаты Z каждого видимого пиксела в пространстве изображения.

Алгоритм:

1) Заполнить буфер кадра фоновым цветом
2) Заполнить Z-буфер минимальным значением глубины
3) Преобразовать каждый многоугольник в растровую форму в произвольном порядке

    3.1) Для каждой точки экрана, покрываемой многоугольником вычислить значение Z(X, Y)

    3.2) Если Z(X, Y) > Zбуф(X, Y) => Zбуф(X,Y)=Z(X, Y)

    Цвет (X,Y) = Цвет текущего многоугольника
4) Отобразить результат

2.2 Алгоритм, использующий трассировку лучей

Наблюдатель видит объект посредством испускаемого неким источником света, который падает на поверхность объекта и затем как-то доходит до наблюдателя. В алгоритме отслеживаются лучи в обратном направлении: от наблюдателя к объекту.

В простейшем виде суть алгоритма заключается в следующем:

1) Экран представляется как физический объект, находящийся на некотором удалении от наблюдателя
2) В каждую точку экрана пускается луч
3) Для каждого луча ищется пересечение с охватывающими оболочками объектов сцены

4) В случае нахождения пересечения с охватывающей оболочкой, ищется пересечение луча с самим объектом
5) Если пересечение есть, для данной точки рассчитывается суммарная интенсивность света от каждого из источников

2.3 Метод закраски Гуро

Данный вид закраски основан на билинейной интерполяции вектора нормали, что позволяет реалистично передать зеркальные блики.

Алгоритм:

1) Вычисление векторов нормалей к каждой грани
2) Вычисление векторов нормалей к каждой вершине грани через усреднение нормалей к граням
3) Вычисление интенсивностей в вершинах грани
4) Интерполяция интенсивности вдоль ребер грани:
   $I_q = u*I_A + (1-u)*I_B$, $0 \le u \le 1$, $u = (AQ)/(AB)$
   $I_r = w*I_B + (1-w)*Ic$, $0 \le w \le 1$, $w=(BR)/(BC)$
5) Линейная интерполяция интенсивности вдоль сканирующей строки
   $I_P = t*I_q + (1-t)*I_r$, $0 \le t \le 1$, $t = (QP)/(QR)$

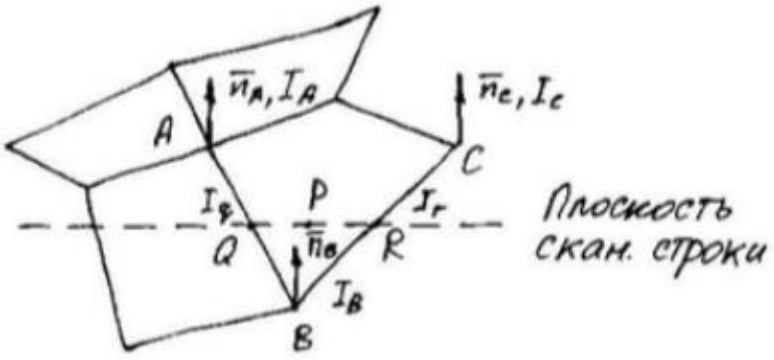

Рисунок 6 – пример метода закраски Гуро

2.4 Метод закраски Фонга

Данный вид закраски основан на билинейной интерполяции вектора нормали, что позволяет реалистично передать зеркальные блики.

Алгоритм:

6) Вычисление векторов нормалей к каждой грани
7) Вычисление векторов нормалей к каждой вершине грани через усреднение нормалей к граням
8) Интерполяция векторов нормалей вдоль ребер грани:
   $n_q = u*n_A + (1-u)*n_B$, $0 \le u \le 1$, $u = (AQ)/(AB)$
   $n_r = w*n_B + (1-w)*nc$, $0 \le w \le 1$, $w=(BR)/(BC)$
9) Линейная интерполяция интенсивности вдоль сканирующей строки
   $n_P = t*n_q + (1-t)*n_r$, $0 \le t \le 1$, $t = (QP)/(QR)$
10) Вычисление интенсивности в очередной точке сканирующей строки

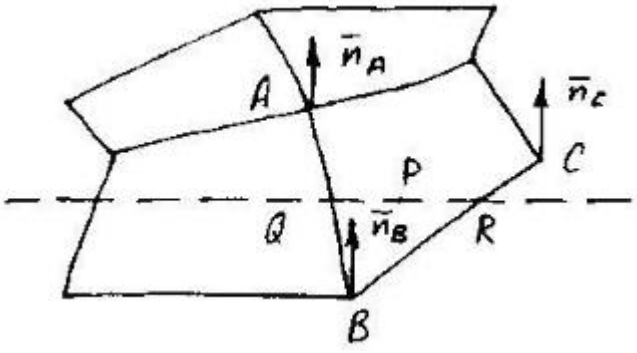

Рисунок 7 – пример метода закраски Фонга

2.5 Простой метод освещения

В простом методе освещения интенсивность рассчитывается по закону Ламберта:

$$I = I0*\cos(\alpha), \text{ где}$$

I – результирующая интенсивность света в точке

I0 – интенсивность источника

α – угол между нормалью к поверхности и вектором направления света

2.6 Моделирование движения маятника Фуко

Рассмотрим сферический маятник длиной l и массой m, совершающий колебания в гравитационном поле Земли. Система отсчета, жестко связанная с Землей, является неинерциальной, поэтому нужно учитывать силы инерции. Пусть маятник находится на широте j. Выберем систему координат следующим образом: направим ось Ox по параллели, ось Oy – по меридиану, ось Oz – по местной вертикали. Тогда проекции угловой скорости вращения Земли $\vec{\omega}$ будут на оси соответственно равны, $\omega_x = 0$, $\omega_y = -\omega\cos\varphi$, $\omega_z = \omega\sin\varphi$

