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 КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Разработка программы для визуализации оптического преломления через линзы, заданные математическими уравнениями»

Студент ИУ7-52Б _____ Ю. Н. Городский
(Группа) (Подпись, дата) (И. О. Фамилия)

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

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Преломление света — это изменение направления распространения светового пучка при его переходе через границу двух сред с разными показателями преломления. Это явление лежит в основе работы множества оптических приборов, таких как линзы, которые находят широкое применение в фотографии, микроскопии, телескопах и многих других сферах. Классическим примером демонстрации преломления является прохождение света через выпуклую или вогнутую линзу.

Современные методы моделирования и визуализации позволяют исследовать и воспроизводить это явление с высокой степенью точности. Программное обеспечение для визуализации преломления света через линзы позволяет исследовать и анализировать оптические системы, а так же создавать реалистичные эффекты в кинематографии.

Целью данной курсовой является разработка программы для визуализации процесса преломления света через аналитически заданные линзы. Программа должна предоставить пользователю возможность выбора источников освещения и объектов, а также параметров сцены для анализа оптических эффектов.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучение явления преломления света и законов оптики с физической точки зрения;
- анализ существующих алгоритмов визуализации оптических эффектов;
- выбор алгоритма для решения поставленной задачи;
- проектирование архитектуры программы и графического интерфейса;
- реализация структур данных и алгоритмов;
- описание структуры программного обеспечения;
- реализация программы;
- исследование производительности программы.

1 Аналитическая часть

В данном разделе рассмотрены физические основы преломления, существующие алгоритмы построения реалистических изображений и способы хранение трехмерных объектов. Обосновывается выбор алгоритмов и форматов 3д объектов для реализации поставленной задачи.

1.1 Физическая модель оптических линз

Оптическая линза (далее просто линза) - деталь из прозрачного однородного материала, имеющая две преломляющие поверхности, например, обе сферические или же одну плоскую, другую - сферическую. Для объяснения физики преломления света при прохождении через оптические линзы, необходимо ввести следующие понятия [1]:

- свет - электромагнитное излучение, распространяемое в виде волны;
- поляризованный свет — это свет, в котором направление колебаний вектора напряженности электрического поля упорядочено;
- неполяризованный свет - естественный свет, в котором вектор напряженности электрического поля может с равной вероятностью иметь любое направление колебаний в плоскости, перпендикулярной направлению распространения света;
- световой луч (луч) – линия, вдоль которой распространяется световая волна;
- фаза луча – состояние световой волны в момент времени (функция координат и времени);
- длина волны – расстояние между двумя ближайшими друг к другу точками в пространстве, в которых колебания происходят в одинаковой фазе;
- абсолютный коэффициент преломления – безразмерная физическая величина, характеризующая различие фазовых скоростей света в двух средах;

– относительный показатель преломления – безразмерная величина, показывающая отношение абсолютных показателей преломления двух сред.

– интенсивность луча – энергия, переносимая лучом в заданном направлении (от нее зависит яркость излучения).

На рисунке 1.1 представлено поведение луча на границе двух сред с разными коэффициентами преломления. Луч, падающий на границу двух веществ, лежит в плоскости чертежа и задан вектором k, а нормаль к поверхности раздела задана вектором n. Линия пересечения плоскости падения луча и границы раздела двух сред задана осью x, ось y направлена перпендикулярно к плоскости раздела сред. Траектория луча в таком случае представлена на рисунке 1.1.

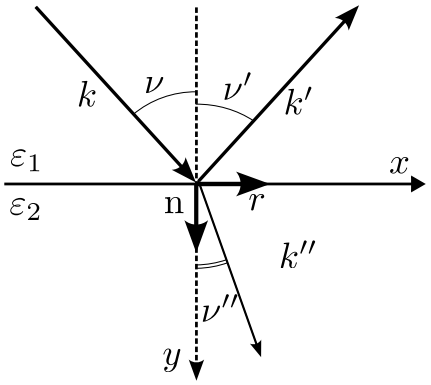


Рисунок 1.1 – Траектория луча на границе двух сред

Преломленный луч лежит в одной плоскости с падающим лучом и нормалью, востановленной в точке падения. Отношение синуса угла падения к синусу угла преломления есть постоянная величина для заданных веществ.[2] При этом углы преломления и отражения задаются уравнениями 1.1 и 1.2 соответственно.

$$n_1 \sin \nu = n_2 \sin \nu'' \tag{1.1}$$

$$\nu = \nu' \tag{1.2}$$

Существует два основных коэффициента отражения: для света с параллельной поляризацией (R_p) и для света с перпендикулярной поляризацией

(R_s):

$$R_s = \left(\frac{n_1 \cos(\theta_1) - n_2 \cos(\theta_2)}{n_1 \cos(\theta_1) + n_2 \cos(\theta_2)} \right)^2 \tag{1.3}$$

$$R_p = \left(\frac{n_2 \cos(\theta_1) - n_1 \cos(\theta_2)}{n_2 \cos(\theta_1) + n_1 \cos(\theta_2)} \right)^2, \tag{1.4}$$

где:

– n_1 и $_2$ – показатели преломления первой и второй среды;

– θ_1 – угол падения луча на поверхность раздела сред;

– θ_2 – угол преломления, который вычисляется по закону 1.1.

Для неполяризованного света коэффициент отражения рас считывается по формуле:

$$R = \frac{R_s + R_p}{2} \tag{1.5}$$

Коэффициент преломления будет рассчитываться по формуле:

$$T = 1 - R \tag{1.6}$$

При прохождение луча через линзу может наблюдаться явление дисперсии - расхождение световых лучей разного цвета.

