Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Моделирование сцены, расположенной за прозрачной поверхностью»

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

Студент _____ Д.Р.Жигалкин
(Подпись, дата) (И.О.Фамилия)

2020 г.

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана» (МГТУ им. Н.Э. Баумана)

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

Моделирование сцены, расположенной за прозрачной поверхностью
(Тема курсового проекта)

Студент Жигалкин Д.Р гр. ИУ7-55Б
(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 программное обеспечение для создания реалистичного изображения методом обратной трассировки лучей. Изображение должно представлять собой сцену, на которую будет можно установить только заданные в программе объекты: сферу, параллелепипед, плоскость, а также выбрать их характеристики: цвет, зеркальное отражения (по шкале), отражающую способность (можно выбрать предел рекурсии) (по шкале), прозрачность (по шкале). Предоставить возможность выбрать заданные в программе источники света: направленный, точечный и окружающее освещение и выбрать их интенсивность, а также менять положение объектов, источников света (кроме окружающего освещения). Должна быть возможность перемещать камеру наблюдателя на заданные координаты и поворачивать на заданный угол вокруг заданной оси. Окно программы должно иметь фиксированный размер.

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 _____ О.В.Кузнецова
(Подпись, дата) (И.О.Фамилия)

Студент _____ Д.Р.Жигалкин
(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение

В компьютерной графике на сегодняшний день большое внимание уделяется алгоритмам получения реалистических изображений. Эти алгоритмы являются самыми затратными по времени. Обусловлено это тем, что они должны предусматривать множество физических явлений, таких как преломление, отражение, рассеивание света. Профессиональные программы для кинематографа учитывают еще больше явлений (дифракцию, интерференцию, зависимость коэффициентов преломления, отражения, поглощения от длины волны падающего света, вторичное, третичное отражение света).

В моей курсовой работе для рендеринга применяется алгоритм обратной трассировки лучей. На сегодняшний день он считается одним из лучших для формирования реалистических изображений, его используют большинство трехмерных графических редакторов. В таких редакторах применяется так же алгоритм z-буфера, но им пользуются там, где крайне важна скорость.

Цель данной работы – моделирование реалистичной сцены, расположенной за прозрачной поверхностью.

Для достижения поставленной цели требуется решить следующие задачи:

1) описать структуру трехмерной сцены, включая объекты, из которых состоит сцена, и дать описание выбранных свойств;

2) выбор и/или модифицирование существующих алгоритмов трехмерной графики, которые позволят визуализировать трехмерную сцену;

3) реализация данных алгоритмов для создания трехмерной сцены;

4) разработать программное обеспечение, которое позволит отобразить трехмерную сцену и визуализировать оптические эффекты.

1. Аналитическая часть

1.1 Обзор и анализ существующего программного обеспечения и обоснование необходимости разработки

На данный момент существует множество программных систем для синтеза изображения методом обратной трассировки лучей, но я выделил для себя основные:

- BRL-CAD
- Blender

1.1.1 BRL-CAD

BRL-CAD представляет собой мощную 3D систему автоматизированного проектирования составных объемных тел методом конструктивной блочной геометрии. Включает в себя интерактивный геометрический редактор, параллельную трассировку лучей, рендеринг и геометрический анализ.

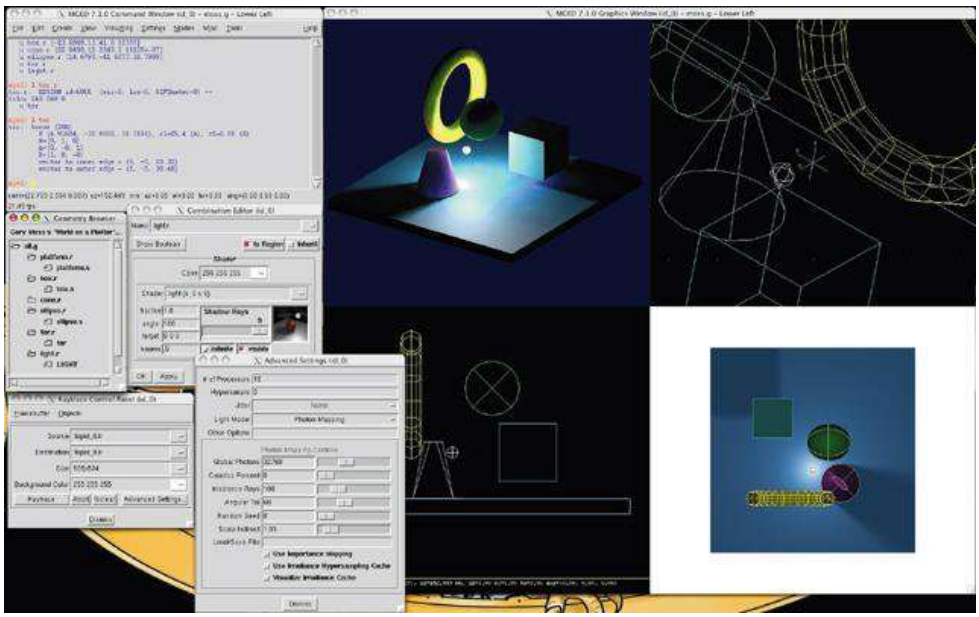


Рисунок 1.1 – Интерфейс программы BRL-CAD.

Недостатки: большой объем программы, непростой интерфейс для неподготовленного пользователя.

Преимущества: кроссплатформенность, регулярные обновления, большие возможности для подготовленного пользователя.

1.1.2 Blender

Blender – профессиональное свободное программное обеспечение для создания трехмерной графики, включающее в себя множество возможностей, в том числе и рендеринг.

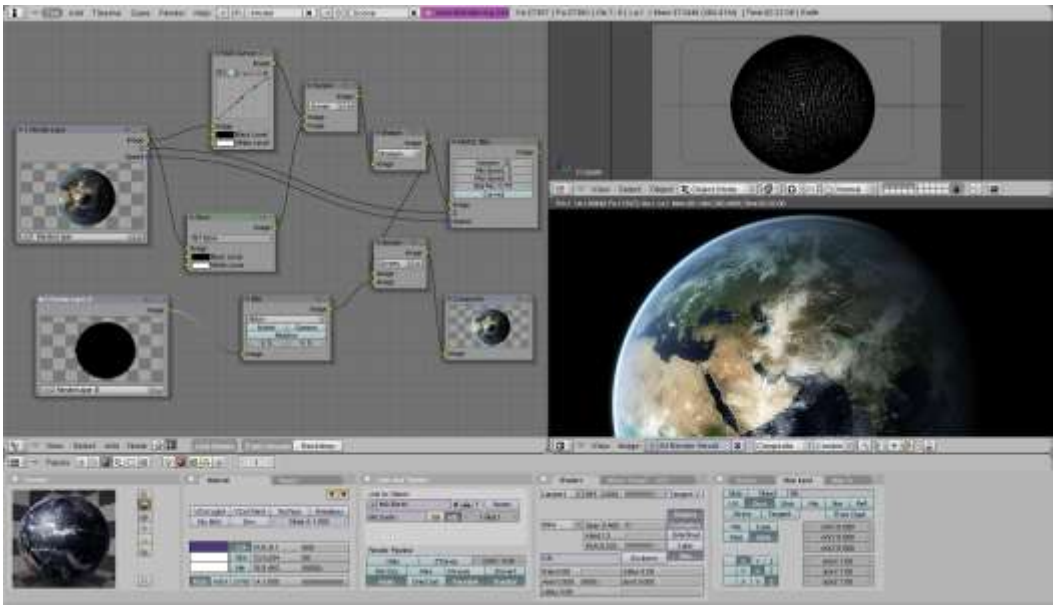


Рисунок 1.2 – Интерфейс программы Blender.

Недостатки: имеет репутацию программы, сложной для изучения, практически каждая функция имеет соответствующее ей сочетание клавиш. Высокие системные требования для комфортной работы с приложением.

Преимущества: характерной особенностью является небольшой размер. Поддержка разнообразных геометрических примитивов, включая полигональные модели, универсальные встроенные механизмы рендеринга.

1.1.3 Вывод

Не смотря на то, что существует немало многофункционального программного обеспечения для синтеза сцены методом обратной трассировки лучей, характерным для него недостатком является сложный и не совсем понятный для неподготовленного пользователя интерфейс. В своем программном продукте я хочу устранить данный недостаток.

1.2 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов.

- Источников света:
    1. Окружающее освещение. Описывается коэффициентом освещенности.
    2. Точечный источник света. Описывается тремя координатами положения и коэффициентом освещенности.
    3. Направленный источник света. Описывается вектором направления и коэффициентом освещенности.
    4. Источник света, учитывающий размеры заданные пользователем, представляющий собой диск, лежащий в плоскости, параллельной плоскости XoZ. Описывается тремя координатами положения центра, коэффициентом освещенности и величиной радиуса.
- Плоскость земли – ограничивающая плоскость. Изначально расположена внизу сцены, параллельна XoZ.
- Объемные тела, каждое из которых характеризуется следующими основными параметрами: координаты центра, цвет, коэффициент диффузного отражения, коэффициент зеркального отражения, коэффициент прозрачности.
- Луч – невидимый объект, который задается параметрическим уравнением P= O + t * direction. Где O – координаты точки начала луча, t - параметр, изменяя который можно получить любую точку на луче, direction – вектор направление луча.

1.3 Обзор существующих методов синтеза изображения

1.3.1 Метод бросания лучей

В основе рейкастинга (ray casting) лежит принцип обратимости световых лучей. Из камеры на каждый пиксель испускаются один луч, и находится ближайший объект, который блокирует путь распространения этого луча. После вычисляется цвет точки попадания и заносится в соответствующий пиксель. Данный метод не позволяет синтезировать сцену с учётом отражения и естественной проекции теней.

1.3.2 Метод конечных элементов

Метод конечных элементов, известный также как radiosity, основан на равновесии обмена света между объектами сцены (другими словами на законе сохранения энергии) и разбиении поверхностей на маленькие участки, которые считаются локально плоскими. Освещенность находится решением системы линейных уравнений, описывающих обмен энергии между участками. При использовании все меньших участков результат будет стремиться к реальной физической модели. Алгоритмы является довольно медленным, так как производится расчет для множества маленьких участков поверхностей. Подобные алгоритмы имеют проблемы с представлением резких теней.

1.3.3 Алгоритм обратной трассировки лучей

В основе алгоритма обратной трассировки лучей лежит принцип обратимости световых лучей, то есть вместо просчёта всех лучей испускаемых из источников сцены, можно изначально испускать первичные лучи из камеры, что значительно повышает эффективность визуализации. При попадании первичного луча в объект вычисляется цвет точки попадания. Для этого создаются вторичные лучи в зависимости от типа поверхности объекта.

1.3.4 Вывод

Главный алгоритм, используемый в программе для синтеза изображения, это алгоритм обратной трассировки лучей. Выбор был сделан в его пользу, так

как в сравнении с его аналогами он даёт наиболее приемлемый результат за небольшое время синтеза. Алгоритм можно модернизировать, добавив в него параллельные вычисления для уменьшения времени синтеза сцены. Также он позволяет строить качественные тени с учетом большого числа источников света.

Достоинством алгоритма является то, что он не требователен к памяти. А недостатком является то, что он не позволяет строить изображения в реальном времени.

1.4 Модели освещения

1.4.1 Модель освещения Ламберта

Локальная модель освещения Ламберта моделирует идеальное диффузное отражение от поверхности. Это означает, что свет при попадании на поверхность рассеивается равномерно во все стороны с одинаковой интенсивностью.

$$I = I_A + I_m \frac{\langle \vec{N}, \vec{L} \rangle}{|\vec{N}||\vec{L}|}$$

где I_m – интенсивность источника m,

I_A – интенсивность окружающего освещения,

$\vec{L}$ – направление на источник света,

$\overrightarrow{N}$ – нормаль в данной точке.

Имеется простая зависимость между силой света, излучаемого плоской рассеивающей площадкой dS в каком-либо направлении от угла α между этим направлением и перпендикуляром к dS.

$$I_\alpha = I_0 \cos(\alpha)$$

Последнее выражение означает, что сила света плоской поверхности максимальна (I_0) по перпендикуляру к ней и, убывая с увеличением α, становится равной нулю в касательных к поверхности направлениях.

1.4.2 Модель освещения Фонга

Модель Фонга – модель освещения, представляющая собой комбинацию диффузной составляющей и зеркальной составляющей, и работает таким образом, что кроме равномерного освещения на материале может появляться блик.

$$I = I_A + I_m \frac{\langle \vec{N}, \vec{L} \rangle}{|\vec{N}||\vec{L}|} + I_m \left(\frac{\langle \vec{R}, \vec{V} \rangle}{|\vec{R}||\vec{V}|} \right)^{\beta}$$

где I_m – интенсивность источника m,

I_A – интенсивность окружающего освещения,

$\vec{L}$ – направление на источник света,

$\overrightarrow{N}$ – нормаль в данной точке,

$\vec{R} = 2\vec{N}\langle \overrightarrow{N}, \vec{L} \rangle - \vec{L}$,

$\vec{V}$ – вектор направления на наблюдателя,

β – коэффициент блеска.

1.4.3 Вывод

Для синтеза сцены с помощью обратной трассировки лучей была выбрана модель освещения Фонга, поскольку она неплохо аппроксимирует физическую модель освещения и не требует много вычислительных мощностей.

1.5 Описание трехмерных преобразований сцены

Координаты точки представлены в однородных координатах в виде вектора-строки $M(x, y, z)$.

Сдвиг точки:

$$\begin{cases} X = x + dx \\ Y = y + dy \\ Z = z + dz \end{cases}$$

Масштабирование относительно начала координат:

$$\begin{cases} X = x \cdot k_x \\ Y = y \cdot k_y \\ Z = z \cdot k_z \end{cases}$$

Любое вращение в трёхмерном пространстве может быть представлено как композиция поворотов вокруг трёх ортогональных осей (например, вокруг осей декартовых координат). Этой композиции соответствует матрица, равная произведению соответствующих трёх матриц поворота.

Матрицами поворота вокруг оси декартовой системы координат на угол α в трёхмерном пространстве с неподвижной системой координат являются:

Матрица поворота относительно оси OX на заданный угол α:

$$M_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

Матрица поворота относительно оси OY на заданный угол α:

$$M_y(\alpha) = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix}$$

Матрица поворота относительно оси OZ на заданный угол α:

$$M_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

1.6 Выводы из аналитического раздела

В данном разделе был рассмотрен алгоритм обратной трассировки лучей для синтеза сцены. Были изучены его преимущества и недостатки, проведено сравнение с алгоритмом Z-буфера. Также были определены и формализованы объекты сцены, преобразования сцены с объектами, выбрана модель освещения.

2. Конструкторская часть

2.1 Описание алгоритма трассировки лучей.

Методы трассировки лучей на сегодняшний день считаются наиболее мощными методами создания реалистических изображений. Универсальность методов трассировки в значительной степени обусловлена тем, что в их основе лежат простые и ясные понятия, отражающие наш опыт восприятия окружающего мира.

Рассмотрим, как формируется изображение. Оно получается из-за того, что свет попадает в камеру. Выпустим из источников света множество лучей. Назовем их первичными лучами. Часть этих лучей улетит в свободное пространство, а часть попадет на объекты. На них лучи могут отразиться, при этом часть энергии луча поглотится. Отраженные лучи образуют множество вторичных лучей. Далее эти лучи опять же отразятся и образуют новое поколение лучей. В конечном итоге часть лучей попадет в камеру и сформирует изображение.

Существуют алгоритмы, работающие по такому алгоритму. Но они крайне неэффективны, так как большинство лучей, исходящих из источника, не попадают в камеру. А приемлемая картинка получается, если трассировать большое число лучей, что займет очень много времени. Данный алгоритм называется прямой трассировкой лучей.

Метод обратной трассировки лучей позволяет значительно сократить перебор световых лучей. Этот метод разработали в 80-х годах Уиттед и Кэй. В этом методе отслеживаются лучи не от источников, а из камеры. Таким образом, трассируется определенное число лучей, равное разрешению картинки.