Уравнение движения подвешенного груза (материальной точки):

$$m\vec{a} = m\vec{g} + \vec{T} + 2m\vec{v} \times \vec{\omega} \quad (1)$$

где $\vec{a}$ – ускорение груза; $\vec{g}$ – ускорение свободного падения; $\vec{T}$ – сила натяжения троса; $\vec{v}$ - скорость груза. Последнее слагаемое – сила Кориолиса, направленная под некоторым углом к плоскости, в которой расположены силы тяжести и натяжения.

Спроектируем уравнение на оси выбранной системы координат:

$$ma_x = T_x + 2m(v_y\omega_z - v_z\omega_y) \quad (2)$$

$$ma_y = T_y - 2mv_x\omega_z \quad (3)$$

$$ma_z = -mg + T_z + 2mv_x\omega_y \quad (4)$$

где $T_x = -Tsin\alpha cos\theta = -T\frac{x}{l}$

$T_y = -Tin\alpha cos\theta = -T\frac{y}{l}$

$Tz = Tcos\alpha = T\frac{l-z}{l}$; α – угол отклонения маятника от вертикали; θ – азимутальный угол

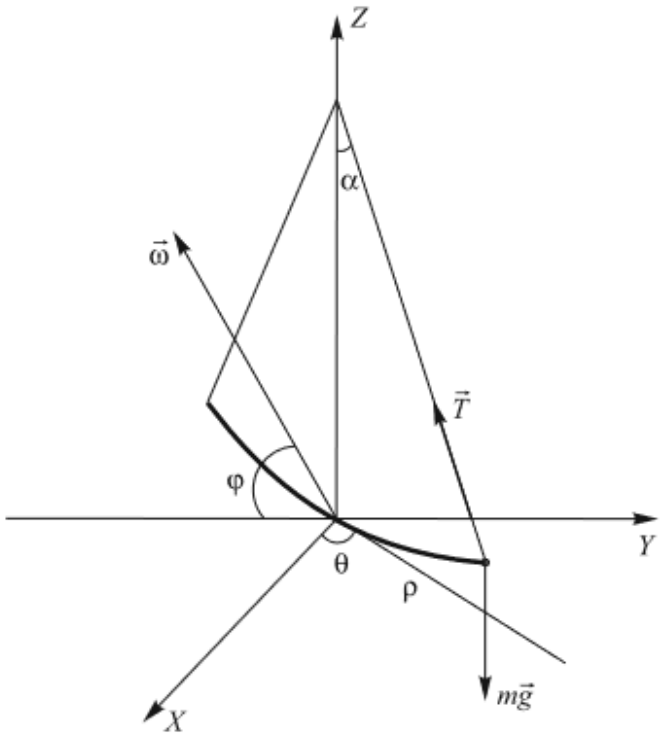

Рисунок 8 – Система отсчета

При малых углах α (малые колебания) sinα ≈ α, cosα ≈ 1, поэтому z = a_z = ν_z = 0. Из проекции уравнения на ось Z (4) также следует, что T ≈ mg. Поэтому можно считать, что груз движется в плоскости xOy. С учетом выражений для координат силы натяжения и после деления на массу уравнения (2) и (3) примут вид:

$$\frac{d^2x}{dt^2} = \omega_0^2 x + 2\frac{dy}{dt}\omega_z \tag{5}$$

$$\frac{d^2y}{dt^2} = -\omega_0^2 x - 2\frac{dx}{dt}\omega_z \tag{6}$$

где $\omega_0 = \sqrt{\frac{g}{l}}$ - собственная частоты колебания маятника.

Для решения системы линейных уравнений с постоянными коэффициентами при начальных условиях x(0) = x_0, y(0) = 0, $\nu_y(0) = \nu_0$, используем преобразование Лапласа $F(s) = \int_0^\infty f(t)e^{-st}\,dt$. В результате получим систему линейных уравнений:

$$(s^2 + \omega_0^2)X(s) - 2s\omega_z Y(s) = sx_0$$

$$(s^2 + \omega_0^2)Y(s) + 2s\omega_z X(s) = v_0 + 2\omega_z x_0$$

Решение полученной системы уравнений:

$$X(s) = \frac{x_0\omega_- - \nu_0}{2\Omega}\frac{s}{s^2 + \omega_+^2} + \frac{x_0\omega_+ + \nu_0}{2\Omega}\frac{s}{s^2 + \omega^2}$$

$$Y(s) = -\frac{x_0\omega_0^2 - \nu_0\omega_+}{2\Omega}\frac{1}{s^2 + \omega_+^2} + \frac{x_0\omega_0^2 + \nu_0\omega_-}{2\Omega}\frac{1}{s^2 + \omega_-^2}$$

где $\Omega = \sqrt{\omega_0^2 + \omega_z^2},\ \omega_\pm = \Omega \pm \omega_z$

Применив обратное преобразование Лапласа, получим уравнения, полностью описывающие движение маятника:

$$x(t) = \frac{x_0\omega_- - \nu_0}{2\Omega}cos\omega_+ t + \frac{x_0\omega_+ + \nu_0}{2\Omega}cos\omega_- t$$

$$y(t) = -\frac{x_0\omega_- - \nu_0}{2\Omega}sin\omega_+ t + \frac{x_0\omega_+ + \nu_0}{2\Omega}sin\omega_- t$$

3. Технологический раздел

3.1 Выбор среды разработки и языка программирования

Для разработки программы выбран объектно-ориентированный подход. Он позволяет обеспечить общий интерфейс взаимодействия с различными объектами, а также делает программу достаточно гибкой для дальнейшего расширения и модификаций, что является весомым аргументом в пользу этого подхода.