1.2 Формализация задачи

Формализованная задача представлена на рисунке

Рисунок 1.2 – Формализованная задача с учетом выбранных алгоритмов

1.3 Формализация объектов сцены

На сцене могут присутствовать следующие типы объектов:

– Сферы с параметрами радиуса, прозрачности, коэффициентом преломления, цветом, уветом излучения и интенсивностью излучения (при излучении света, лучи направлятся в направлении нормали к поверхности сферы);

– Линзы с параметрами радиуса, радиуса кривизны, прозрачности, цвета и коэффициента преломления;

– Наблюдатель (задается положением, напрвлением зрения и углом обзора);

– Полигоны (задаются тремя точками и цветом);

1.4 Модель освещения

Модель освещения определяет интенсивность освещения I в каждой точке. Наиболее распространенными моделями освеения являются глобальная и лоакльная [3].

В локальной модели освещения каждый объект рассматривается по отдельности, без учета взаимодействия с другими объектами. Глобальная модель освещения учитывает взаимное расположение объектов. Поскольку для корректного отображения преломления света через линзы необходимо учитывать их взаимное расположение, будет использоваться глобальная модель освещения.

В глобальной модели освещения для каждой точки P пространства сцены интенсивность освещения I будет рассчитываться из 4 состовляющих:

– фоновое освещение, существующее в каждой точке пространства сцены, не зависящее от источников света и координаты точки.

– рассеянный свет, распространяющийся равномерно во все стороны, при попадании луча на поверхность объекта, зависящий от ориентации поверхности (нормали N к поверхности в точке), направления на источник света L и интенсивности источника;

– зеркальная составляющая, зависящая от зеркальности поверхности и от того, насколько близки направления на наблюдателя и отраженного луча;

– преломленная составляющая, зависящая от интенсивности преломленного луча.

Поскольку в задача работы состоит в отображении преломления через лензы, предлагается упростить модель освещения, не учитывая рассеянный свет. В таком случае итоговая составляющая будет рассчитываться по формуле:

$$I = k_\alpha I_\alpha + \sum_j I_j(\overline{N} \cdot \overline{L_j}) + k_r I_r + k_s I_s \tag{1.7}$$

где:

k_α – коэффициент фонового освещения;

k_r – коэффициент пропускания;

k_s – коэффициент отражения;

I_j – интенсивность j-го источника света;

I_r – интенсивность преломленного луча;

I_s – интенсивность отраженного луча;

$\overline{N}$ – нормаль к поверхности в точке;

$\overline{L_j}$ – вектор, направленный к j-му источнику.

1.4.1 Метод фотонных карт

Алгоритм создания изображения состоит из трех шагов: на первом шаге испускаются прямые лучи из источников света, распространяются по сцене и формируют распределение фотонов на диффузных поверхностях сцены; на втором шаге на основе полученного распределения фотонов формируются фотонные карты для объектов сцены; на последнем шаге из камеры испускаются обратные лучи и ищется их пересечение с фотонными картами[4]. В точках пересечения с фотонными картами происходит перерасчет светового потока,

переносимого фотоном, в яркость вторичного излучения, наблюдаемую в данном направлении:

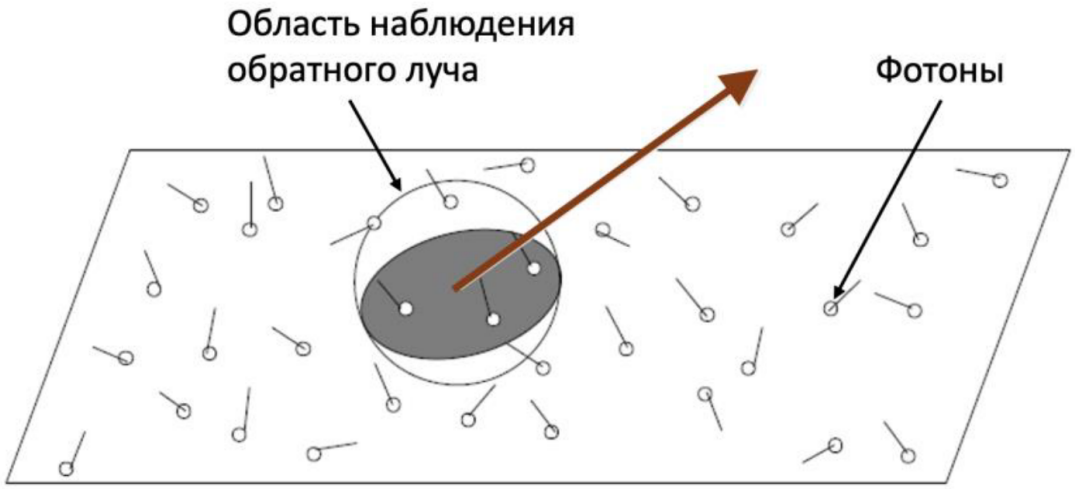


Рисунок 1.3 – Сбор распределения яркости с фотонов при трассировке обратного луча

1.4.2 Простроение освещения с помощью трассировки лучей

В алгоритме используется модификация алгоритма, описанного в пункте 1.5.4. Для расчета непрямого освещения для каждой точки дифуззной поверхности при пересечении с лучом трассировки испускается заданное число случайных лучей. После чего они трассируются и на основе их увета формируется непрямое освещение. Такой метод очень трудозатратный, т.к. для количество лучей трассировки резко возрастает.

1.5 Алгоритмы удаления невидимых поверхностей

Удаление невидимых линий и поверхностей заключается в определении тех частей объектов, которые видимы или невидимы для наблюдателя из определенной точки пространства. Эта задача может решаться в двух пространствах:

- Объектное пространство – используется физическая система координат, в которой описаны объекты сцены, обеспечивает высокую точность изображения;

- Пространство изображения – используется система координат экрана, на котором визуализируется полученное изображение, точность вычислений ограничена разрешающей способностью экрана.

Самыми распространенными алгормами, решающими задачу являются: алгоритм Робертса, z-буффера, трассировки лучей.