Предположим, что у нас есть камера и экран, находящийся на расстоянии d от нее. Разобьем экран на квадратики. Дальше будем по очереди проводить лучи из камеры в центр каждого квадратика (первичные лучи). Найдем пересечение каждого такого луча с объектами сцены и выберем среди всех пересечений самое близкое к камере. Далее, применив нужную модель

освещения, можно получить изображение сцены. Это самый простой метод трассировки лучей. Он позволяет лишь отсечь невидимые грани.

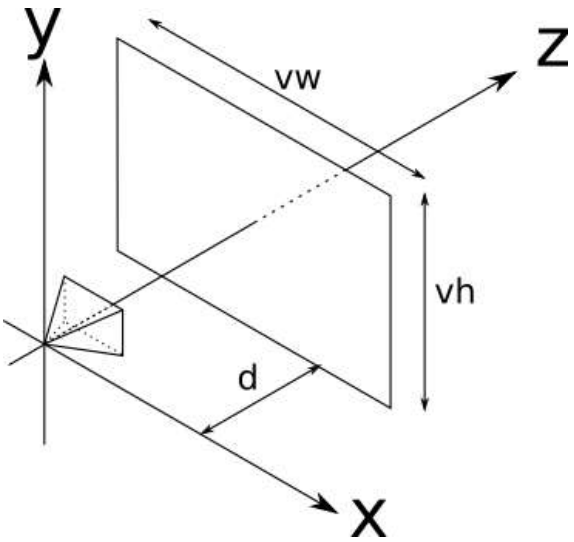


Рисунок 2.1 – Расположение камеры и сцены

Но можно пойти дальше. Если мы хотим смоделировать такое явление, как отражение, нам необходимо из самого близкого пересечения пустить вторичные лучи. Например, если поверхность отражает свет и она идеально ровная, то необходимо отразить первичный луч от поверхности и пустить по этому направлению вторичный луч.

О прозрачности. Когда луч падает на прозрачную поверхность, мы пускаем вторичный луч и вычисляем дополнительный цвет — цвет света, проходящего *сквозь* объект.

2.2 Поиск пересечений с объектами

Наилучшим способом представления лучей для нашей цели будет использование параметрического уравнения. Мы знаем, что луч проходит через O (точка, в которой находится камера), и мы знаем его направление (из O в V), поэтому мы можем выразить любую точку P луча как

$$P = O + t(V - O)$$

где t — произвольное действительное число. Давайте обозначим $(V - O)$, то есть направление луча, как $\vec{D}$ тогда уравнение примет простой вид

$$P = O + t\vec{D} \quad (1)$$

2.2.1 Барицентрические координаты

Барицентрические координаты особенно важны в компьютерной графике. У них есть несколько функций, и они являются ключом к текстурированию, а также к алгоритму пересечения луча и треугольника, предложенному Мёллер-Трумбором.

Барицентрические координаты могут использоваться для выражения положения любой точки, расположенной на треугольнике, с тремя скалярами. Расположение этой точки включает в себя любое положение внутри треугольника, любое положение на любом из трех краев треугольников или любую из самих вершин трех треугольников.

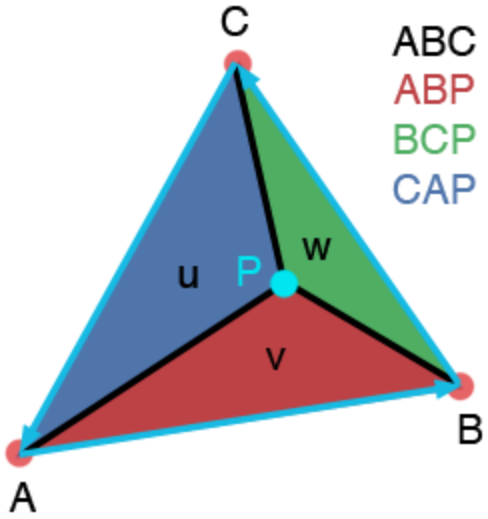


Рисунок 2.2 – Барицентрические координаты в треугольнике

Чтобы вычислить положение этой точки с использованием барицентрических координат, используем следующее уравнение:

$$P = uA + vB + wC$$

где A, B и C – вершины треугольника,

u, v, w – барицентрические координаты (три действительных числа).

Барицентрические координаты нормированы, то есть $u + v + w = 1$. Если какая-либо из координат меньше нуля или больше единицы, точка находится за пределами треугольника. Если любая из них равна нулю, P находится на одной из прямых, соединяющих вершины треугольника.

Барицентрические координаты также известны как *площадные координаты*. Этот термин указывает на то, что координаты u, v и w пропорциональны площади трех подтреугольников, определяемых буквой P, точкой, расположенной на треугольнике, и вершинами треугольника (A, B, C). Эти три подтреугольника обозначаются ABP, BCP, CAP.

Это приводит нас к формулам, используемым для вычисления барицентрических координат:

$$u = \frac{TriangleCAP_{Area}}{TriangleABC_{Area}}$$

$$v = \frac{TriangleABP_{Area}}{TriangleABC_{Area}}$$

$$w = \frac{TriangleBCP_{Area}}{TriangleABC_{Area}}$$

Теперь вычислить площадь треугольника несложно. Если продублировать треугольник и отразите его по самому длинному краю, получим параллелограмм.

$$Parallelogram_{Area} = \|(B - A)(C - A)\|$$

$$Triangle_{Area} = \frac{Parallelogram_{Area}}{2}$$

2.2.2 Алгоритм Мёллера-Трумбора

Алгоритм Мёллера-Трумбора – это алгоритм быстрого поиска пересечения лучей и треугольников. Пусть P – точка пересечения луча и треугольника, тогда:

$$P = wA + vB + vC$$

Также известно, что $w = 1 - u - v$, поэтому можно сделать замену:

$$P = (1 - u - v)A + vB + vC$$

Раскрыв скобки и приведя слагаемые, получим:

$$P = A - uA - vA + uB + vC = A + u(B - A) + v(C - A)$$

Используя уравнение (1) получим:

$$O - A = -tD + u(B - A) + v(C - A)$$

Справа от знака равенства у нас 3 неизвестных(t, u, v). Преобразуем эту часть в умножение вектора на матрицу.

$$[-D\ (B - A)\ (C - A)] \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - A$$

Воспользуемся правилом Крамера, для решения данного уравнения:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{[|-D\ E_1\ E_2|]} \begin{bmatrix} |\ T & E_1 & E_2\ | \\ |-D & T & E_2\ | \\ |\ -D & E_1 & T\ | \end{bmatrix}$$

где $T = O - AE_1 = B - A$,

$E_2 = C - A$.

Определитель (матрицы 3x3) - это не что иное, как тройное скалярное произведение. С учетом этого:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D \times E_2) \cdot E_1} \begin{bmatrix} (T \times E_1) \cdot E_2 \\ (D \times E_2) \cdot T \\ (T \times E_1) \cdot D \end{bmatrix} == \frac{1}{P \cdot E_1} \begin{bmatrix} Q \cdot E_2 \\ P \cdot T \\ Q \cdot D \end{bmatrix}$$

где $P = (D \times E_2)$,

$Q = (T \times E_1)$.

Откуда и получаем значения t, u, v.

2.2.3 Пересечение луча и сферы

Что такое сфера? Сфера — это множество точек, лежащих на постоянном расстоянии (называемом *радиусом* сферы) от фиксированной точки (называемой *центром* сферы):

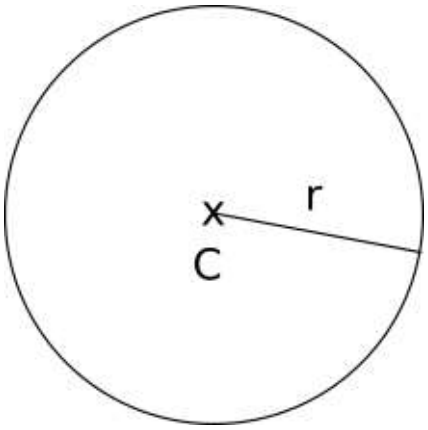


Рисунок 2.3 – Изображение сферы.

Если C — центр сферы, а r — радиус сферы, то точки P на поверхности сферы удовлетворяют следующему уравнению:

$$distance(P, C) = r$$

Расстояние между P и C- это длина вектора из P в C:

$$|P - C| = r$$

Длина вектора — это квадратный корень его скалярного произведения на себя:

$$\sqrt{\langle P - C, P - C \rangle} = r$$

Или

$$\langle P - C, P - C \rangle = r^2 \ (2)$$

Поскольку P — это одна и та же точка в обоих уравнениях, мы можем заменить P в первом на выражение для P во втором. Это даёт нам:

$$t^2\langle\vec{D},\vec{D}\rangle + t\big(2\langle\overrightarrow{OC},\vec{D}\rangle\big) + \langle\overrightarrow{OC},\overrightarrow{OC}\rangle - r^2 = 0$$

Это квадратное уравнение, тогда его решения

$$t_1, t_2 = \frac{-k_2 \pm \sqrt{{k_2}^2 - 4k_1k_3}}{2k_1}$$

Это точно соответствует случаям, когда луч не пересекает сферу, луч касается сферы и луч входит и выходит из сферы:

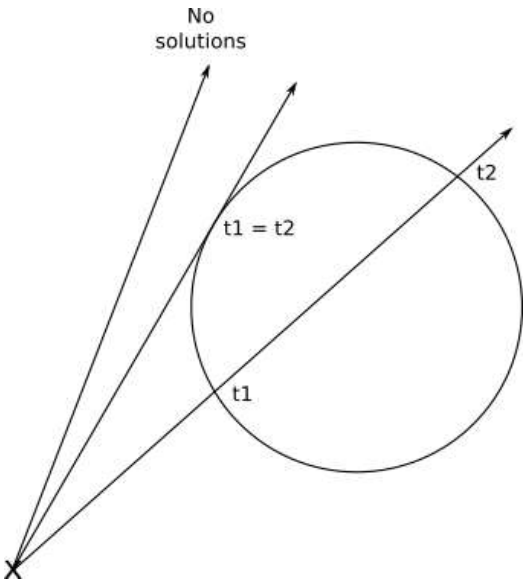


Рисунок 2.4 – Возможные случаи пересечения луча и сферы

2.2.4 Пересечение луча и плоскости

Пусть $\vec{n}$ – нормаль к плоскости, $\vec{q}$ – точка на плоскости. Тогда точка $\vec{p}$ принадлежит плоскости, если:

$$\vec{n}(\vec{p} - \vec{q}) = 0,$$

Подставим уравнение луча:

$$\vec{n}(\vec{o} + \vec{d}t - \vec{q}) = 0$$

$$t = \frac{\vec{n}(\vec{q} - \vec{o})}{\vec{d}\vec{n}}$$

Если $\vec{d}\vec{n} = 0$, то луч параллелен плоскости и точки пересечения нет или их бесконечного много. Если $t < 0$, то точки пересечения нет, в ином случае $\vec{o} + \vec{d}t -$ искомая точка пересечения.

2.3 Освещение

Для начала необходимо указать некоторые допущения. Во-первых, считается, что всё освещение имеет белый цвет. Это позволит охарактеризовать любой источник освещения единственным действительным числом i, называемым *яркостью* освещения.

Во-вторых, атмосфера не будет учитываться при расчете интенсивности. Это значит, что освещение не становятся менее яркими, независимо от расстояния.

Рассмотрим все типы моделируемых источников освещения.

2.3.1 Точечный источник освещения

Точечный источник испускает свет из фиксированной точки в пространстве, называемой его позицией. Свет испускается равномерно во всех направлениях.

Следовательно, точечный источник полностью характеризуется его позицией и интенсивностью света.

Давайте зададим вектор $\vec{L}$ как направление из точки P в сцене к источнику освещения Q. Этот вектор, называемый световым вектором, просто равен $Q - P$. Поскольку Q фиксирована, а P может быть любой точкой сцены, то в общем случае $\vec{L}$ будет разным для каждой точки сцены.

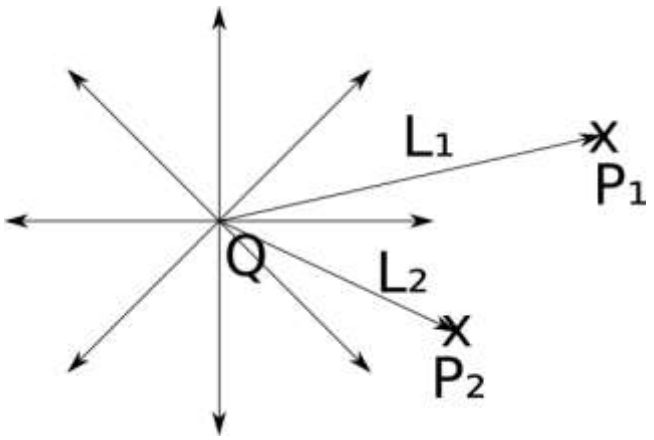


Рисунок 2.5 – Точечный источник освещения

2.3.2 Направленный источник освещения

Для аппроксимации Солнца мы зададим *направленный источник освещения*. Как и точечный источник, направленный источник имеет интенсивность, но он не имеет позиции. Вместо позиции у него задано *направление освещения*.

Можно воспринимать его как бесконечно удалённый точечный источник, светящий в определённом направлении. В случае точечных источников нам нужно вычислять новый световой вектор $\vec{L}$ для каждой точки P сцены, но в этом случае $\vec{L}$ уже задан.

2.3.3 Окружающий источник освещения

Третий тип источников освещения, называется *окружающим освещением*, которое характеризуется только интенсивностью. Считается, что оно носит безусловный вклад освещения в каждую точку сцены. Это упрощение чрезвычайно сложного взаимодействия между источниками освещения и поверхностями сцены, но оно передает правильную и реалистичную картину.

2.3.4 Дисковый источник освещения

Данный тип источников освещения используется для визуализации так называемых мягких теней. Характерной его особенностью является то, что он

имеет определенные размеры, то есть не является материальной точкой, а также параллелен оси XoZ.

Дисковый источник характеризуется позицией, интенсивностью освещения и величиной радиуса диска.

Принцип расчета интенсивности довольно простой: при обработке данной точки сцены, необходимо взять случайную точку на дисковом источнике и построить вектор до неё, вычислить интенсивность. Операция продолжается до тех пор, пока не будет достигнуто нужное качество тени.

2.4 Моделирование диффузного отражения

Когда луч света падает на матовый объект, то из-за неровности его поверхности на микроскопическом уровне, он отражает луч в сцену равномерно во всех направлениях, то есть получается «рассеянное» («диффузное») отражение.

С другой стороны, количество отражённого света зависит от угла между лучом света и поверхностью. Интуитивно это понятно – энергия, переносимая лучом, в зависимости от угла должна распределиться по меньшей или большей поверхности, то есть энергия на единицу площади, отражённая в сцену, будет соответственно выше или ниже:

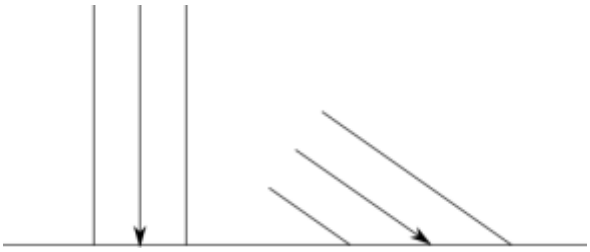


Рисунок 2.6 – Падение светового луча

Итак, луч света с направлением $\vec{L}$ и яркостью I падает на поверхность с нормалью $\vec{N}$. Какая часть I отражается обратно в сцену как функция от $I, \overrightarrow{N}, \vec{L}$?

Для геометрической аналогии представим яркость света как «ширину» луча. Его энергия распределяется по поверхности размером A. Когда $\vec{N}$ и $\vec{L}$ имеют одно направление, то есть луч перпендикулярен поверхности, $I =$

A, а это значит, что энергия, отражённая на единицу площади равна падающей энергии на единицу площади; $< \frac{I}{A} = 1$.

С другой стороны, когда угол между $\vec{N}$ и $\vec{L}$ приближается к 90°, $\lim_{A\to\infty} \frac{I}{A} = 0$. Осталось понять происходит в промежутках. Ситуация отображена на схеме ниже, Мы знаем $\vec{N}$, $\vec{L}$ и P. Нам нужно вычислить $\frac{I}{A}$.