Языком программирования для разрабатываемого приложения был выбран язык C++ в связи со следующими факторами:

- язык является объектно-ориентированным
- обладает высокой эффективностью
- явная работа с памятью (программист всегда знает, на что тратятся вычислительные ресурсы)

В качестве среды разработки использована Qt Creator, так как с её помощью удобно создавать графические интерфейсы, использовать сигнально-слотовую связь для реализации асинхронного взаимодействия классов. Среда обладает кроссплатформенностью и большим выбором настроек проекта.

3.2 Структура и состав классов

В этом разделе будут рассмотрена структура и состав классов, см. рис. 9, рис. 10, рис 11.

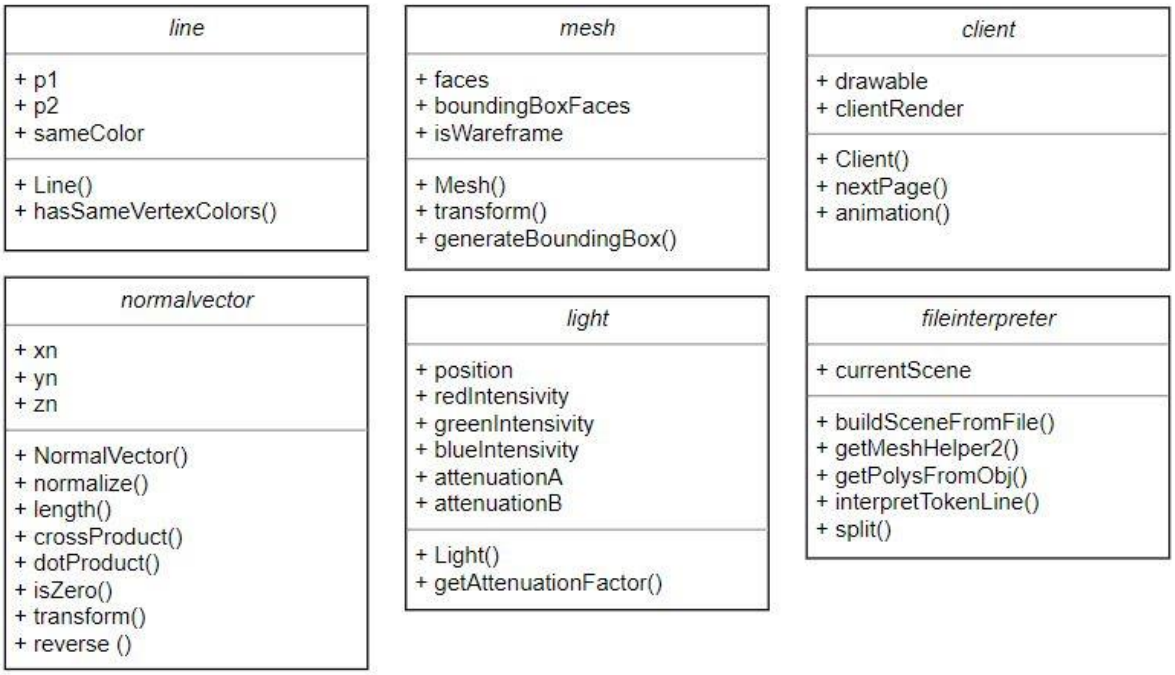

Рисунок 9 - структура классов line, mesh, client, normalvector, light, fileinterpreter

client – обрабатывает поступающие от пользователя сигналы и хранит информацию об анимации

fileinterpreter – обрабатывает получаемые на вход .obj файлы и собирает сетку полигонов

light – хранит положение источника света в пространстве, цвет и коэффициенты затухания, рассчитывает свет в точкой

line – класс линии, хранит её координаты и определяет, имеет ли эта линия одинаковые цвета вершин

mesh – содержит набор полигонов, набор из 6 граней, ограничивающих данный многоугольник, имеет методы преобразования сетки с помощью матрицы

normalvector – хранит координаты вектора нормали, имеет методы нормализации вектора, скалярного произведения, получения длины вектора