1.5.1 Алгоритм Робертса

Алгоритм Робертса — это один из первых методов для удаления невидимых линий, работающий в объектном пространстве. Он удаляет те ребра и грани, которые скрыты самим объектом[5]. Затем оставшиеся видимые ребра сравниваются с другими объектами для определения, какие их части могут быть скрыты.

Трудоемкость алгоритма возрастает пропорционально квадрату количества объектов. Он подходит только для работы с выпуклыми телами, а невыпуклые объекты необходимо разбивать на выпуклые части. Поздние версии алгоритма используют предварительную сортировку для повышения эффективности.

1.5.2 Алгоритм Z–буфера

Алгоритм z-буфера — это один из простейших методов удаления невидимых поверхностей в 3D-графике, предложенный Кэтмулом[6]. Он работает в пространстве изображения, используя отдельный буфер для хранения глубины каждого видимого пикселя. При отрисовке нового пикселя его глубина сравнивается с той, что уже записана в буфер. Если новый пиксель ближе, он заменяет предыдущий в буфере кадра, и z-буфер обновляется его глубиной. Алгоритм прост, не требует сортировки по глубине, но требует большого объема памяти и может быть менее эффективным при прозрачности или антиалиасинге.

Пример работы алгоритма, использующего z–буфер, представлен на рисунке 1.4.

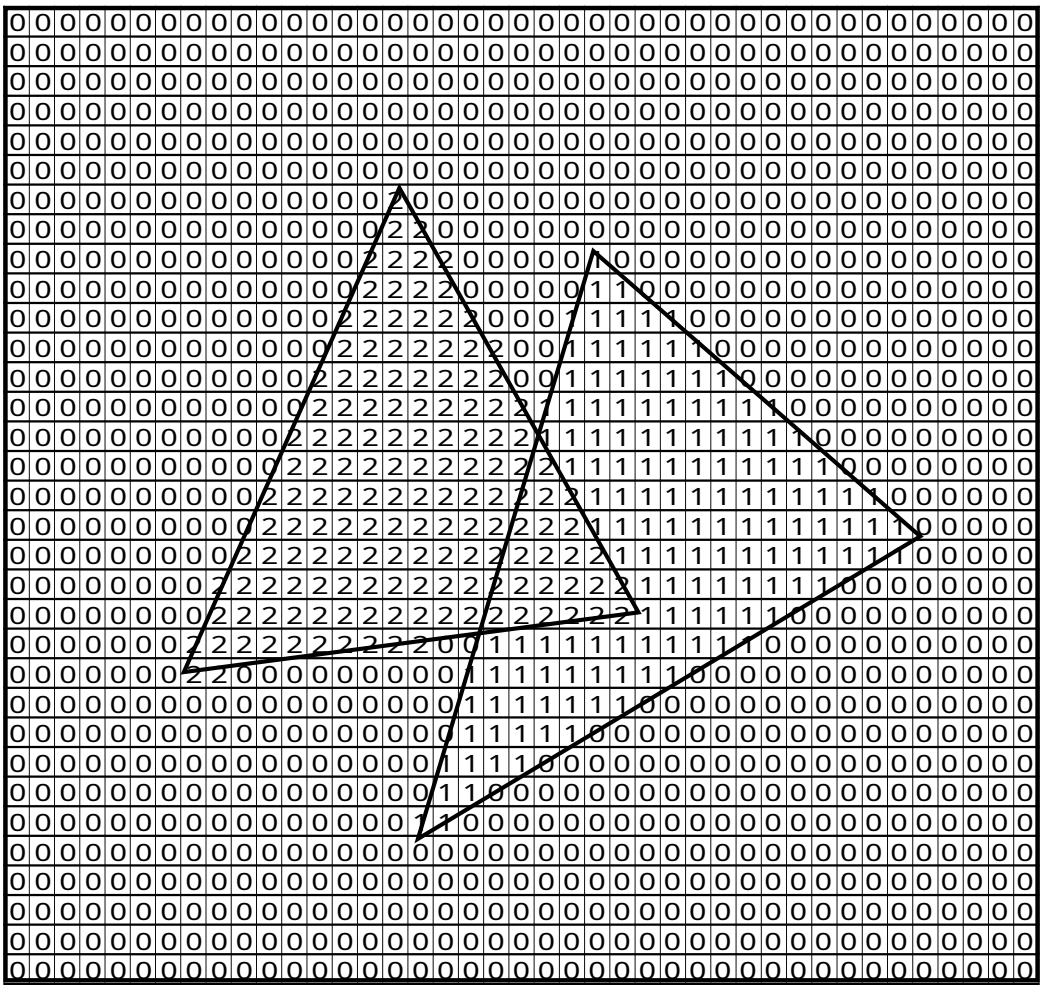


Рисунок 1.4 – Пример работы алгоритма Z–буфера

1.5.3 Алгоритм Варнока

Алгоритм Варнока основан на принципе когерентности изображения, при котором большие однородные области требуют меньше усилий для обработки[7]. Он работает в пространстве изображения и разбивает его на подокна, чтобы определить, простое ли их содержимое. Если содержимое сложное, окно делится дальше, пока не достигнет уровня пиксела. Пример работы алгоритма представлен на рисунке 1.5.