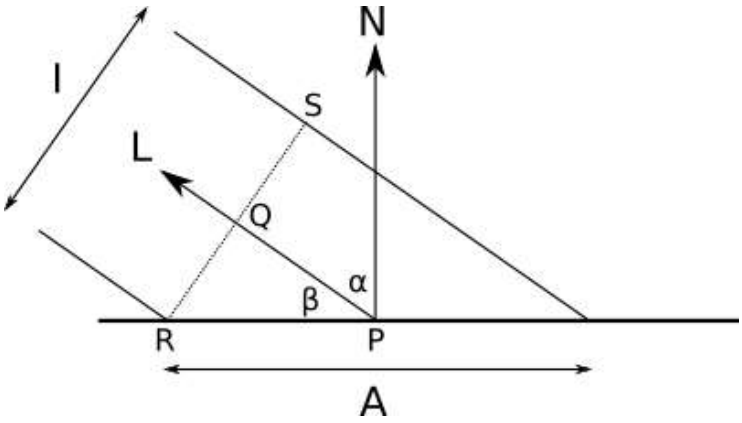


Рисунок 2.7 – Падение луча на поверхность под углом

Поскольку технически луч света не имеет ширины, поэтому мы будем считать, что всё происходит на бесконечно малом плоском участке поверхности. Пусть SR – "ширина луча", по определению она перпендикулярна $\vec{L}$, который также является направлением PQ. Значит треугольник PQR – прямоугольный. Очевидно, что угол $QRP = \alpha$. Рассмотрим треугольник PQR: сторона QR равна $\frac{I}{2}$, а PR равна $\frac{A}{2}$. По определению

$$\cos(\alpha) = \frac{QR}{PR} = \frac{\frac{I}{2}}{\frac{A}{2}} = \frac{I}{A}$$

α – это угол между $\vec{N}$ и $\vec{L}$, то есть его можно вычислить как

$$\cos(\alpha) = \frac{\langle \vec{N}, \vec{L} \rangle}{|\vec{N}||\vec{L}|}$$

Итого

$$\frac{I}{A} = \frac{\langle \vec{N}, \vec{L} \rangle}{|\vec{N}||\vec{L}|}$$

Получено уравнение, связывающее отражённую часть света с углом между нормалью к поверхности и направлением света.

Заметим, что при углах больше $90°$ значение $\cos(\alpha)$ становится отрицательным. Это не имеет никакого физического смысла - угол больше $90°$ просто означает, что свет на самом деле достигает задней части поверхности, и не вносит свой вклад в освещение освещаемой точки. То есть если $\cos(\alpha)$ становится отрицательным, то считаем его равным 0.

2.4.1 Уравнение диффузного отражения

Теперь можно сформулировать уравнение для вычисления полного количества света, полученного точкой P с нормалью $\vec{N}$ в сцене с окружающим освещением яркостью I_A и n точечных или направленных источников света с яркостью I_n и световыми векторами $\overrightarrow{L_n}$ или известными (для направленных источников), или вычисленными для P (для точечных):

$$I_p = I_A + \sum_{i=1}^{n} I_i \frac{\langle \vec{N}, \overrightarrow{L_i} \rangle}{|\vec{N}||\vec{L}_i|}$$

Стоит снова отметить, что члены, в которых $\langle \vec{N}, \overrightarrow{L_i} \rangle < 0$ не должны прибавляться к освещённости точки.

2.5 Моделирование зеркального отражения

Для идеально отполированного зеркала падающий луч света $\vec{L}$ отражается в единственном направлении $\vec{R}$. Именно это позволяет нам чётко видеть объекты в зеркале: для каждого падающего луча $\vec{L}$ есть единственный отражённый луч $\vec{R}$.

Но не каждый объект отполирован идеально, хотя большая часть света отражается в направлении $\vec{R}$, часть его отражается в направлениях, близких к $\vec{R}$. Чем ближе к $\vec{R}$, тем больше света отражается в этом направлении. «Блеск» объекта определяет то, насколько быстро отражённый свет уменьшается при отдалении от $\vec{R}$:

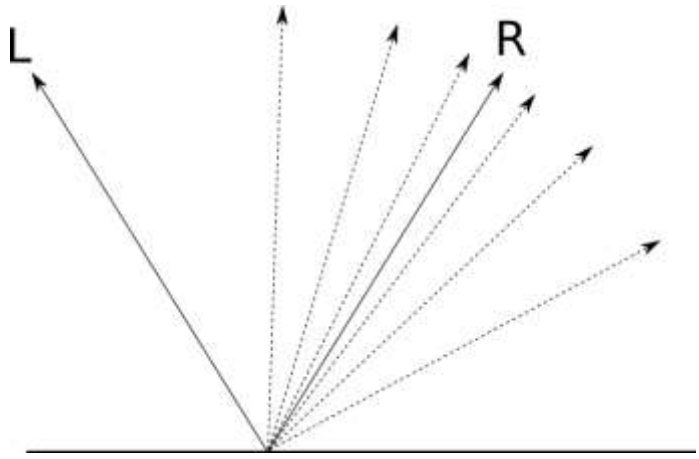


Рисунок 2.8 – Отражение света

Необходимо выяснить, какое количество света от $\vec{L}$ отражается обратно в направлении нашей точки обзора (потому что это свет, который мы используем для определения цвета каждой точки). Если $\vec{V}$ — это «вектор обзора», указывающий из P в камеру, а α — угол между $\vec{R}$ и $\vec{V}$, то получается следующее:

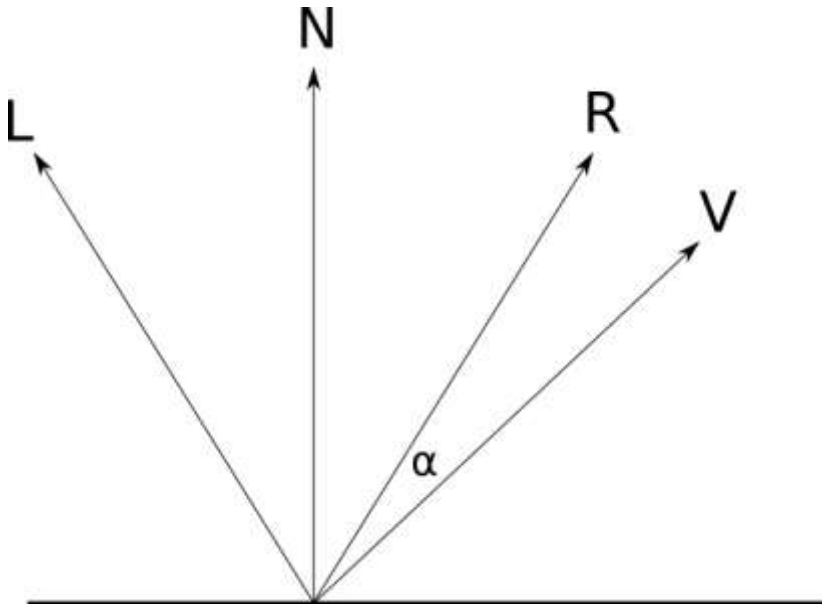


Рисунок 2.9 – Зависимость угла падения

При $\alpha = 0°$ отражается весь свет. При $\alpha = 90°$ свет не отражается. Как и в случае с диффузным отражением, нужно вывести математическое выражение для определения того, что происходит при промежуточных значениях α.

2.5.1 Уравнение зеркального отражения

Блеск — мера того, насколько быстро функция отражения уменьшается при увеличении α. Очень простой способ получения различных кривых блеска заключается в вычислении степени $\cos(\alpha)$ некоего положительного показателя s.

Поскольку $0 \leq cos(\alpha) \leq 1$, то очевидно, что $0 \leq \cos(\alpha)^s \leq 1$. Вот график $\cos(\alpha)^s$ для разных значений s:

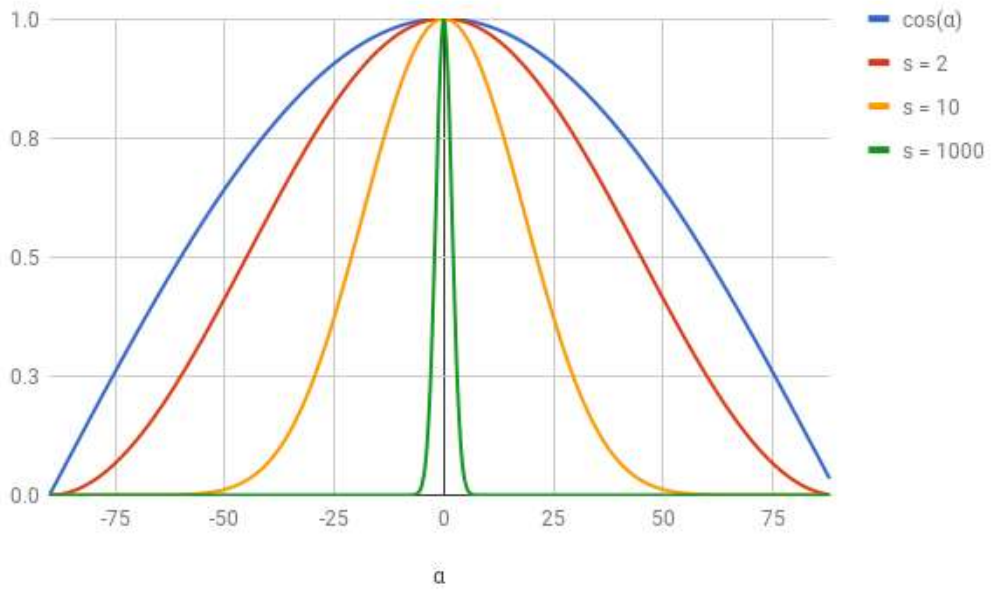


Рисунок 2.10 – График $\cos(\alpha)^s$ при различных s

Коэффициент s называют показателем отражения, он является свойством поверхности.

Получаем, что луч $\vec{L}$ падает на поверхность в точке P, где нормаль $\vec{N}$, а показатель отражения – s. Какое количество света отразится в направлении обзора $\vec{V}$?

Это значение равно $\cos(\alpha)^s$, где α — это угол между $\vec{V}$ и $\vec{R}$. Вектор $\vec{R}$ является вектором $\vec{V}$ отражённым относительно $\vec{N}$. То есть первым шагом будет вычисление вектора $\vec{R}$ из векторов $\vec{N}$ и $\vec{L}$.

Разложим вектор $\vec{L}$ на два вектора $\vec{L}_p$ и $\vec{L}_N$, таких, что $\vec{L} = \vec{L}_p + \vec{L}_N$, где $\vec{L}_N$ параллелен $\vec{N}$, а $\vec{L}_p$ перпендикулярен $\vec{N}$:

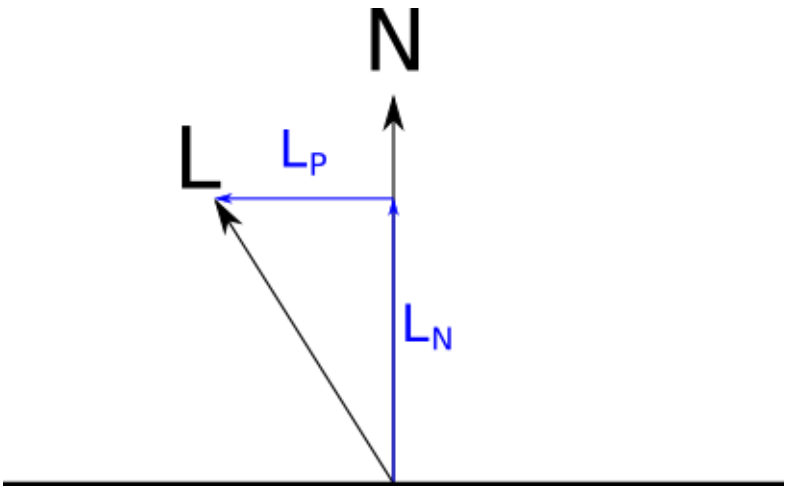


Рисунок 2.11(а) – Разложение вектора $\vec{L}$

Вектор $\vec{L}_N$ — это проекция $\vec{L}$ на $\vec{N}$, по свойствам скалярного произведения и исходя из того, что $|\overrightarrow{N}| = 1$, длина этой проекции равна $\langle \vec{N}, \vec{L} \rangle$.

Определено, что $\vec{L}_N$ будет параллелен $\vec{N}$, поэтому $\vec{L}_N = \vec{N}\,\langle \vec{N}, \vec{L} \rangle$. Поскольку $\vec{L} = \vec{L}_p + \vec{L}_N$, можно сразу получить $\vec{L}_p = \vec{L} - \vec{L}_N = \vec{L} - \vec{N}\langle \vec{N}, \vec{L} \rangle$.

Теперь посмотрим на вектор $\vec{R}$, поскольку он симметричен вектору $\vec{L}$ относительно $\vec{N}$, его компонент, параллельный $\vec{N}$, тот же, что и у $\vec{L}$, а перпендикулярный компонент противоположен компоненту $\vec{L}$. То есть $\vec{R} = \vec{L}_N - \vec{L}_P$:

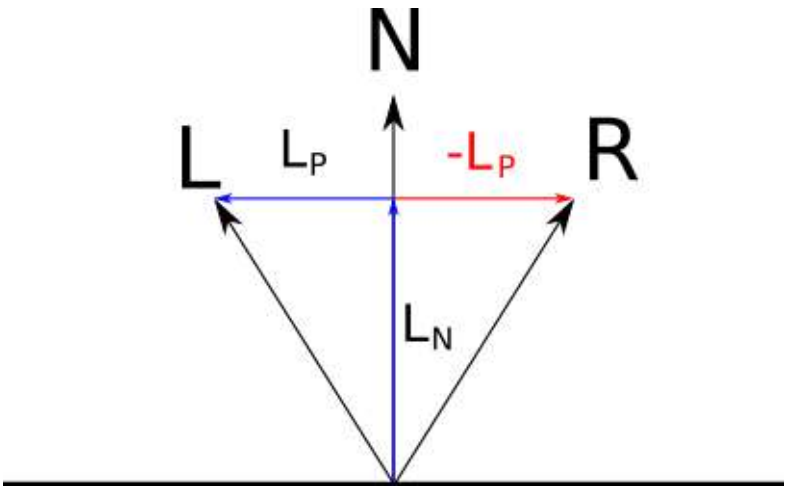


Рисунок 2.12(б) – Разложение вектора $\vec{L}$

Подставляя полученные ранее выражения, мы получим

$$\vec{R} = \vec{N}\langle\vec{N},\vec{L}\rangle - \vec{L} + \vec{N}\langle\vec{N},\vec{L}\rangle$$

и немного упростив, получаем

$$\vec{R} = 2\vec{N}\langle\vec{N},\vec{L}\rangle - \vec{L}$$

Теперь мы готовы записать уравнение «зеркального» отражения:

$$\vec{R} = 2\vec{N}\langle\vec{N},\vec{L}\rangle - \vec{L}$$

$$I_S = I_L\left(\frac{\langle\vec{R},\vec{V}\rangle}{|\vec{R}||\vec{V}|}\right)$$

Как и в случае диффузного освещения, cos(α) может быть отрицательным, и мы снова должны это игнорировать. Кроме того, не каждый объект должен быть блестящим. Для таких объектов значение «зеркальности» вообще не будет вычисляться.

2.6 Тени

Тени появляются там, где есть свет, но его лучи не могут достичь объекта, потому что на их пути есть другой объект.

Можно выделить 2 случая:

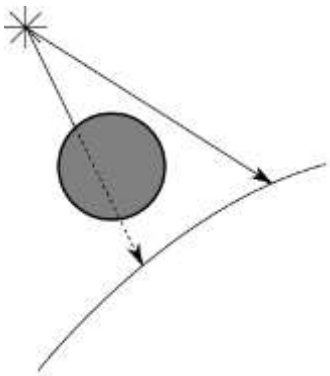


Рисунок 2.13 – Возникновение тени

Начнём с направленного источника. Значение координаты точки P, которая нас интересует, известно, также известен вектор $\vec{L}$ - это часть определения источника освещения.