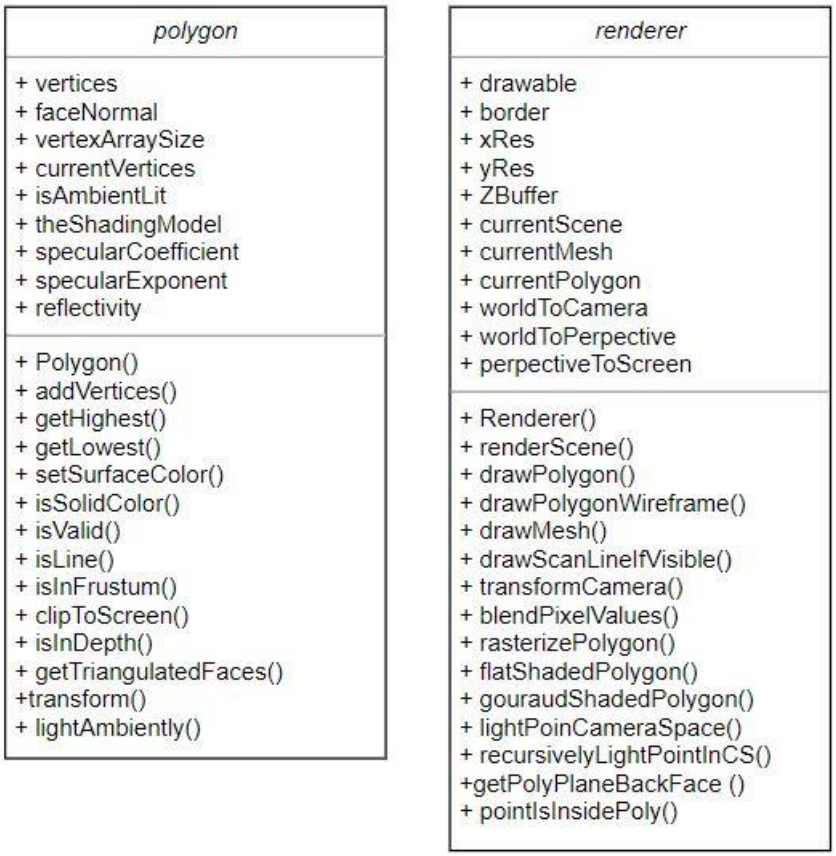

Рисунок 10 – структура классов polygon, renderer

polygon – класс обработки многоугольников, хранит массив точек и нормаль к поверхности этого многоугольника, модель закраски, коэффициент отражения, имеет методы, получения вершин с различными значениями, установки цвета, проверки правильности инициализации, проверки нахождения внутри границ, триангуляции многоугольника, преобразования с помощью матрицы, изменения цвета вершин в зависимости от интенсивности освещения

renderer – класс визуализации сцены, имеет методы растеризации сцены, растеризации линии, растеризация полигона с различными видами закрасок, трассировки лучей, расчёта Z-буффера

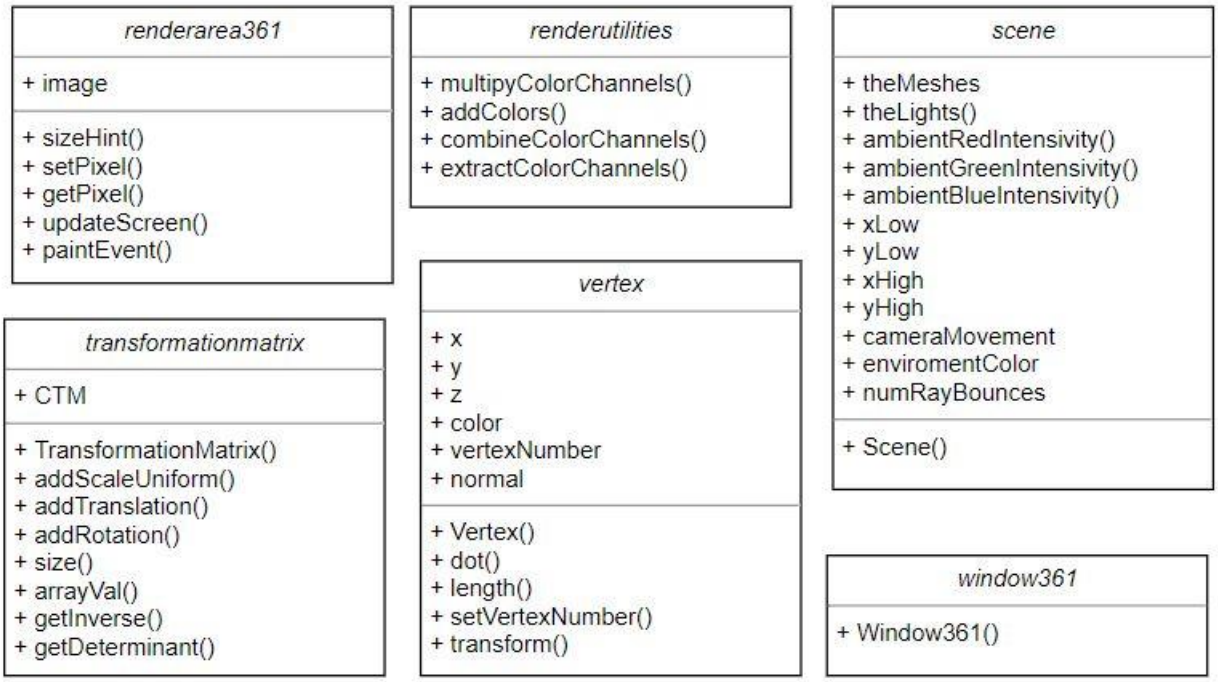

Рисунок 11 – структуры классов renderarea361, renderuilities, scene, transformationmatrix, vertex, window361

renderarea361 – класс вывода сцены на экран

renderutilities - класс работы с цветом

scene – класс сцены, содержит сетки, источник света, настройки камеры

transformationmatrix – имеет методы операций на матрицами, хранит матрицу преобразований

vertex – класс работы с вершинами, имеет методы установки вершины, проверки корректности вершин, произведения, установки номера вершины, хранит координаты вершины, ее цвет по умолчанию и индекс