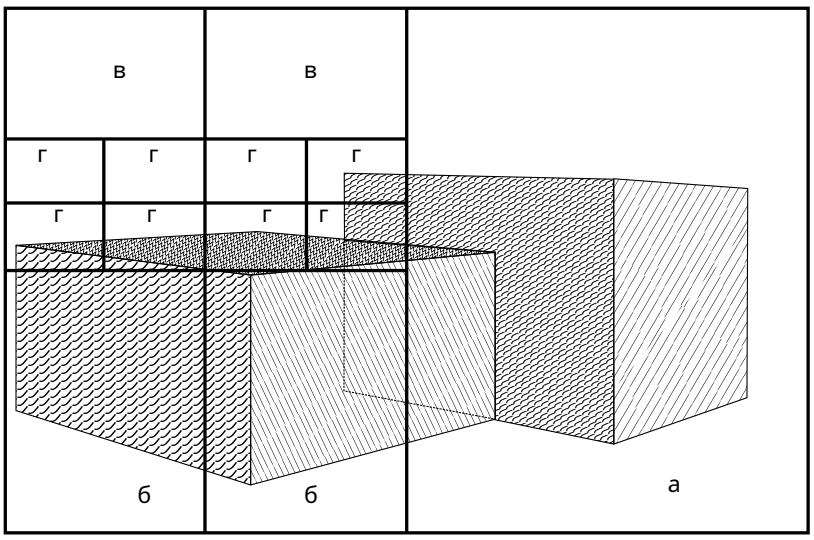


Рисунок 1.5 – Пример работы алгоритма Варнока

Основная идея алгоритма — сосредотачивать вычислительные ресурсы на сложных частях сцены, игнорируя простые области. Метод эффективен для удаления невидимых линий и поверхностей.

1.5.4 Алгоритм обратной трассировки лучей

Алгоритм обратной трассировки лучей заключается в том, что для определения видимых поверхностей сцены лучи отправляются от наблюдателя к объекту[8]. Трассировка каждого луча позволяет выяснить, какие объекты сцены пересекаются с данным лучом, учесть преломления, отражения и прохождение сквозь объекты. Пересечения объектов с лучом упорядочиваются по глубине, и ближайшее пересечение указывает на видимую поверхность для каждого пикселя. Пример трассировки лучей представлен на рисунке 1.6.

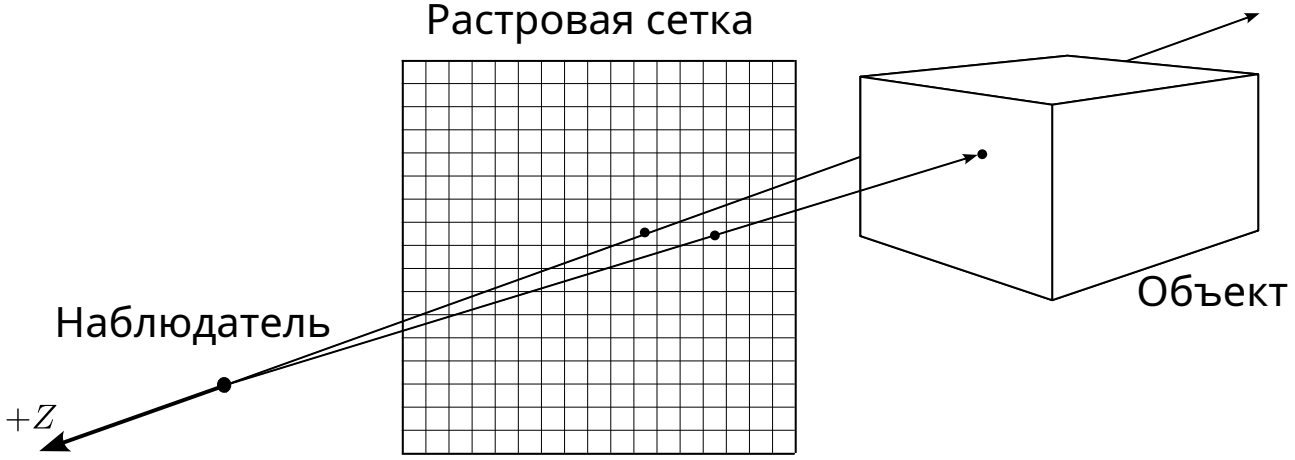


Рисунок 1.6 – Обратная трассировка лучей

Основной сложностью алгоритма является определение точек пересечения луча с объектами сцены. Для оптимизации вычислений вводятся габаритные тесты с объемными оболочками (сферами или параллелепипедами) для исключения объектов, с которыми луч не пересекается.

Алгоритм эффективен, если пересечения вычисляются быстро и точно, что существенно влияет на общую производительность.

1.6 Способы хранения трехмерных объектов

Классификацию форматов трехмерных бъектов можно провести по 2 параметрам – это способ описания их геометрии и способ описания их внешнего вида[9].

1.6.1 Геометрические параметры 3д объектов

Геометрия моделей часто задаются набором трехмерных точек. Поверхность в таком случае сохраняется ввиде ряда полигонов, которые создаются путем индексации этих вершин. В некоторых форматах используются ребра из 2 вершин. При использовании полигональных моделей для отрисовки круглых объектов можно использовать интерполирование нормалей к каждой грани. При этом существует много вариаций методов интерполирования, наиболее простым из которых является усреднение нормалей, сходящихся в одной вершине, и назначение результата этой вершине. Такой способ хранения 3д объектов достаточен для визуализации 3д контента, где точное определение геометрических свойств не требуется.

Для объектов, форму которых можно задать геометрическими примитивами, удобно использовать аналитический способ хранения.

Другим способом создания 3д моделей является конструктивная твердотельная геометрия. Этот метод использует набор логических операций над геометрическими примитивами. Преимуществом такого метода является точное описание формы объекта, если его так или иначе можно представить набором используемых примитивов, а так же возможность изменить модель на любом этапе ее создания. Данный метод широко используется в САПР.

1.7 Вывод

После оценки вышеизложенных алгоритмов сделан вывод, что для этой работы наиболее подходящим алгоритмом будет алгоритм обратной трассировки лучей, т.к. он позволяет корректно отобразить симулируемые физические явления, при этом обеспечивая достаточную производительность.

Для глобального освещения будет использоваться метод фотонных карт, поскольку он предоставляет возможность моделировать сложные эффекты освещения, такие как мягкие тени, многоступенчатое отражение света и дисперсия, с приемлемым балансом между качеством и производительностью.

Для хранения линз и сфер будет использоваться аналитический метод(линзы задаются как пересечение двух сфер).