Имея P и $\vec{L}$, можно задать луч, а именно $P + t\vec{L}$, который проходит из точки до бесконечно отдалённого источника освещения. Пересекает ли этот луч другой объект? Если нет, то между точкой и источником ничего нет, то есть можно вычислить освещённость от этого источника и прибавить его к общей освещённости. Если пересекает, то источник игнорируется.

Важны пересечения со всем после точки P на бесконечное расстояние. Это значит, что $t_{min} = 0$ и $t_{max} = +\infty$.

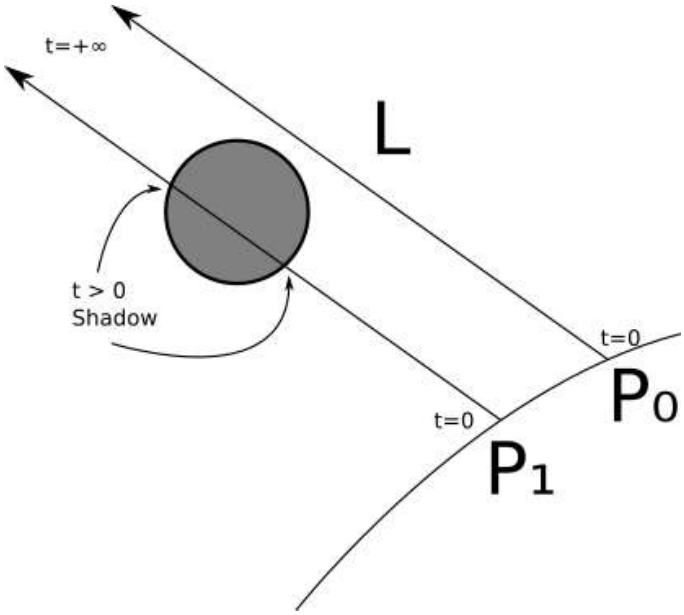


Рисунок 2.14(а) – Испускание лучей

Обработка точечных источников очень похожа, но с двумя исключениями. Во-первых, не задан $\vec{L}$, но его очень просто вычислить из позиции источника и координат P. Во-вторых, важны любые пересечения, начиная с P, но только до $\vec{L}$ (в противном случае, объекты за источником освещения могли бы создавать тени). То есть в этом случае $t_{min} = 0$ и $t_{max} = 1$.

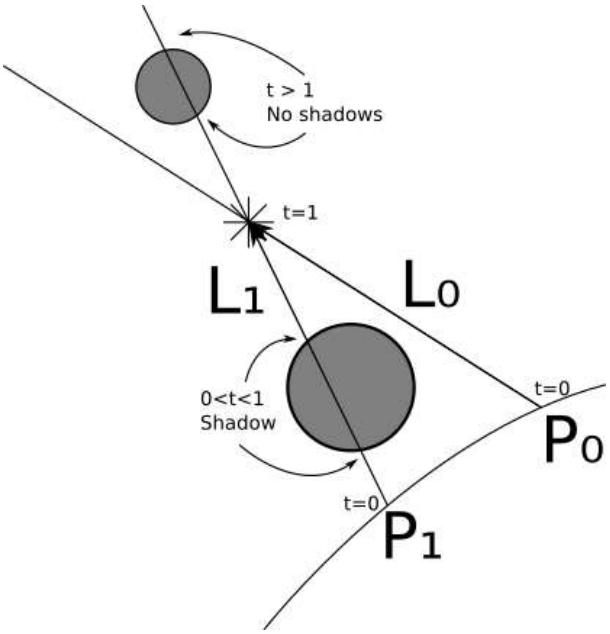


Рисунок 2.14(б) – Испускание лучей

Существует один пограничный случай, который нужно рассмотреть. Возьмём луч $P + t\vec{L}$. Если будем искать пересечения, начиная с $t_{min} = 0$, то вероятнее всего, найдём саму P при $t = 0$, потому что P действительно находится на сфере, и $P + 0 * \vec{L} = P$. Другими словами, каждый объект будет отбрасывать тени на самого себя.

Простейший способ справиться с этим — использовать в качестве нижней границы значений t вместо 0 малое значение є. Геометрически луч начинается немного вдали от поверхности, то есть рядом с P, но не точно в P. То есть для направленных источников интервал будет $[\epsilon, +\infty]$, а для точечных — $[\epsilon, 1]$.

2.6.1 Мягкие тени

Для моделирования мягких теней используется дисковый источник освещения. Принцип вычисления не сильно отличается от расчета тени точечного источника. При обработке данной точки сцены, необходимо взять случайную точку на дисковом источнике и построить вектор до неё, вычислить интенсивность. Операция продолжается до тех пор, пока не будет достигнуто нужное качество тени.

2.7 Режим смешивания цветов

Режимы смешивания цветов в компьютерной графике используются для определения того, как два слоя (цвета) смешиваются друг с другом. Поскольку каждый пиксель имеет числовое представление, существует большое количество способов смешивания двух слоев. Рассмотрим один из них.

2.7.1 Alpha смешивание цветов

Альфа смешивание – это выпуклая комбинация двух цветов, обеспечивающая эффекты прозрачности в компьютерной графике. Значение α цветового кода находится в диапазоне от 0.0 до 1.0, где 0.0 представляет собой полностью прозрачный цвет, а 1.0 представляет собой полностью непрозрачный цвет.

Пусть цвет A накладывается на B с прозрачностью α, тогда значение результирующего цвета определяется по формуле:

$$C = \alpha A + (1 - \alpha)B$$

2.8 Схема алгоритма

На рисунке 2.15 приведена общая схема алгоритма обратной трассировки лучей, где определена функция TraceRay.

На рисунке 2.16 приведена схема данной функции.

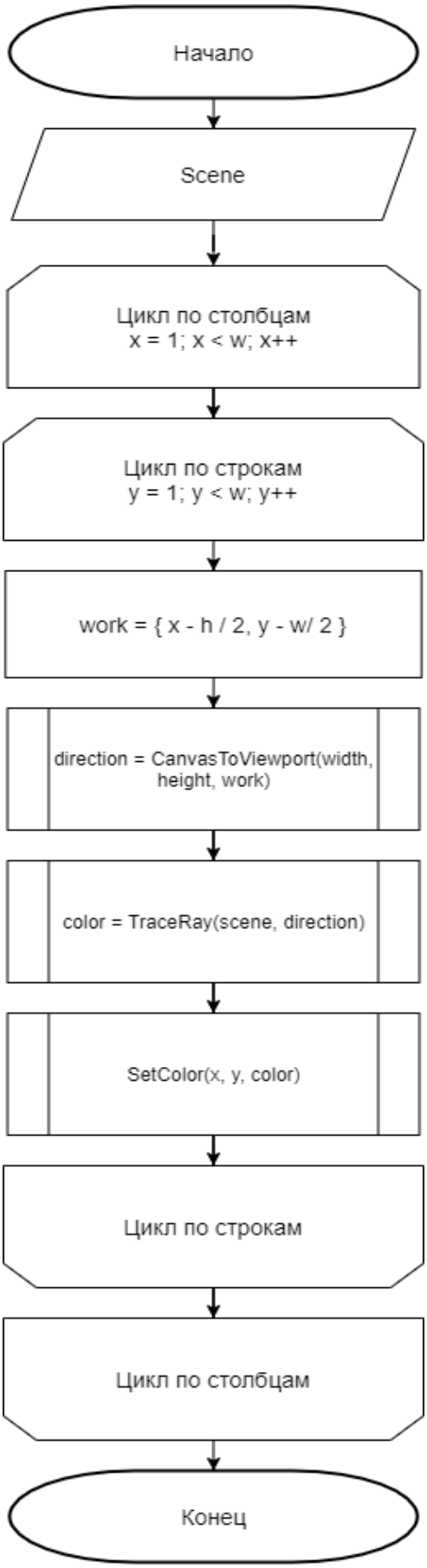


Рисунок 2.15 – Схема общего алгоритма обратной трассировки лучей

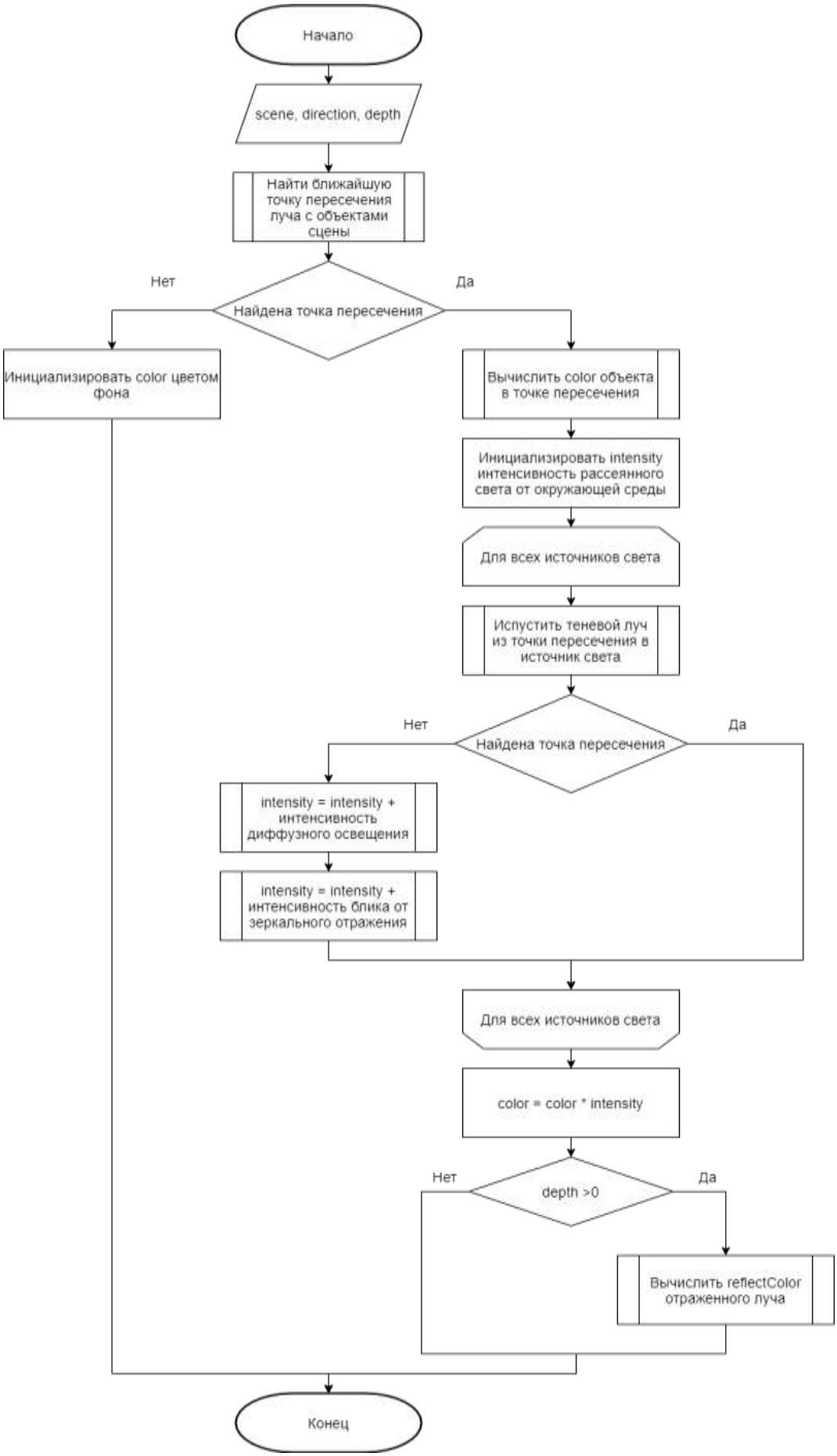


Рисунок 2.16 – Схема алгоритма обратной трассировки лучей

3. Технологическая часть

В данном разделе будут выбраны средства реализации программного обеспечения и структура программы.

3.1 Выбор языка программирования

В качестве языка программирования был выбран C#:

1) является объектно-ориентированным языком программирования;
2) обладает автоматической сборкой мусора;
3) имеется опыт работы на этом языке программирования;
4) обладает удобными средствами для реализации многопоточности.

В качестве среды разработки была выбрана IDE Visual Studio 2017 по следующим причинам:

1) обеспечивает простую и удобную работу с Windows Form;
2) автоматическая сборка проекта на C#;
3) имеется опыт работы в данной среде разработки;
4) бесплатная модель распространения для студентов.

3.2 Структура программы

В этом разделе будут рассмотрены структура и состав классов, разработанного программного продукта.

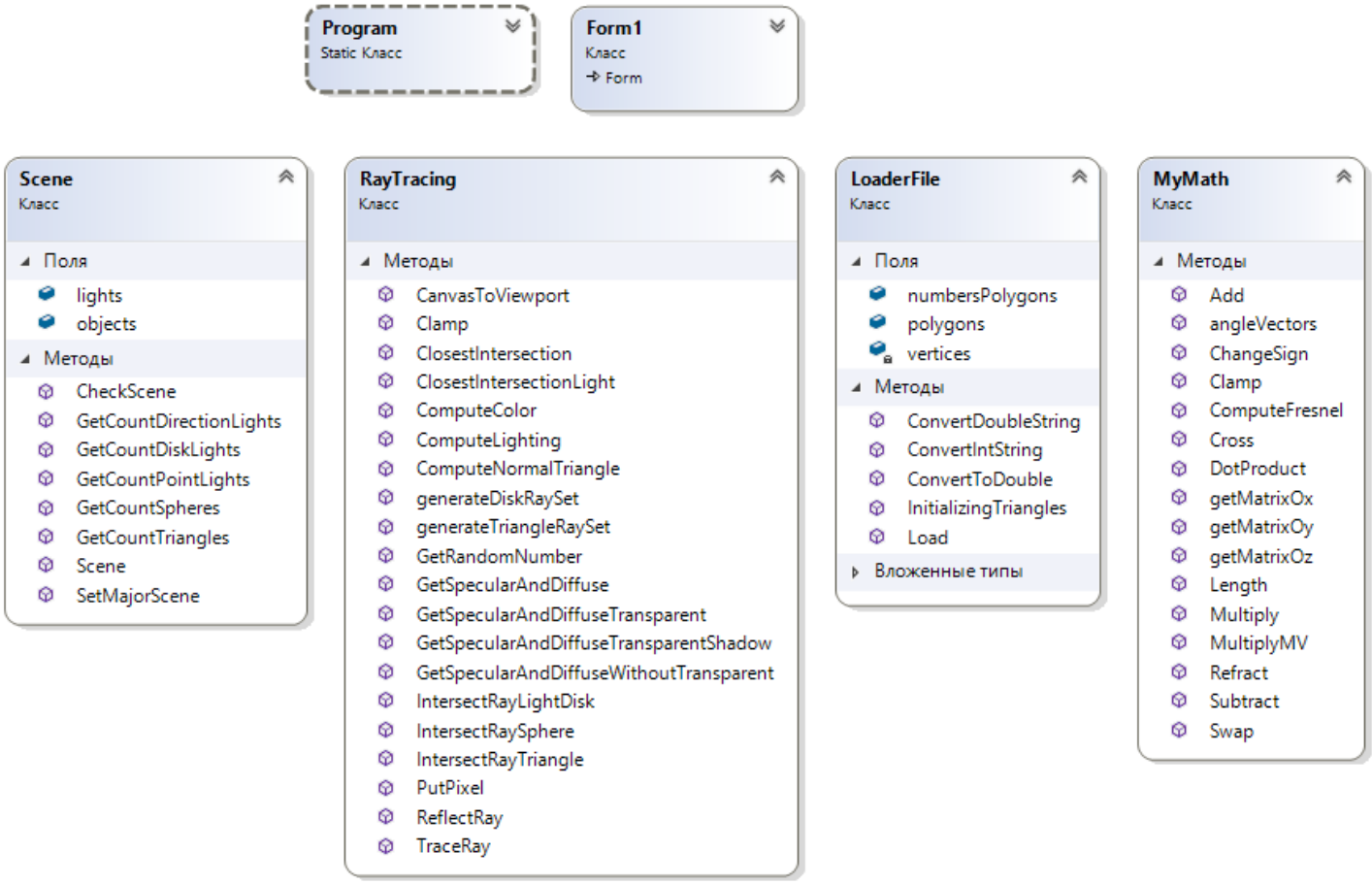


Рисунок 3.1 – Структура классов Program, Form1, Scene, RayTracing, LoaderFile, MyMath

Program – точка входа в программу;

Form1 – класс пользовательского интерфейса;

Scene – хранит информацию о сцене: список источников света, список объектов, имеет методы подсчета количества элементов в списках;

MyMath – содержит в себе методы, необходимые для работы с векторами и точками;

RayTracing – класс вычисления цвета пикселя в заданной точки, содержит необходимые методы;

LoaderFile – класс для загрузки простой obj модели, состоящей из полигонов-треугольников.