3.3 Интерфейс программы

Интерфейс включает в себя следующие элементы:

- Сцена
- Поле ввода широты

- Поля ввода координат камеры
- Кнопка запуска анимации
- Кнопка завершения программы

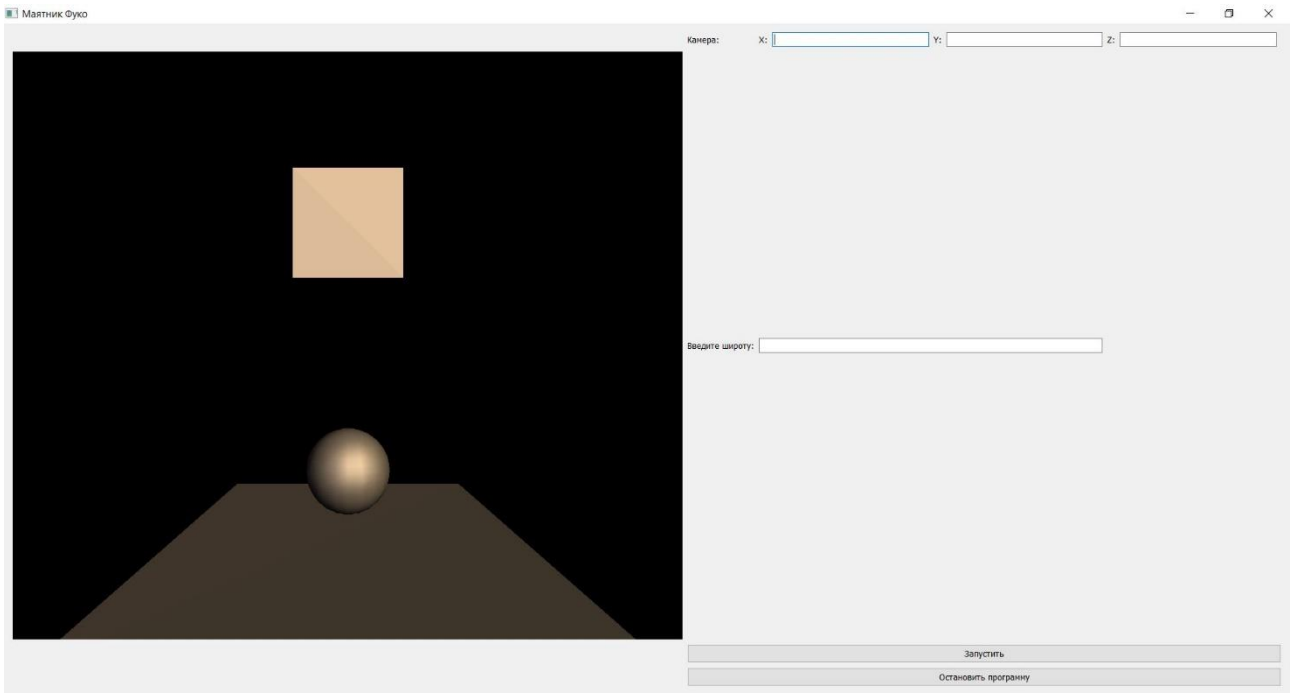

Рисунок 12 – интерфейс программы

4. Экспериментальный раздел

В данном разделе будет проведена апробация реализованной программы для проверки корректности ее работы и поставлен эксперимент по оценки эффективности работы программы.

4.1 Цель эксперимента

Целью эксперимента является проверка правильности выполнения поставленной задачи, оценка эффективности при различных способах закраски объектов.

4.2 Апробация

На рисунках 13-15 показана модель маятника Фуко при использовании плоской закраски, а также закраски по методу Гуро и по методу Фонга.