2 Конструкторский раздел

В разделе рассмотрены требования к программному обеспечению, схемы алгоритмов, выбранных для решения поставленной задачи. Описаны пользовательские структуры данных, описана структура реализуемого программного обеспчеения.

2.1 Функциональная модель программного обеспечения

Алгоритм получения изображения представлен в виде диаграммы в нотации IDEF0, отражающей также общую структуру программы(рисунок 2.1 - 2.2).

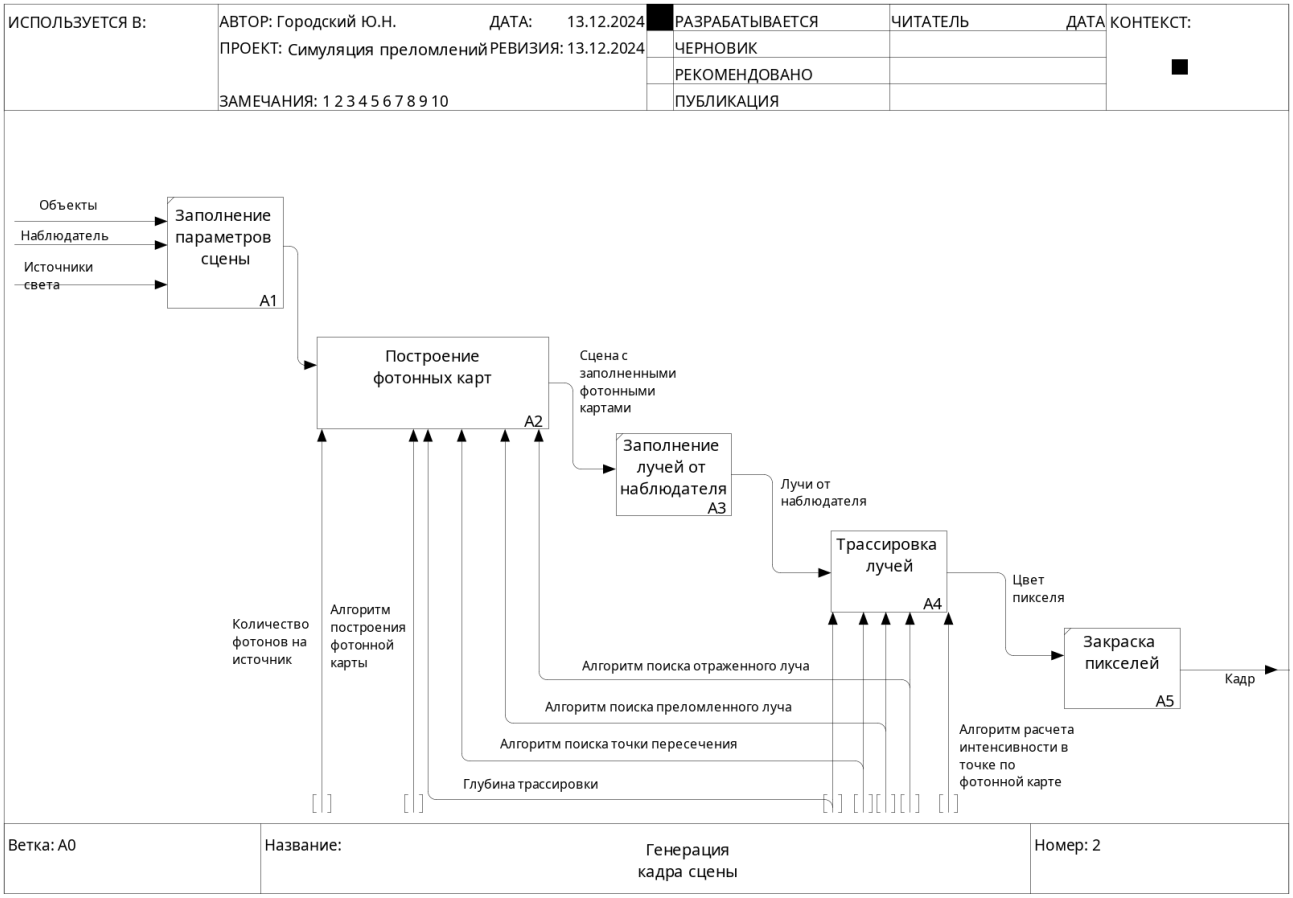


Рисунок 2.1 – Функциональная модель программного обеспечения

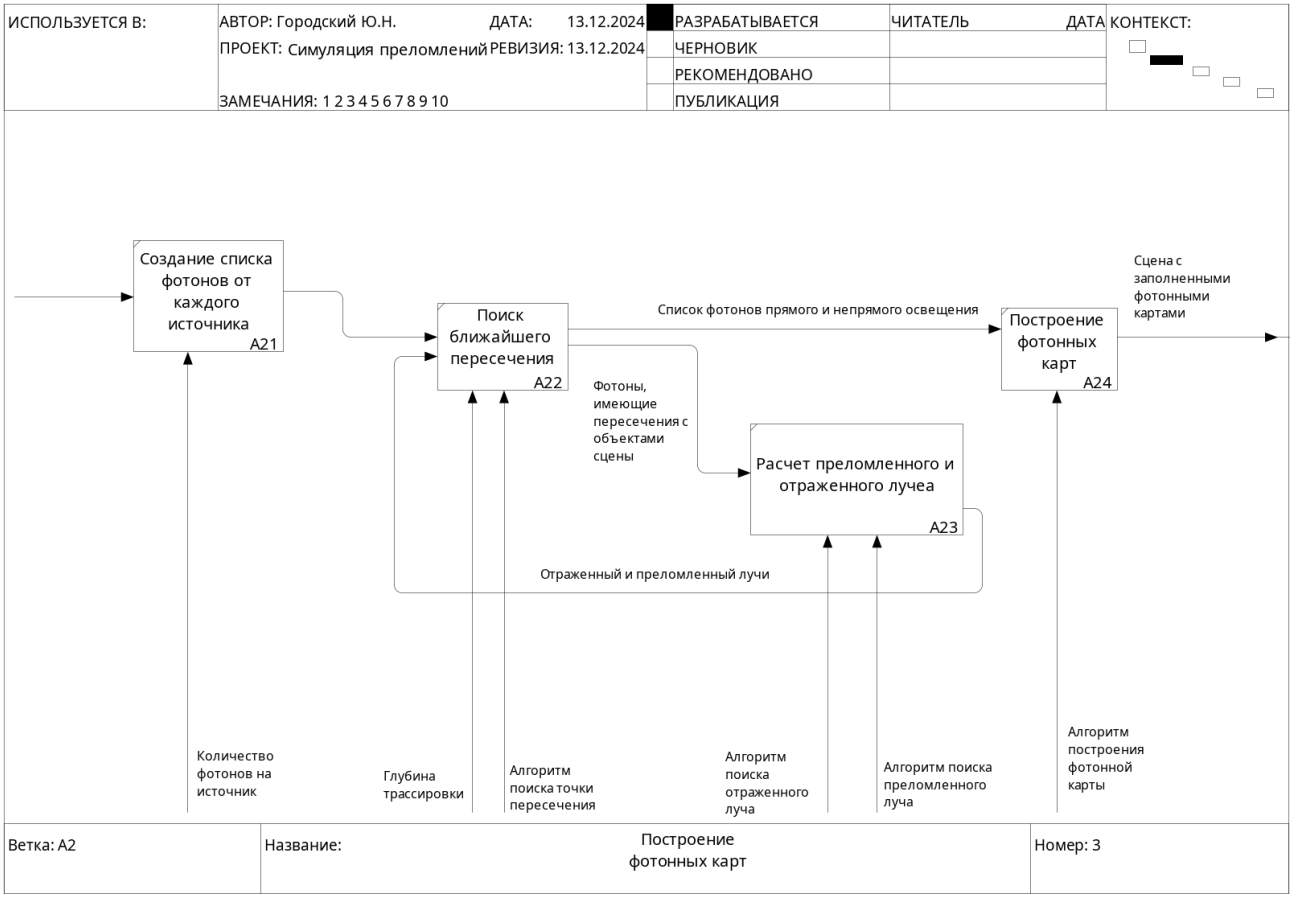


Рисунок 2.2 – Функциональная модель построения фотонных карт

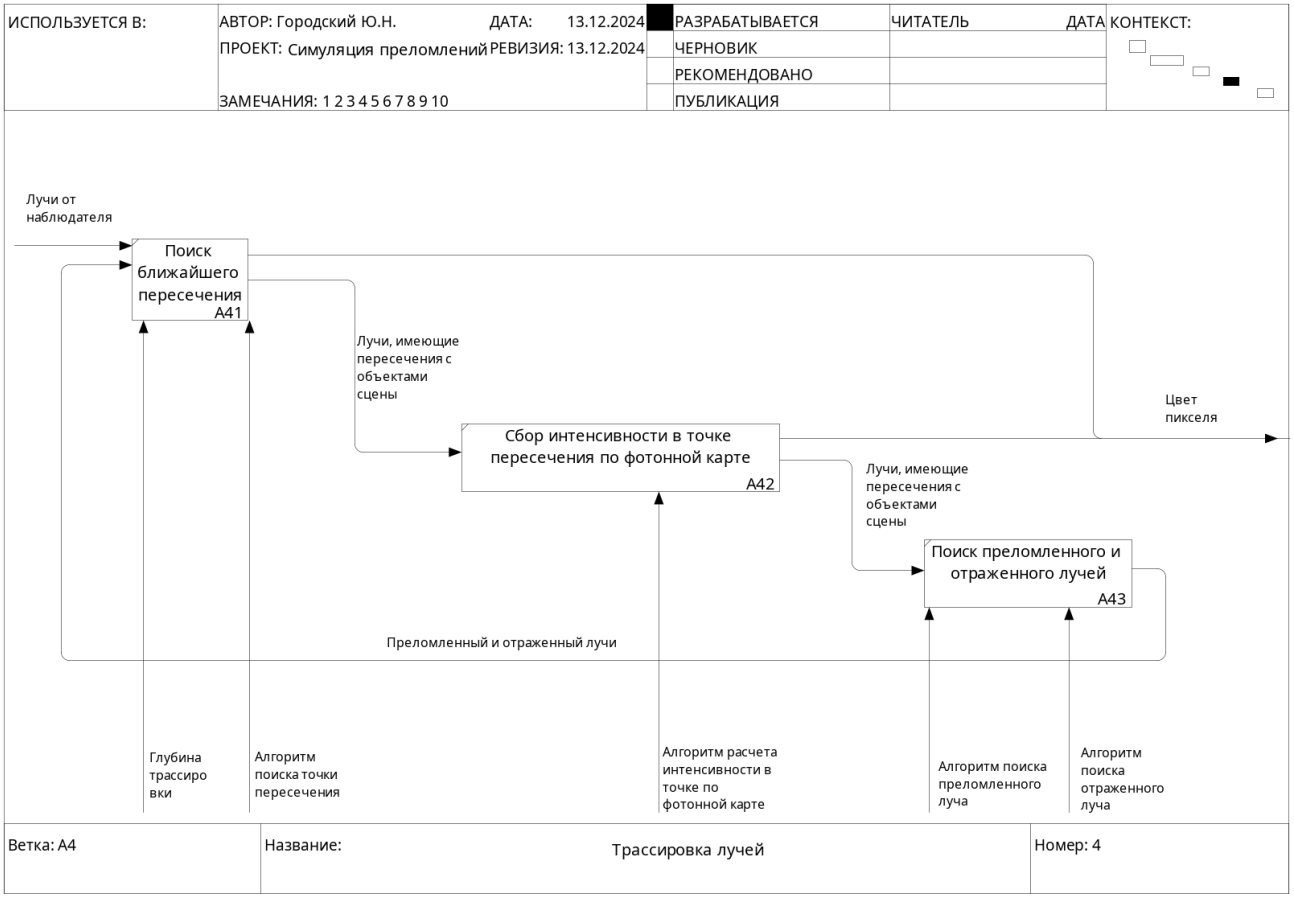


Рисунок 2.3 – Функциональная модель трассировки лучей

2.2 Используемые типы и структуры данных

В программном обеспечении используются следующие типы и структуры данных:

1. Сцена:
   - массив объектов сцены (сферы, линзы, полигоны);
   - наблюдатель (камера);
   - фотонная карта прямого освещения;
   - фотонная карта преломленного и отраженного освещения.
2. Камера:
   - координаты;
   - вектор направления;
   - угол зрения.

3. Сфера:

- радиус;
- расположение центра;
- графические параметры.

4. Полигон с 3 вершинами:

- 3 вершины;
- графические параметры.

5. Линза:

- расположение центра;
- направление центральной оси;
- радиус кривизны;
- общий радиус линзы (расстояние от центральной оси до края линзы);
- графические параметры.

6. Графические параметры:

- цвет поверхности;
- абсолютный коэффициент преломления;
- коэффициент прозрачности;
- коэффициент зеркальности;
- излучение.

7. Излучение

- цвет;
- интесивность.

8. Фотон:

- координаты;

– цвет;

– направление.

9. Узел фотонной карты

– фотон;

– указатель на правый узел;

– указатель на левый узел.

10. Фотонная карта

– указатель на первый узел.

2.3 Формальное описание алгоритмов

В разделе представлены схемы основных алгоритмов, использующихся в программном обеспечении.

2.3.1 Поиск персечения луча и сферы

на рисунке 2.4 представлен алгоритм поиска пересечения луча с сферой. Уравнение сферы можно записать как:

$$(P - C) \cdot (P - C) = r^2$$

где P — точка на поверхности сферы.

Параметрическое уравнение луча задается как:

$$P(t) = O + t \cdot D$$

где t — параметр, определяющий положение точки вдоль луча.

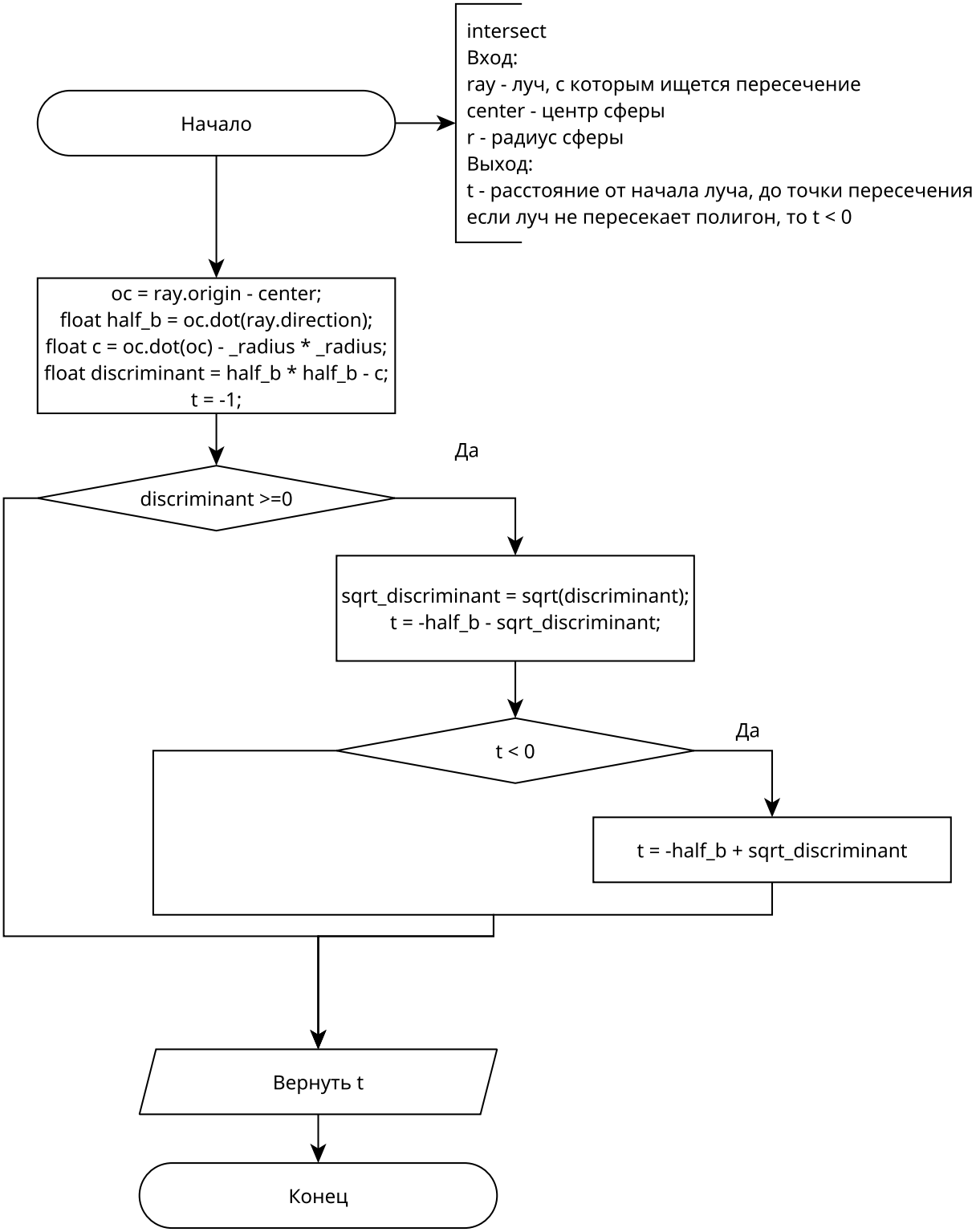


Рисунок 2.4 – Схема алгоритма поиска пересечения с сферой

2.3.2 Поиск пересечения луча и полигона

Для поиска пересечения луча с треугольным полигоном сначала необходимо проверить, пересекает ли луч плоскость, в которой расположен тре-