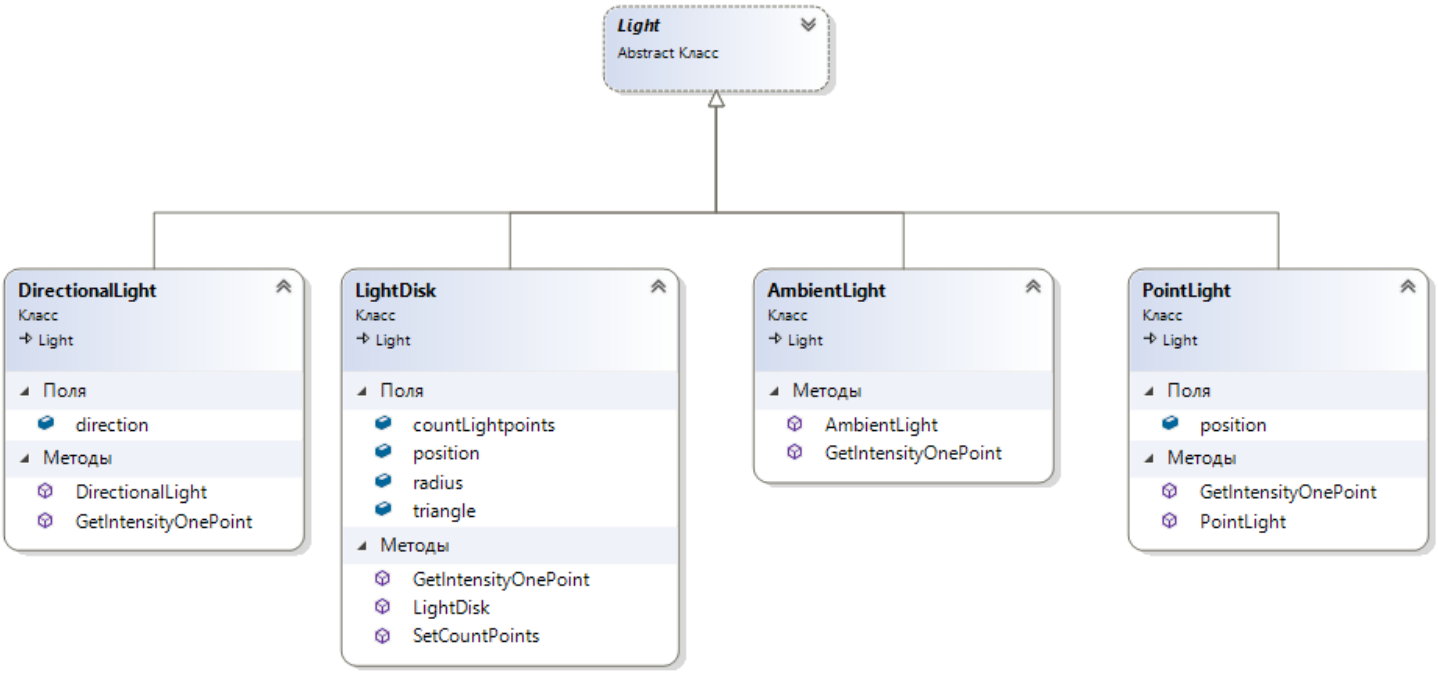


Рисунок 3.2 – Структура производных классов класса Light

Light – абстрактный класс, от которого наследуются все производные классы;

DirectionalLight – класс направленного источника освещения, содержит в себе вектор направления и интенсивность освещения;

LightDisk – класс дискового источника освещения, имеет поля с описанием позиции, величиной радиуса;

AmbientLight – класс окружающего освещения, содержит в себе информацию о интенсивности освещения;

PointLight – класс точеного источника освещения, содержит в себе информацию с описанием позиции, а также интенсивности освещения.

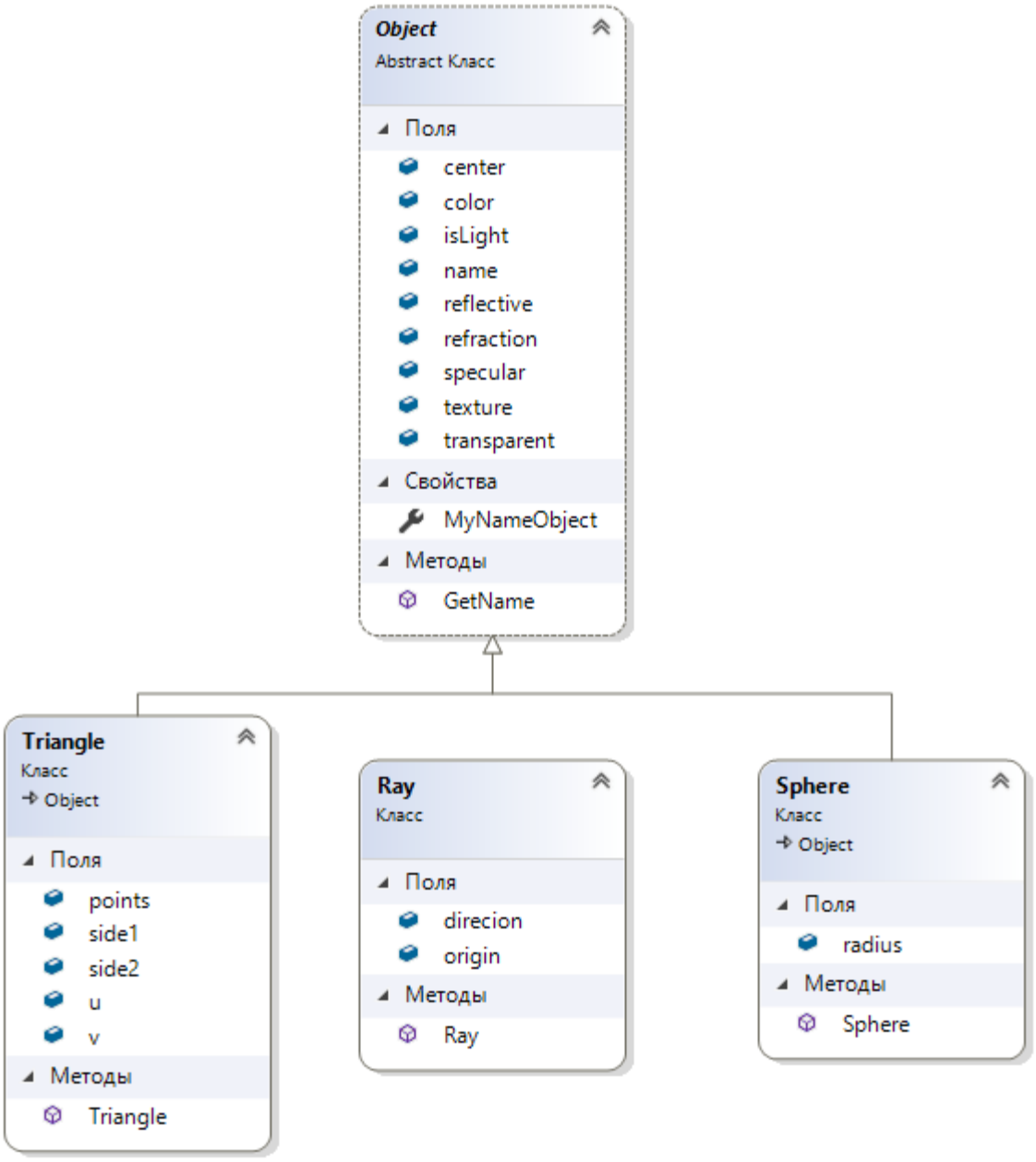


Рисунок 3.3 – Структура производных классов класса Object

Object – абстрактный класс, от которого наследуются все производные;

Triangle – класс объекта треугольник, содержит в себе поля с описанием позиции по трём точкам;

Ray – класс невидимого объекта луч, содержит информацию о векторе направления и точке начала;

Sphere – класс объекта сфера, поля хранят информацию о радиусе и позиции.

3.3 Интерфейс программы

Ниже показано главное окно программы. Пользователю предоставлена возможность добавить готовый объект из списка примитивов (сферу или плоскость-треугольник), а также готовый источник света (точечный, направленный, окружающее освещение, дисковый источник), по нажатию кнопки «Добавить», появится окно с выбором характеристик выбранного объекта.

Пользователь может загружать сложные объекты, состоящие из полигонов-треугольников. Для этого необходимо нажать на кнопку «Загрузить модель», после чего появится окно с выбором нужного файла типа obj. Также имеется возможность установить определенные параметры для загружаемой модели, для этого необходимо нажать на кнопку «Настройка модели», после чего выбрать нужные параметры в появившемся окне выбора.

Для каждого из объектов, а также источников освещения, пользователь может изменять характеристики и положение на сцене. Для этого нужно нажать на имя соответствующего объекта в списке объектов сцены, а затем на кнопку «Изменить», после этого будет открыто соответствующее окно.

С помощью кнопки «Удалить» пользователь может удалить любой объект или источник света сцены. Для этого нужно нажать на имя соответствующего объекта в списке объектов сцены, а затем на кнопку «Удалить».

Кнопка «Начать рендер» запускает процесс синтеза сцены. Нажав на кнопку «Остановить рендер» произойдет остановка процесса синтеза сцены и пользователю будет предоставлен текущий результат.

Для перемещения по сцене пользователю предоставлена возможность перемещать камеру, а также поворачивать её на заданный угол вокруг заданной оси координат.

В качестве настроек синтеза сцены пользователь может изменять параметры «Глубина рекурсии отражений» и «Число потоков».