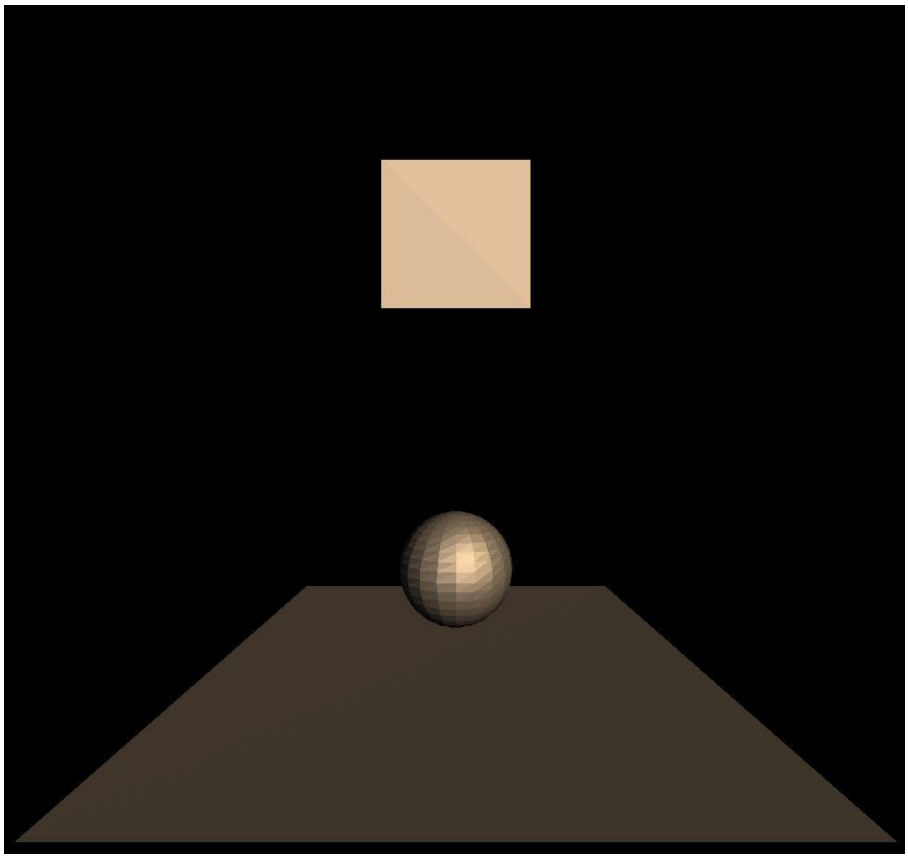

Рисунок 13 – плоская закраска

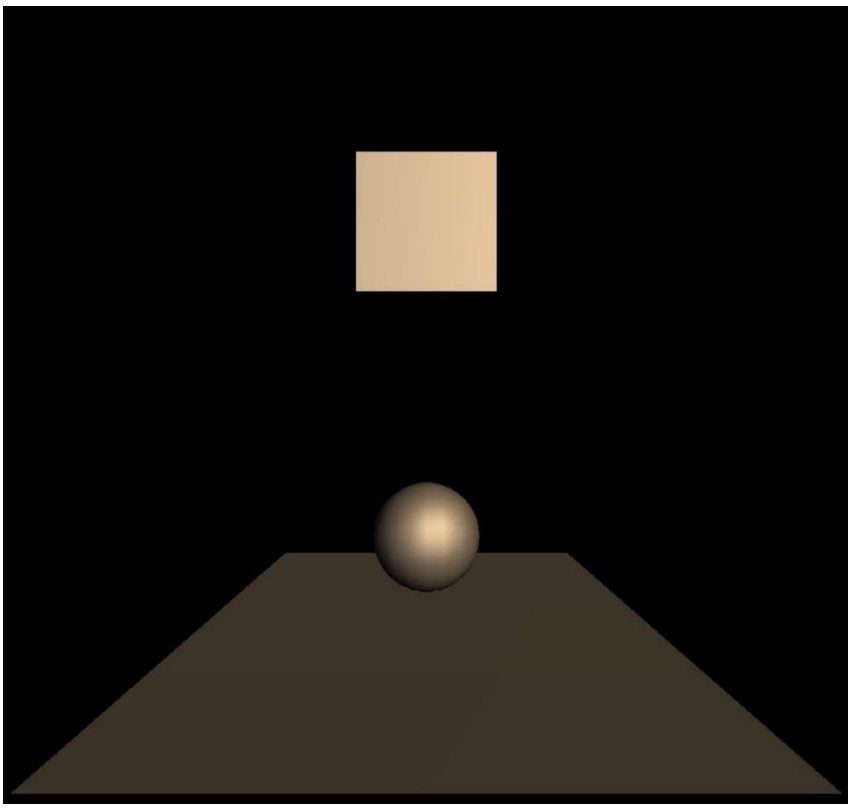

Рисунок 14 – закраска по методу Гуро

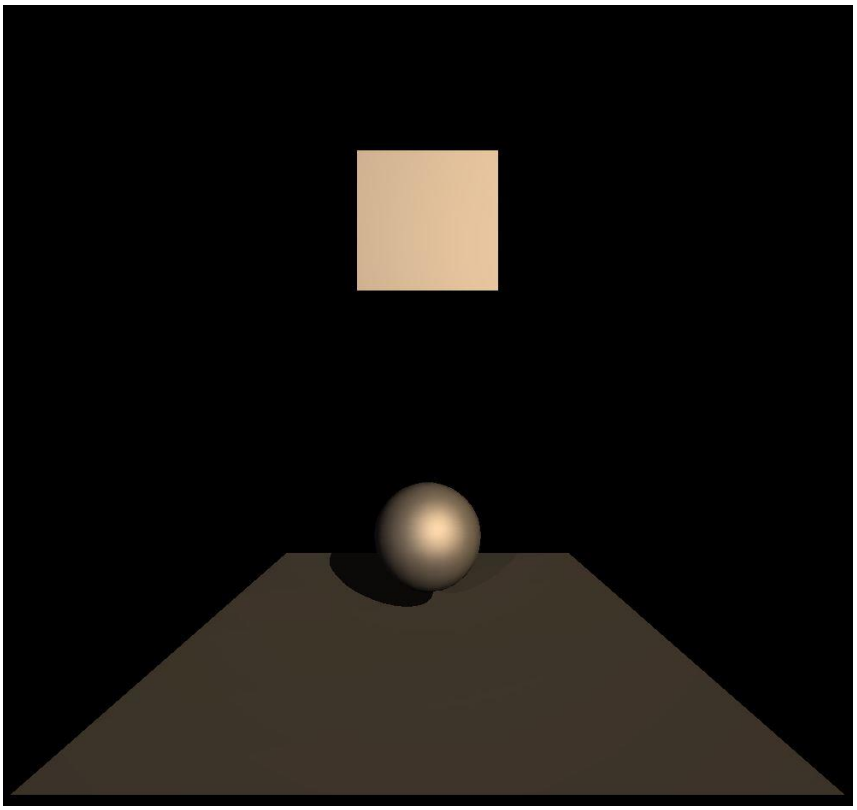

Рисунок 15 – закраска по методу Фонга

На рисунках показана одна и та же модель, построение изображения работает корректно при разных закрасках. При использовании закраски по Фонгу с помощью трассировки лучей появляются тени. Однако наиболее оптимальной является комбинация из плоской закраски (куб и плоскость) и закраски по методу Гуро (шар), так как для анимации необходимо добиться наиболее быстрого построения сцены, и при этом не потерять физическую достоверность металлического шарика (рисунок 16).