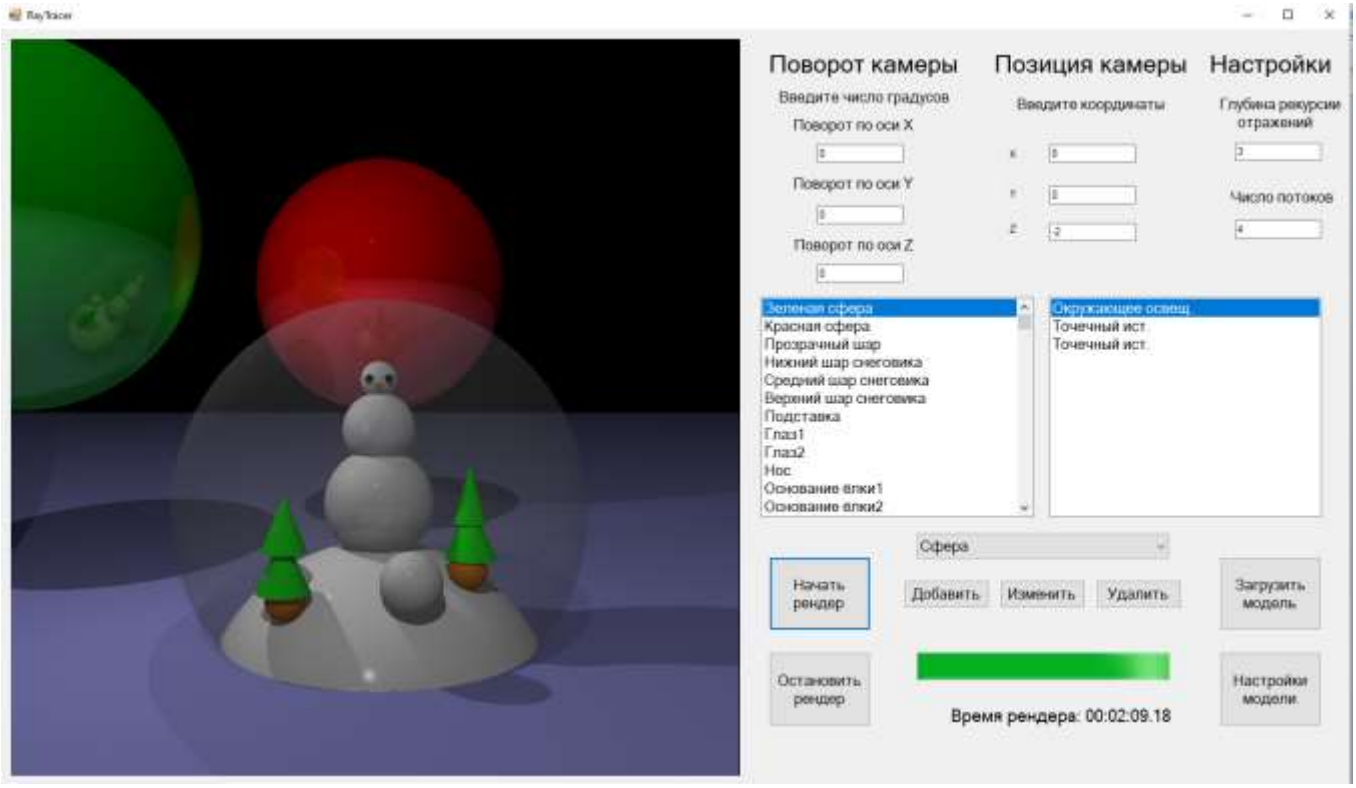


Рисунок 3.4 – Главное окно программы

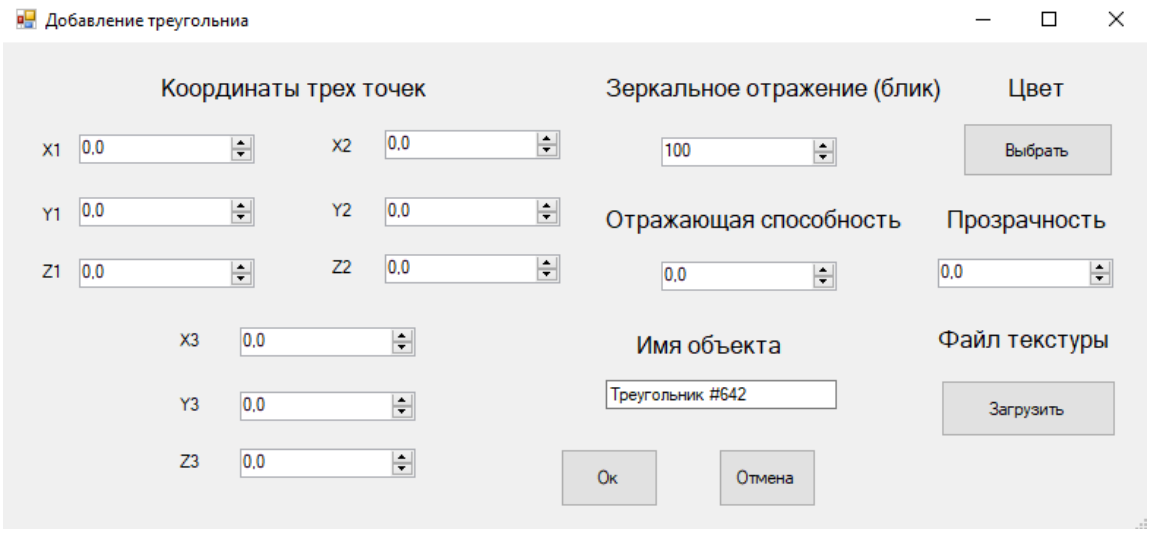


Рисунок 3.5 – Окно добавления объекта треугольника

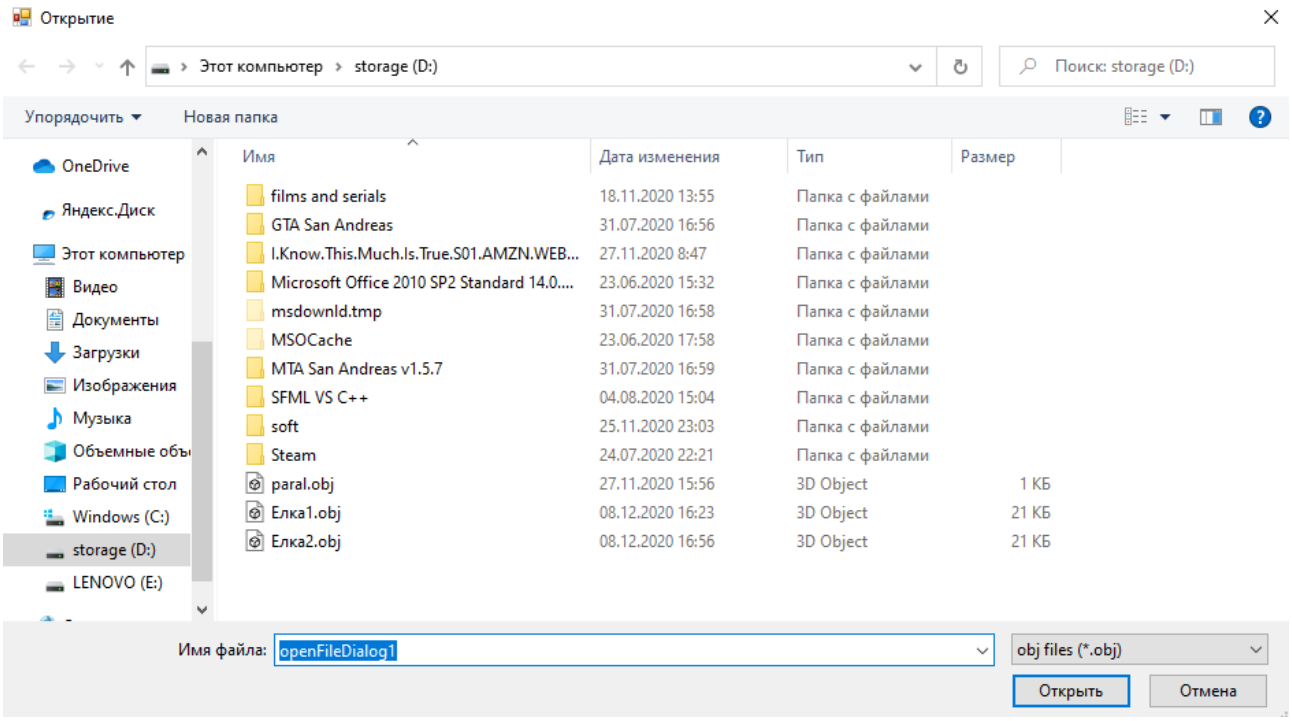


Рисунок 3.6 – Окно выбора загружаемой модели

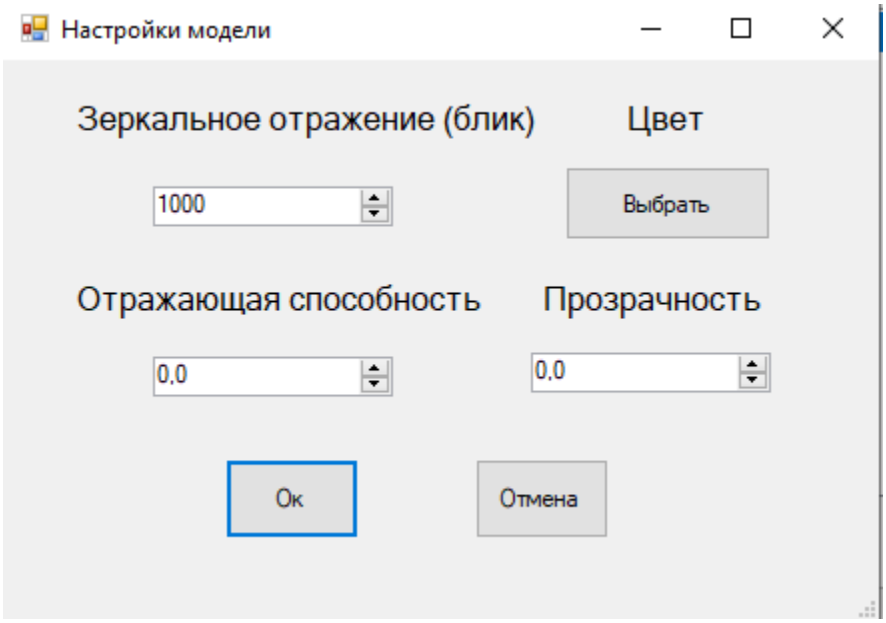


Рисунок 3.7 – Окно настройки загружаемой модели

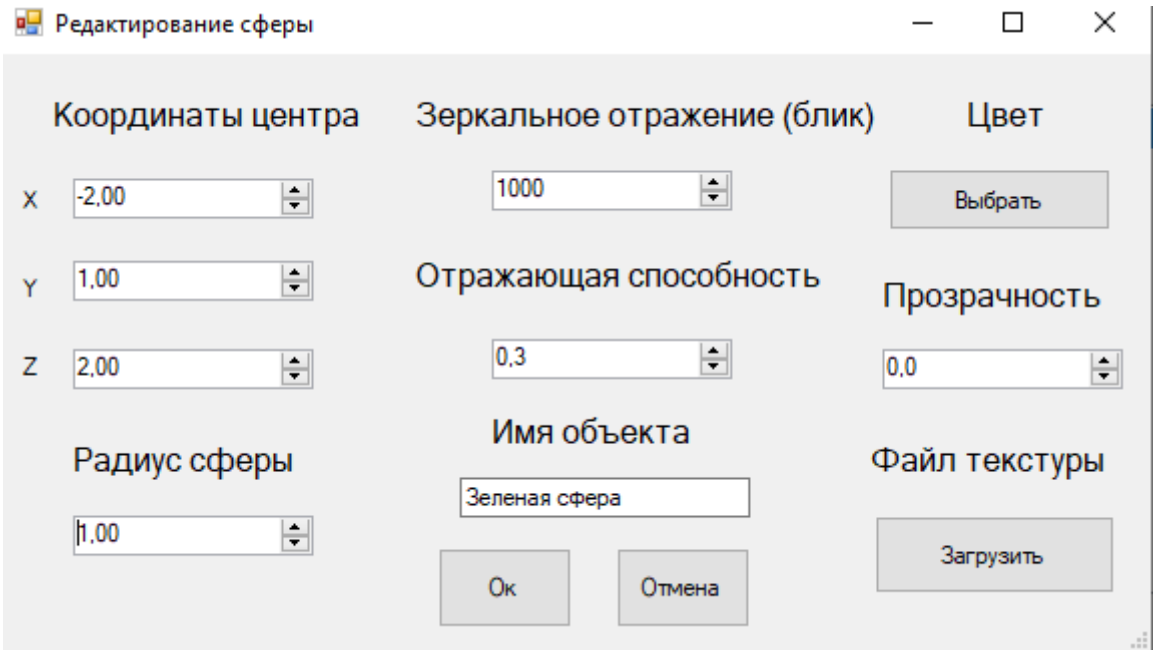


Рисунок 3.8(а) – Окно изменения объекта сфера

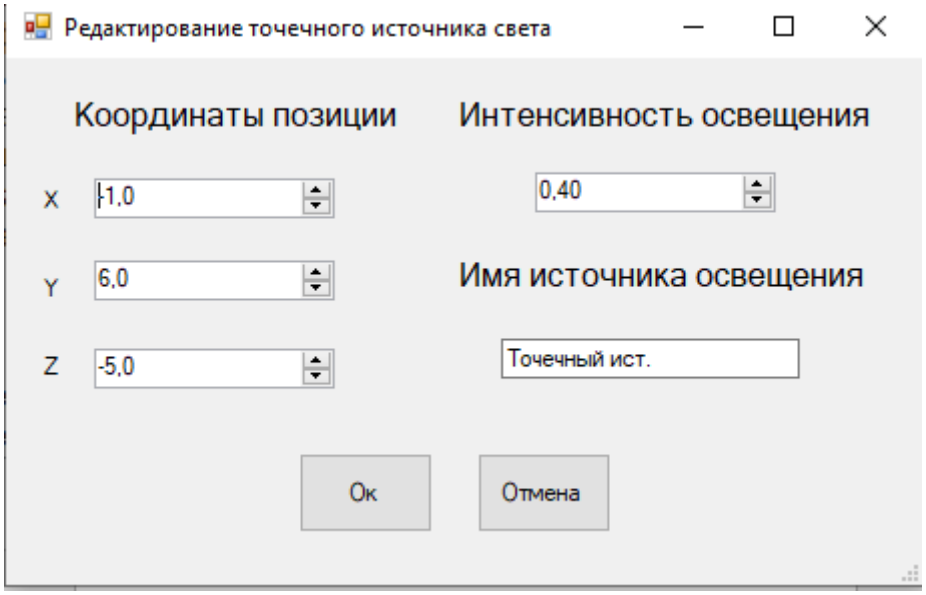


Рисунок 3.8(б) – Окно изменения точечного источника освещения

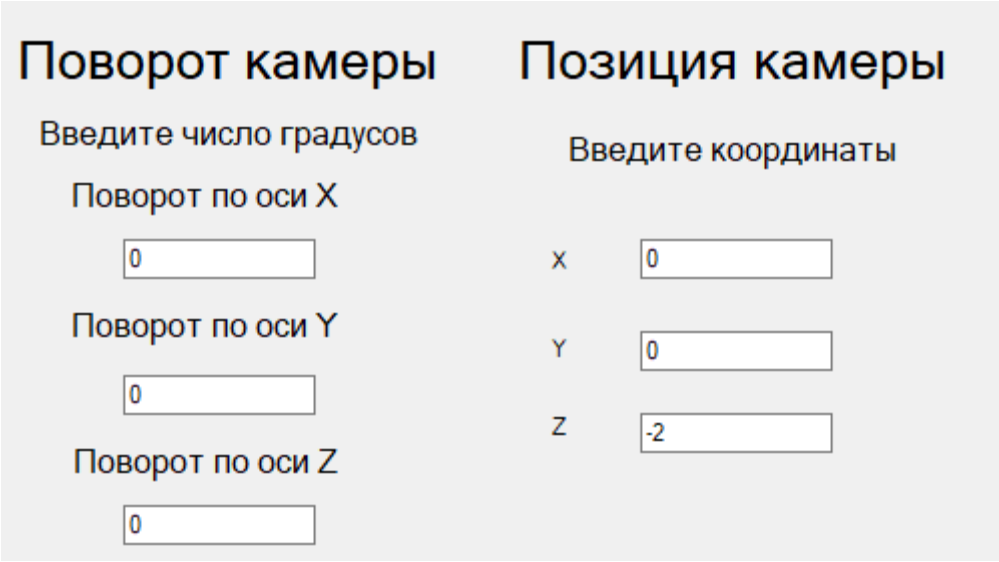


Рисунок 3.9 – Интерфейс для задания нужной позиции и ориентации камеры

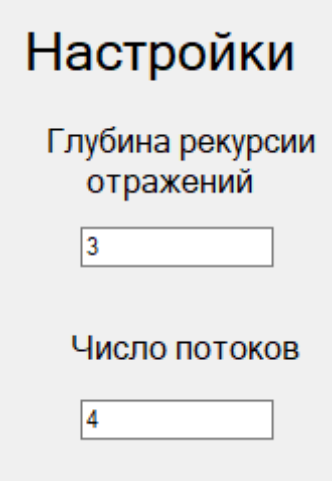


Рисунок 3.10 – Интерфейс для параметров синтеза сцены.

4. Экспериментальная часть

В данном разделе будут проведены эксперименты для проведения сравнительного анализа скорости синтеза изображения и исследование визуальных характеристик полученных изображений. Исследования проводились на ноутбуке с процессором Intel(R) Core(TM) i3-8130U CPU 2.20 GHz с 4 логическими ядрами под управлением Windows 10 с 8 Гб оперативной памяти.

4.1 Исследование скорости синтеза сцены

В рамках данного исследования был проведен эксперимент по вычислению зависимости времени визуализации от количества потоков.

В эксперименте использовалась сцена, состоящая из прозрачной сферы, двенадцати непрозрачных сфер с небольшой отражательной способностью, двух сфер с высокой отражательной способностью, двух источников света, поверхности земли и четырех конусов, состоящих из 642 полигонов-треугольников.

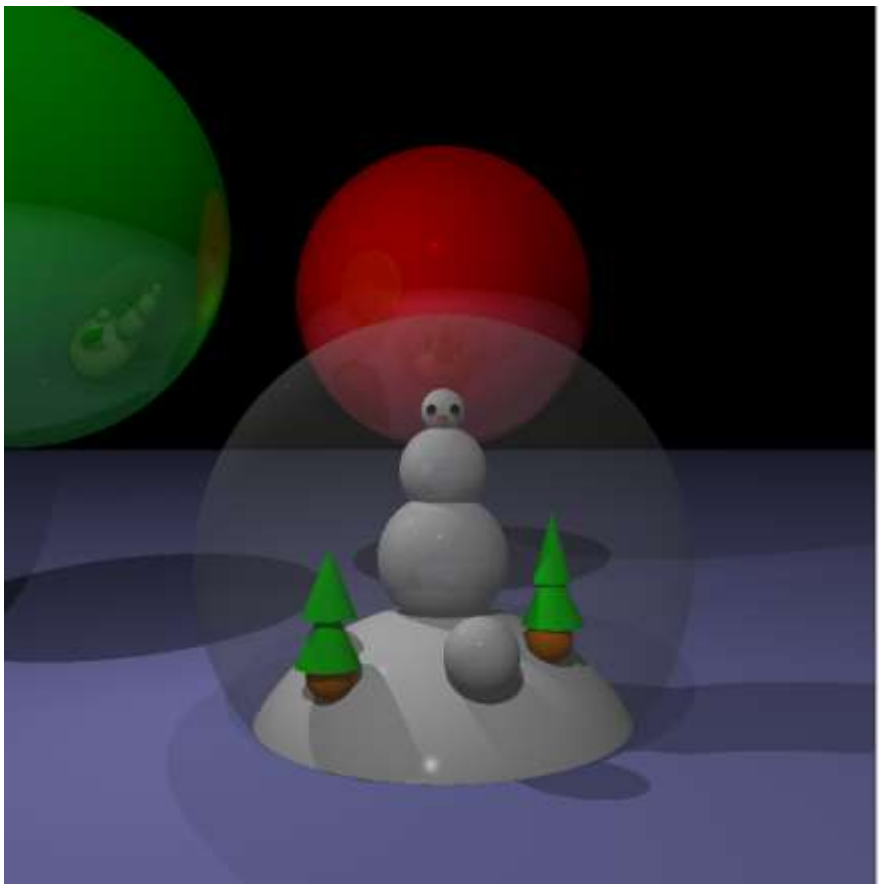


Рисунок 4.1 – Синтезируемая в эксперименте сцена

В каждом замере сцена одинаковая и синтезируется в разрешении 809 на 817 пикселей.

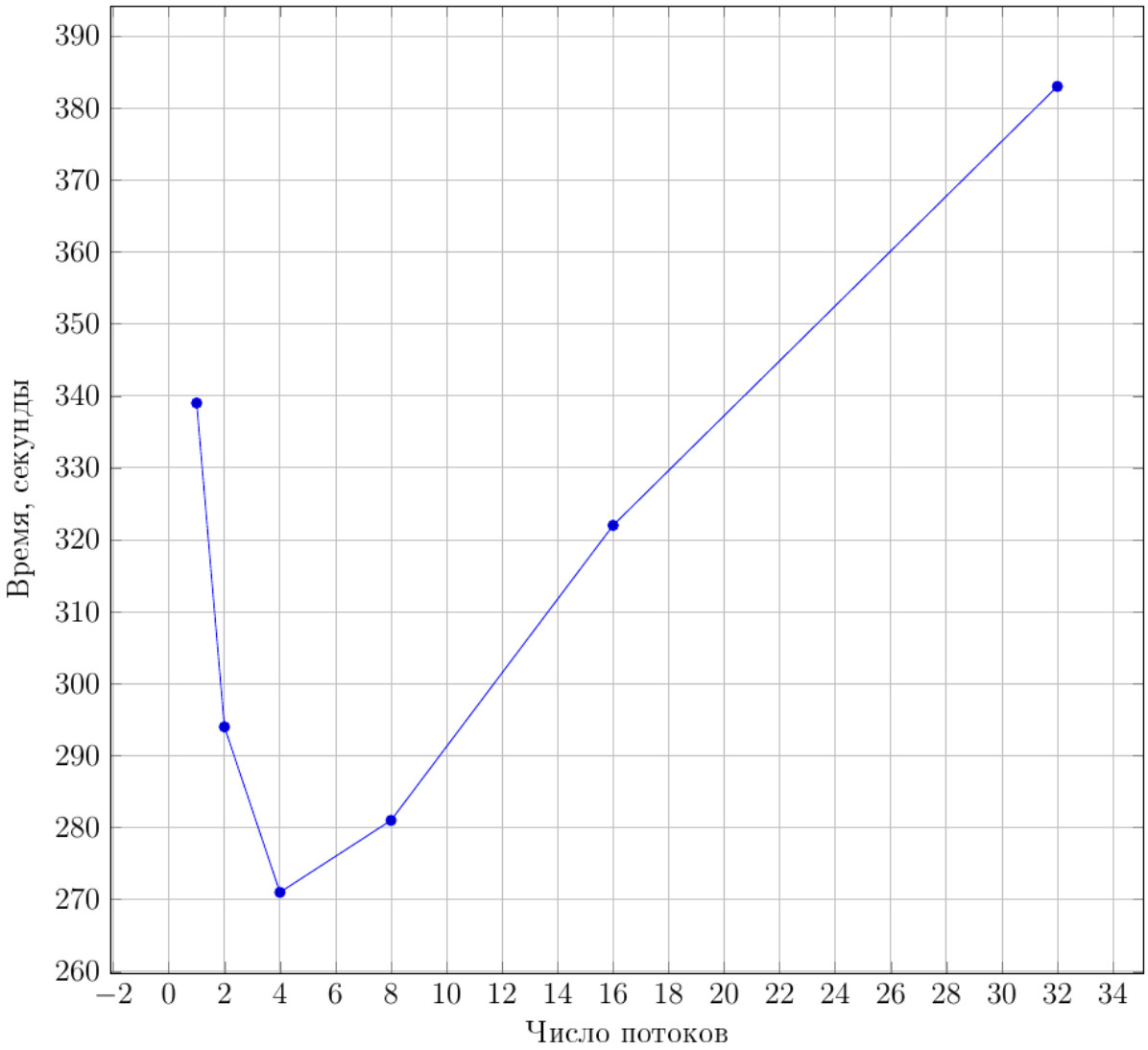


Рисунок 4.2 – График зависимости времени синтеза сцены от количества потоков

При количестве потоков большем, чем количество логических ядер процессора системе необходимо постоянно переключаться между ними, что требует времени, так как при переключении сохраняется полный контекст задачи. Распараллеливание процесса синтеза изображения уменьшило время работы алгоритма более чем в 1.25 раза.

Максимальная скорость работы достигается при количестве потоков равных количеству логических ядер процессора, что и подтверждает практика.

4.2 Исследование визуальных характеристик

В рамках данного исследования были проведены следующие эксперименты:

1) синтез сцены, расположенной за прозрачной поверхностью разной степени прозрачности;

2) синтез сцены с объектами, обладающими высоким коэффициентом отражения;

3) синтез сцены, с текстурированной сферой;

4) синтез сцены, с текстурированным треугольником;

5) синтез сцены, с использованием мягких теней.

4.2.1 Прозрачная поверхность

На каждом последующем изображении уменьшается степень прозрачности сферы. На первом изображении прозрачность наивысшая, на последнем наинизшая.

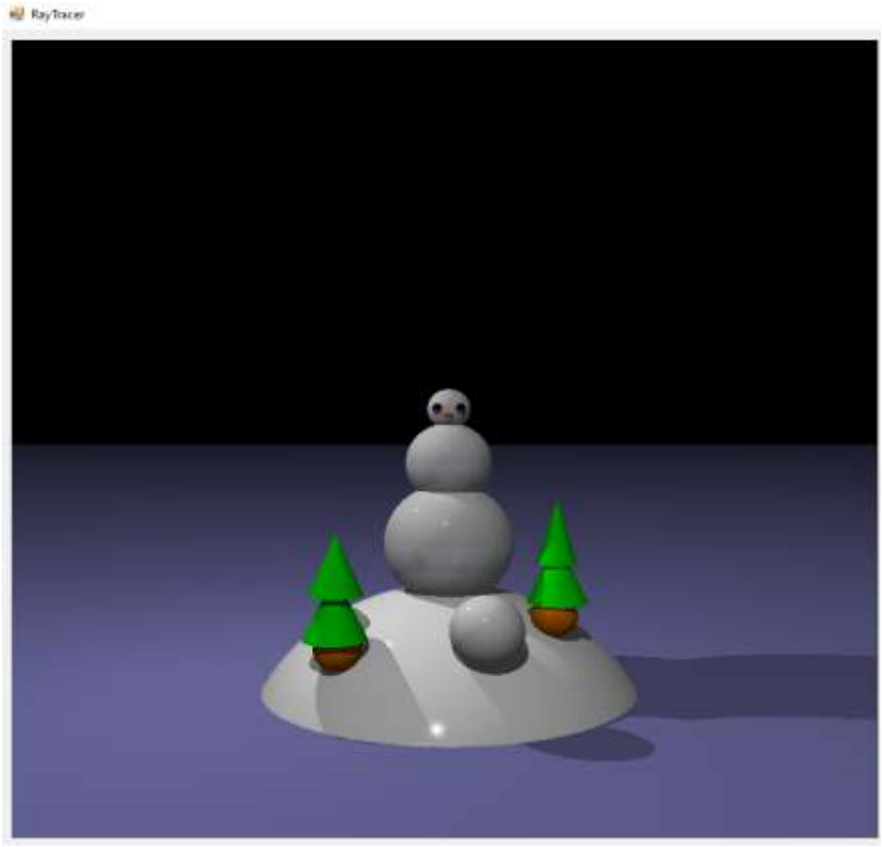


Рисунок 4.3 – Синтез сцены, расположенной за прозрачной поверхностью, настройка прозрачности равна 1

Рисунок 4.4 – Синтез сцены, расположенной за прозрачной поверхностью, настройка прозрачности равна 0.8

Рисунок 4.5 – Синтез сцены, расположенной за прозрачной поверхностью, настройка прозрачности равна 0.6

Рисунок 4.6 – Синтез сцены, расположенной за прозрачной поверхностью, настройка прозрачности равна 0.4

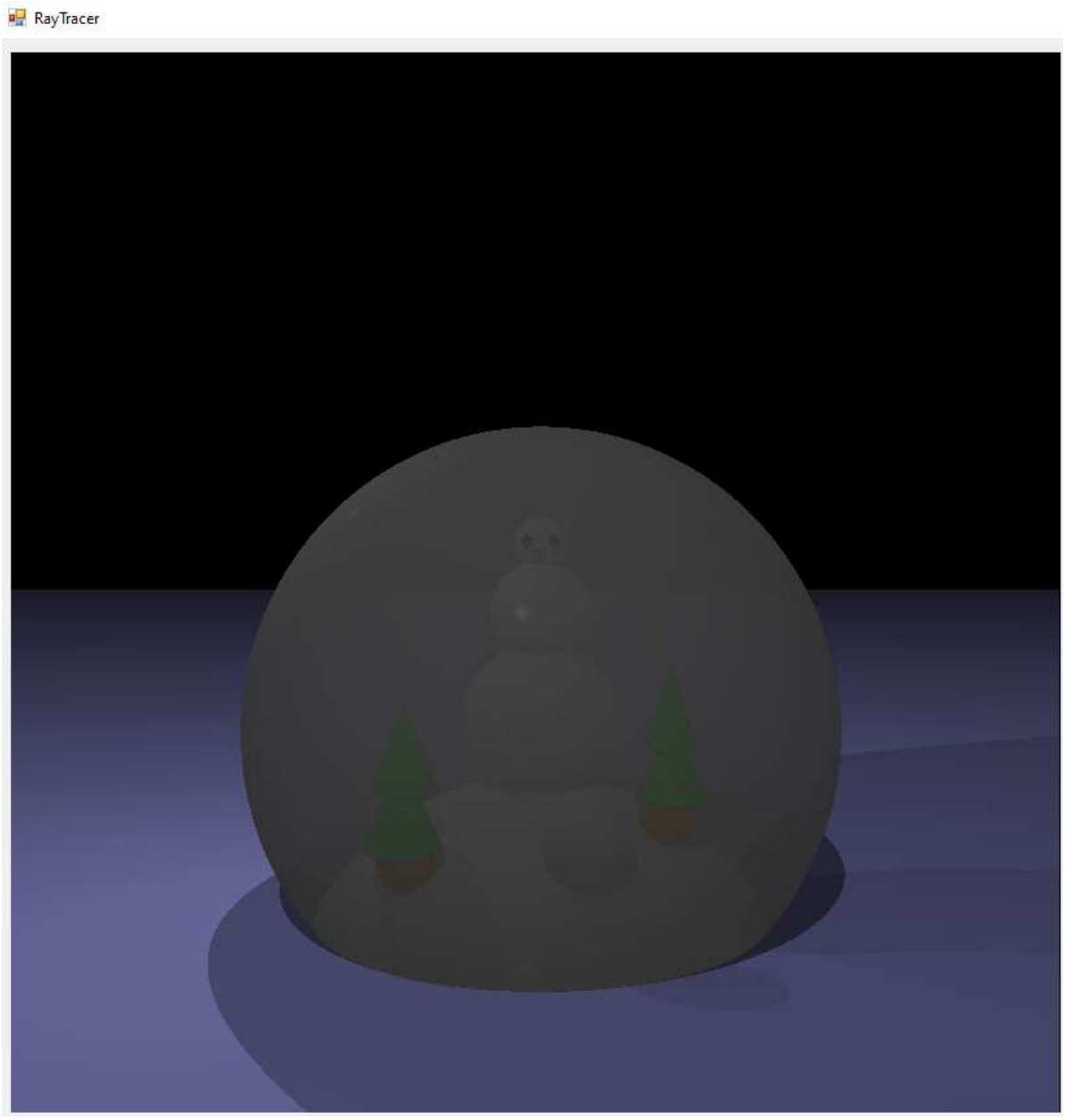


Рисунок 4.7 – Синтез сцены, расположенной за прозрачной поверхностью, настройка прозрачности равна 0.2

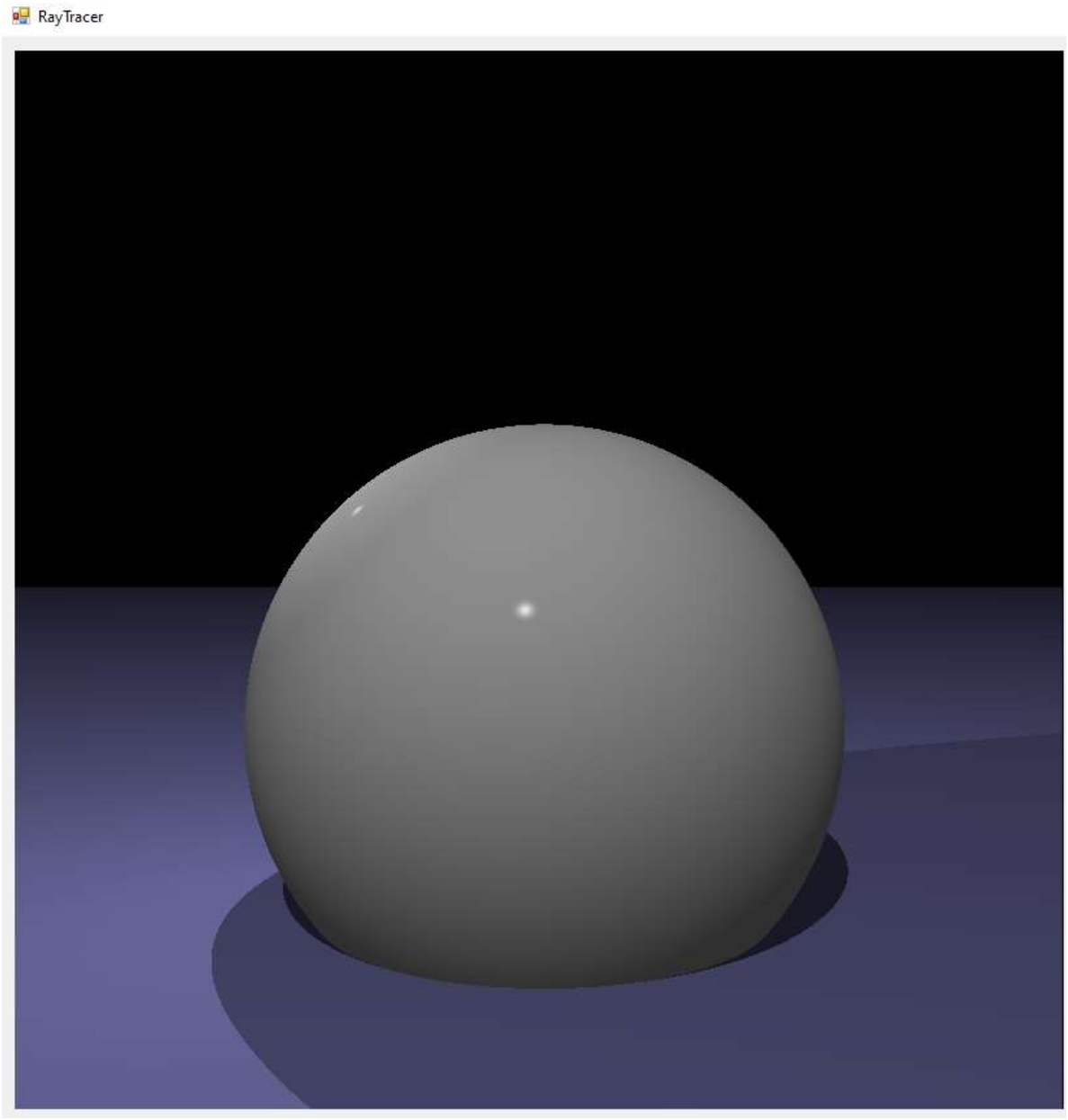


Рисунок 4.8 – Синтез сцены, расположенной за прозрачной поверхностью, настройка прозрачности равна 0

Приведенные результаты наглядно показывают корректный синтез сцены, находящейся за прозрачной поверхностью.

При уменьшении коэффициента прозрачности тень от прозрачной сферы увеличивает свою интенсивность, а тень от объектов, находящихся в ней, уменьшает свою интенсивность.

4.2.2 Две сферы с высоким коэффициентом отражения

В нижеприведенных результатах работы программного обеспечения можно детально рассмотреть отражения, наблюдаемые на зелёной и красной сферах.