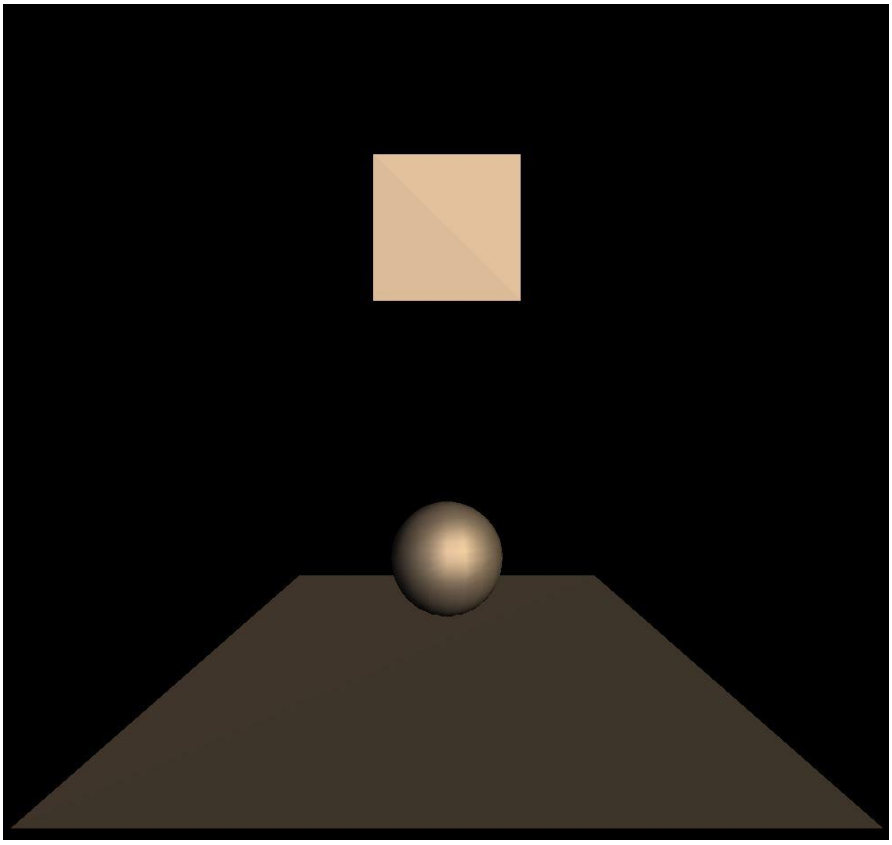

Рисунок 16 – комбинированная закраска

На рисунке 17 показана возможность изменения цвета объектов.

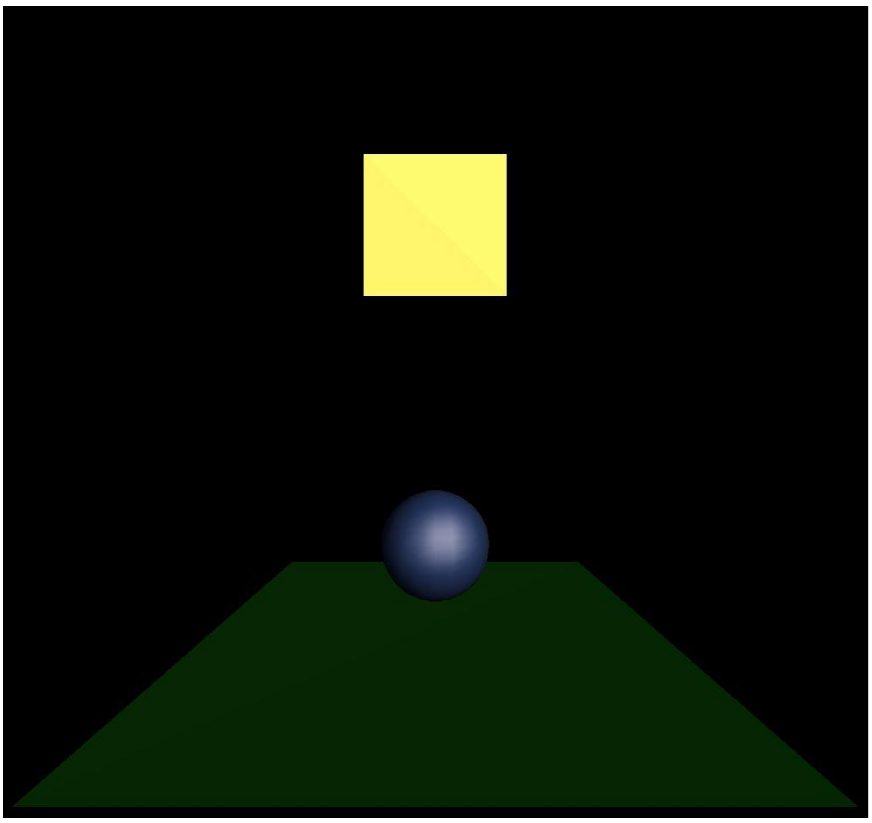

Рисунок 17 – комбинированная закраска с цветными объектами

По умолчанию координаты камеры заданы как (0, 1, -4.05). На рисунке 18 представлен пример изменения положения камеры.