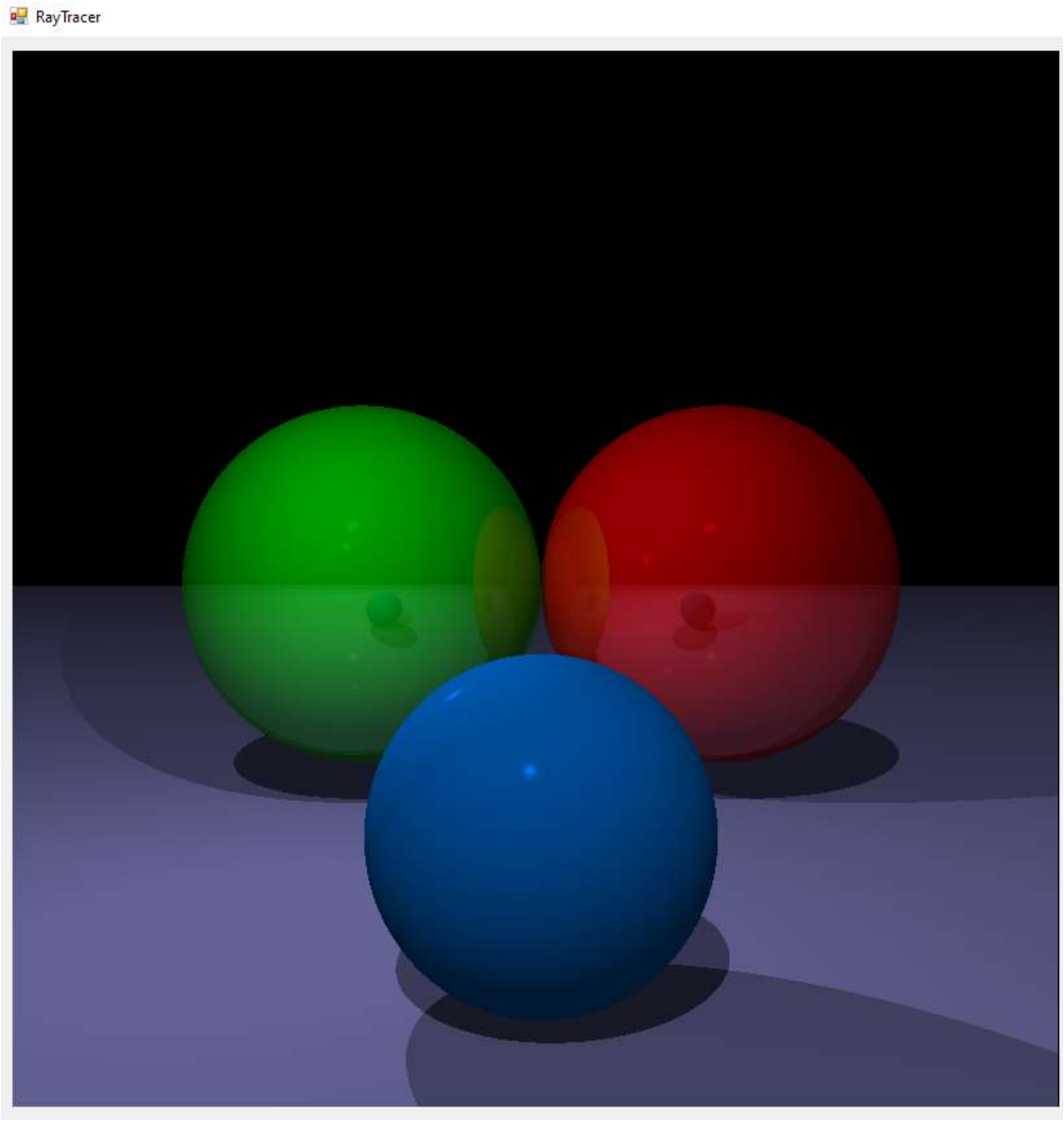
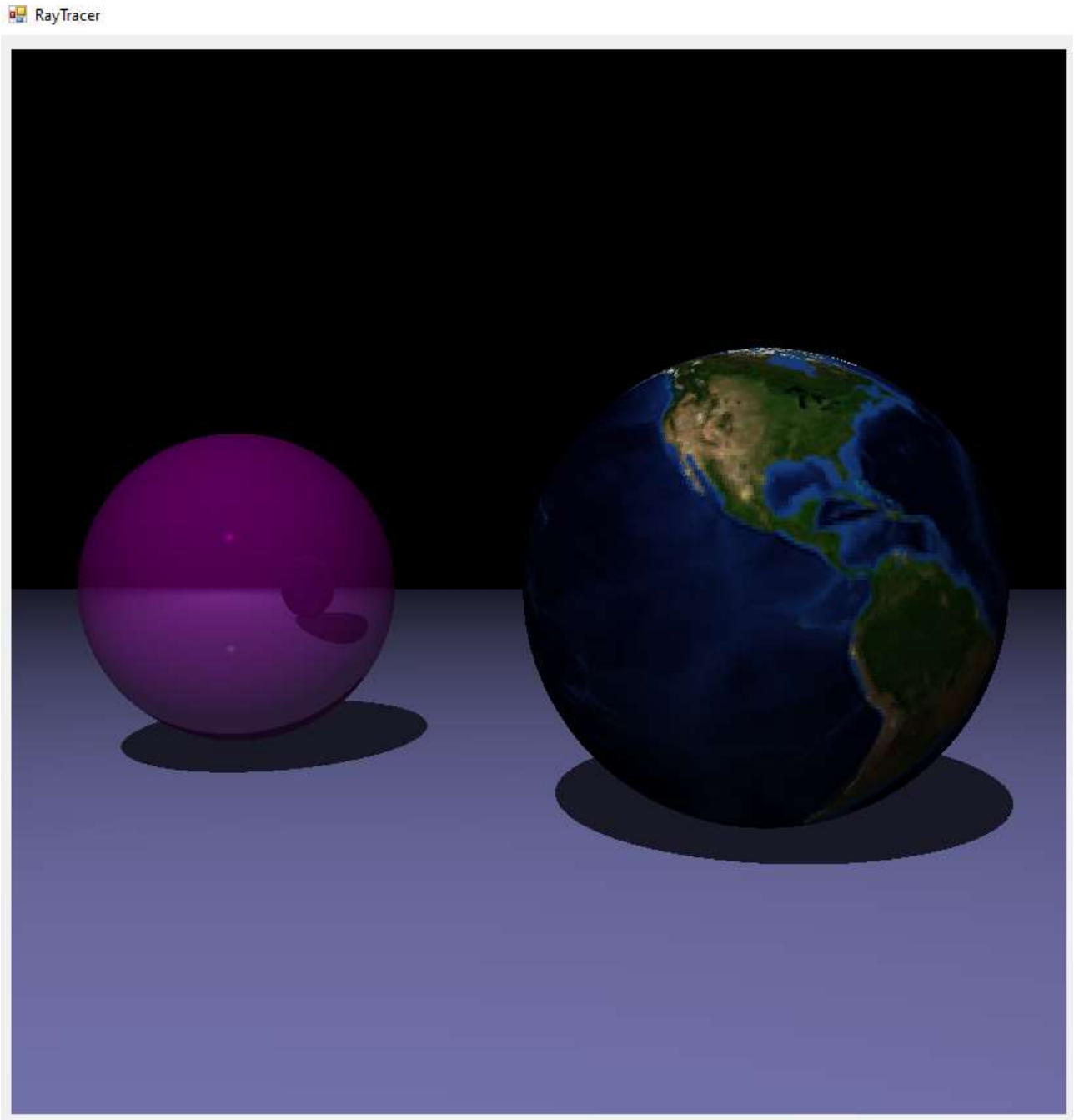
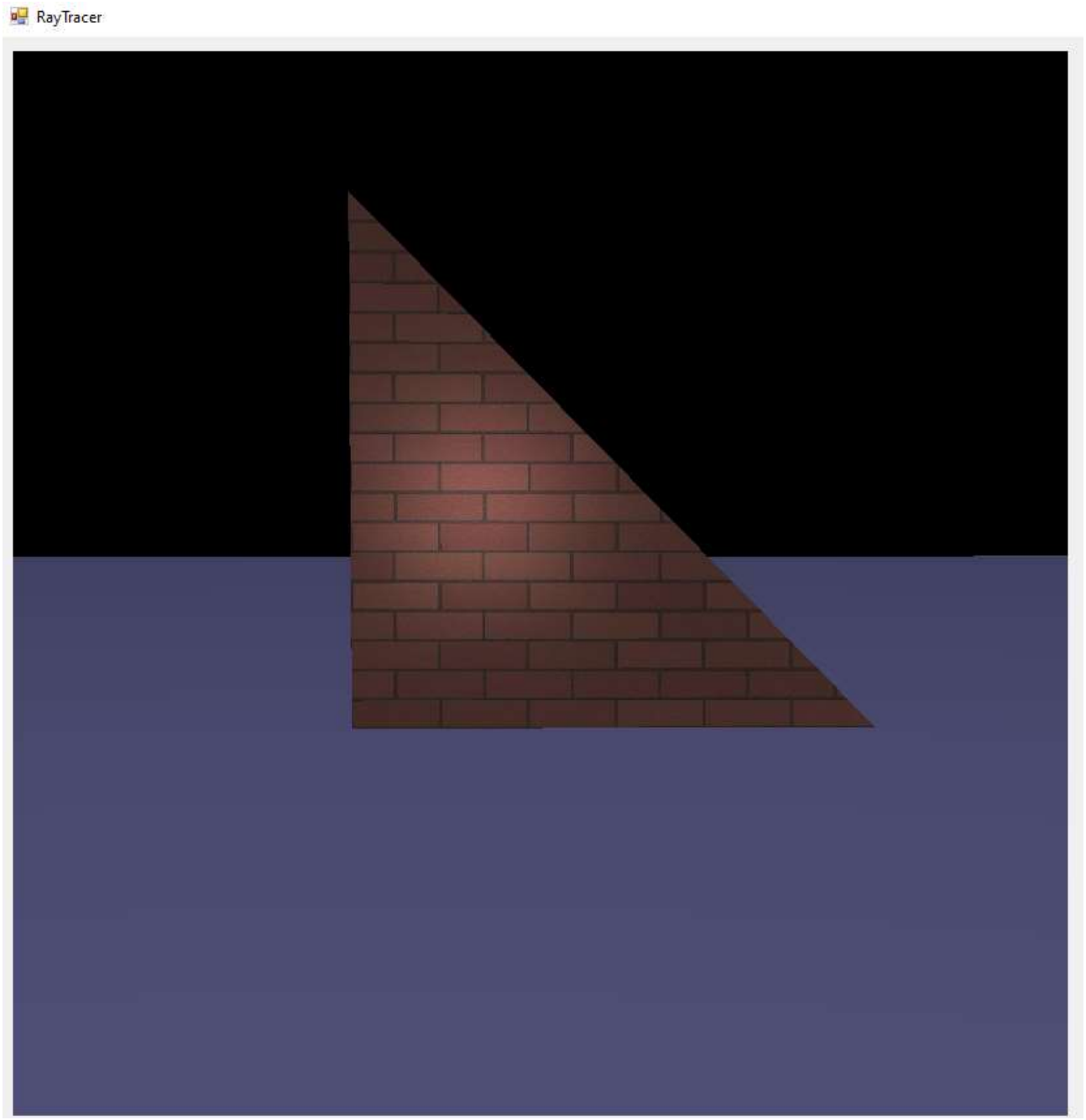


Рисунок 4.9 – Синтез сцены, с тремя сферами

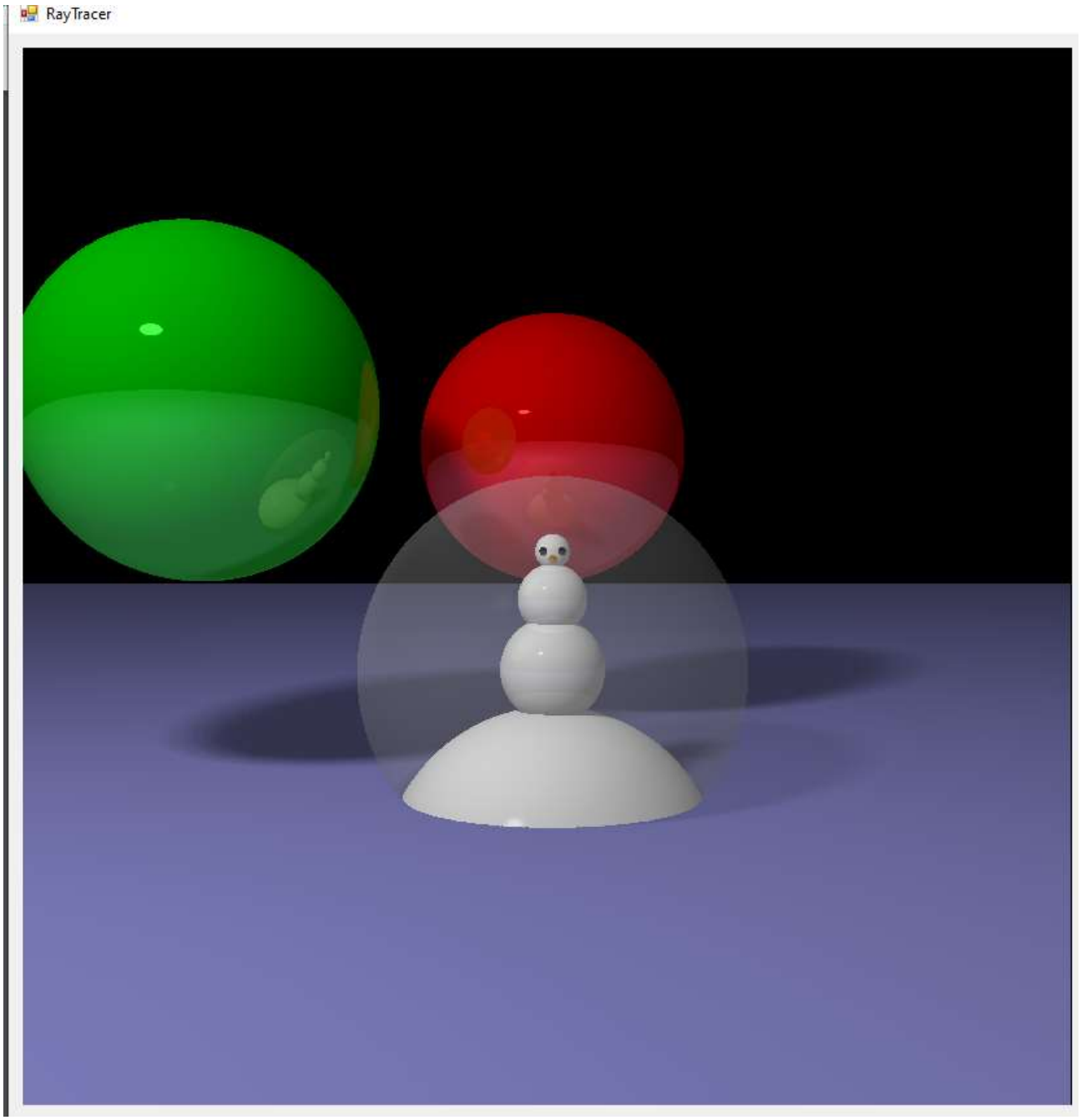
4.2.3 Текстурированная сфера



4.2.4 Текстурированный треугольник



4.2.5 Синтез сцены с мягкими тенями



Заключение

В данной работе был реализован трёхмерный редактор сцены, который позволяет переносить, изменять характеристики объектов (сферы и треугольника плоскости) и источников освещения (точечного, направленного, окружающее освещение, дисковый источник), а также изменять положения камеры в пространстве сцены.

Была предусмотрена загрузка сложных объектов, состоящих из полигонов-треугольников. Выбор характеристик таких объектов предусмотрен в графическом интерфейсе.

Для создания реалистичного изображения использовался метод обратной трассировки лучей, модель освещения Фонга.

В качестве улучшения алгоритма был реализован параллельный синтез сцены, с помощью которого время работы уменьшилось на 25% в сравнении со стандартной реализацией алгоритма обратной трассировки лучей.

Список использованных источников

1. Blender. // [Электронный ресурс]. Режим доступа: https://www.blender.org/, (дата обращения: 9.08.2020).

2. BRL-CAD feature overview. // [Электронный ресурс]. Режим доступа: http: //write.flossmanuals.net/contributors-guide-to-brl-cad/feature-overview/, (дата обращения: 10.08.2020).

3. Ray Tracing soft shadow. // [Электронный ресурс]. Режим доступа: https://medium.com/@alexander.wester/ray-tracing-soft-shadows-in-real-time-a53b836d123b, (дата обращения: 4.11.2020).

4. Проблемы трассировки лучей – из будущего в реальное время.. // [Электронный ресурс]. Режим доступа: https://nvworld.ru/articles/ray-tracing/3/, (дата обращения: 15.11.2020).

5. IDE Visual Studio 2017. // [Электронный ресурс]. Режим доступа: https:// visualstudio.microsoft.com/ru/vs/, (дата обращения: 25.11.2020).

6. Thread. // [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/ru-ru/ dotnet/api/system.threading.thread?view=netcore-3.1, (дата обращения: 1.12.2020).

7. C#. // [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/ru-ru/ dotnet/csharp/, (дата обращения: 2.12.2020).