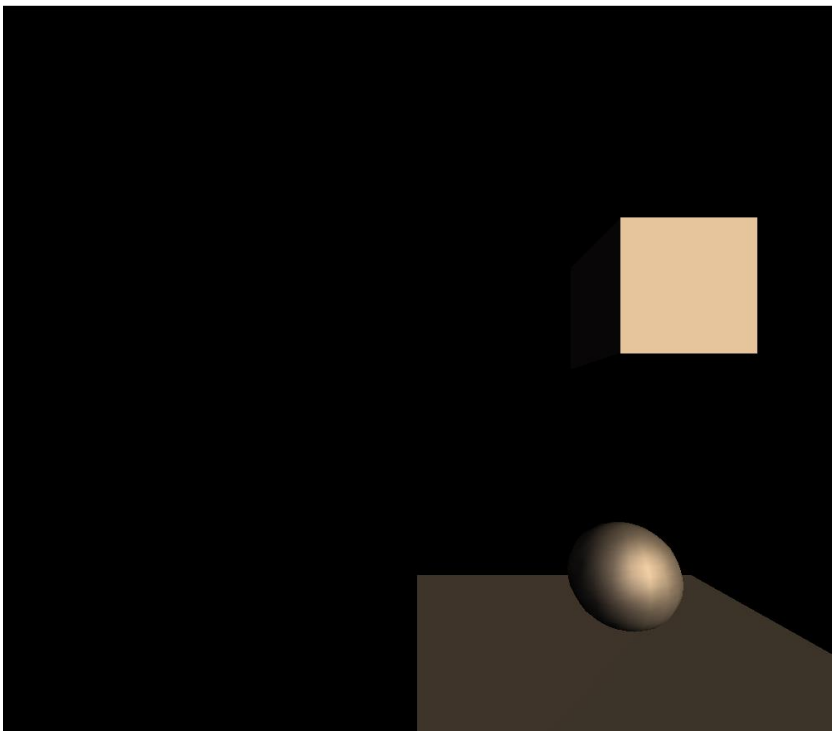

Рисунок 18 – пример изменения положения камеры

4.3 Описание эксперимента

Были реализованы 3 вида закраски, причем при использовании закраски по методу Фонга растеризация объекта выполняется с помощью трассировки лучей, а при использовании плоской закраски или закраски по методу Гуро растеризация выполняется с помощью Z-буффера.

Несмотря на то, что с помощью трассировки лучей можно добиться высокой реалистичности получаемого изображения, этот алгоритм плохо совместим с анимацией, так как на создание одной сцены уходит много времени. С другой стороны, наиболее выигрышный по времени вариант с плоской закраской всех объектов не подходит из-за того, что неверно передаёт форму объекта (даёт плохую аппроксимацию кривых поверхностей).

Эксперимент проводился на компьютере со следующими характеристиками:

- Intel® Core™ i5-7200U
- 2 ядра
- 4 логических процессора
- 8 Гб оперативной памяти

В данном эксперименте проведены замеры времени визуализации одной сцены при разных типах закраски:

- 115201 мс – плоская закраска всех объектов
- 1188023 мс закраска всех объектов по методу Гуро
- 11449480 мс – закраска всех объектов по методу Фонга
- 1078115 мс – наиболее оптимальная комбинированная закраска (плоскость и куб – плоская закраска, шар – закраска по методу Гуро)

Из представленных данных видно, что плоская закраска работает в 10,1 раз быстрее закраски по методу Гуро, и в 99,3 раз быстрее закраски по методу

Фонга. Оптимальная комбинированная закраска в 1,1 раз быстрее закраски по методу Гуро, и в 11 раз быстрее закраски по Фонгу.

Заключение

В результате проделанной работы были исследованы алгоритмы удаления невидимых линий, построения теней, методы закрашивания. Были проанализированы их достоинства и недостатки, выбраны наиболее подходящие для решения поставленной задачи. Также была разработана формализация сцены, произведен вывод физических формул траектории движения маятника Фуко.

В ходе выполнения поставленной задачи были изучены возможности QT Creator и языка программирования C++, получены знания в области компьютерной графики.

В ходе выполнения экспериментальной части было установлено, что комбинированная закраска с использованием Z-буфера позволяет уменьшить время работы практически в 100 раз по сравнению с изначально выбранным алгоритмом трассировки лучей.

Список источников

1. Роджерс Д., Математические основы машинной графики. / Роджерс Д., Адамс Д. – М.: Мир, 2001
2. Куров А.В., Курс лекций по компьютерной графике в МГТУ им. Н.Э. Баумана / Куров А.В. – М.: [б.н.], 2017 г.
3. Жолнеревич И. И., Опыт Фуко. Аналитическое решение уравнения движения маятника. / Жолнеревич И. И. – Вестник БГУ. Сер. 1. 2015. № 2
4. Булатов Л.А., Анализ движения маятника Фуко / Булатов Л.А., Бертяев В.Д., Киреева А.Е. – ТулГУ