угольник. Это можно сделать, проверив, не является ли луч параллельным плоскости, что происходит, если скалярное произведение направления луча и нормали плоскости равно нулю:

$$(D, n) \neq 0$$

Если пересечение с плоскостью возможно, вычисляется параметр t, который определяет точку пересечения луча с плоскостью:

$$t = \frac{(A - V) \cdot n}{(D, n)}$$

где A, B и C — вершины треугольника, V — начальная точка луча, D — направление луча.

Точка пересечения P вычисляется по формуле:

$$P = V + t \cdot D$$

После нахождения точки пересечения необходимо проверить, лежит ли она внутри границы треугольника. Для этого задача сводится к двумерному пространству, обнуляя координаты по оси Z у вершин треугольника и точки пересечения.

Затем вычисляются векторные произведения векторов сторон треугольника и векторов, соединяющих первую вершину с точкой пересечения:

$$v_1 = [AB, AP], \quad v_2 = [BC, BP], \quad v_3 = [CA, CP]$$

Если все произведения имеют одинаковый знак, то точка пересечения P лежит внутри границы треугольника. Схему алгоритма представлена на рисунке 2.5

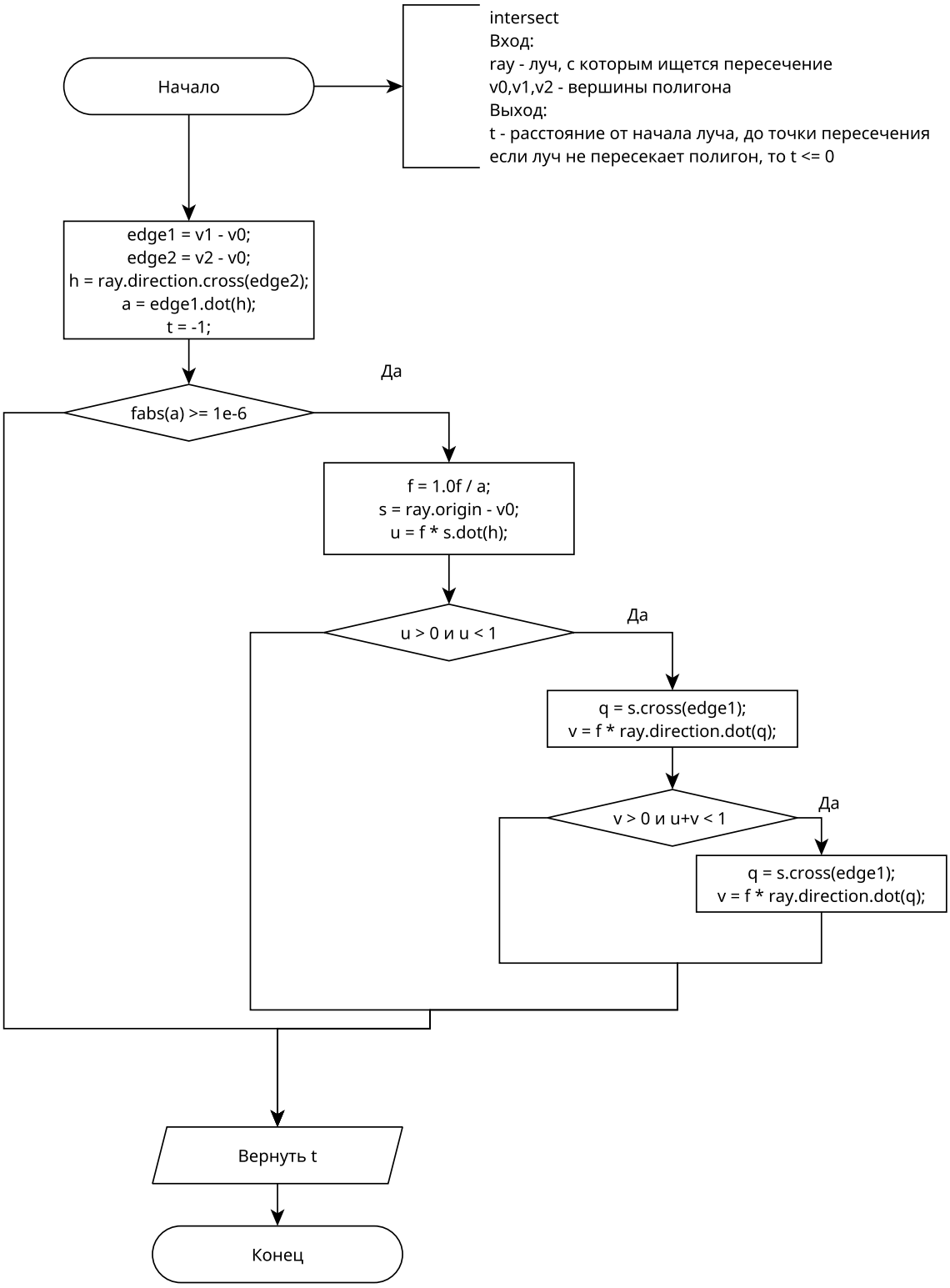


Рисунок 2.5 – Схема алгоритма поиска пересечения с полигоном

2.3.3 Персечение с оптической линзой

Поскольку линза задается пересечением двух сфер, поиск персечения луча с ней сводится к поиск пересечения с этими сферами. Схема алгоритма

представлена на рисунке 2.6.

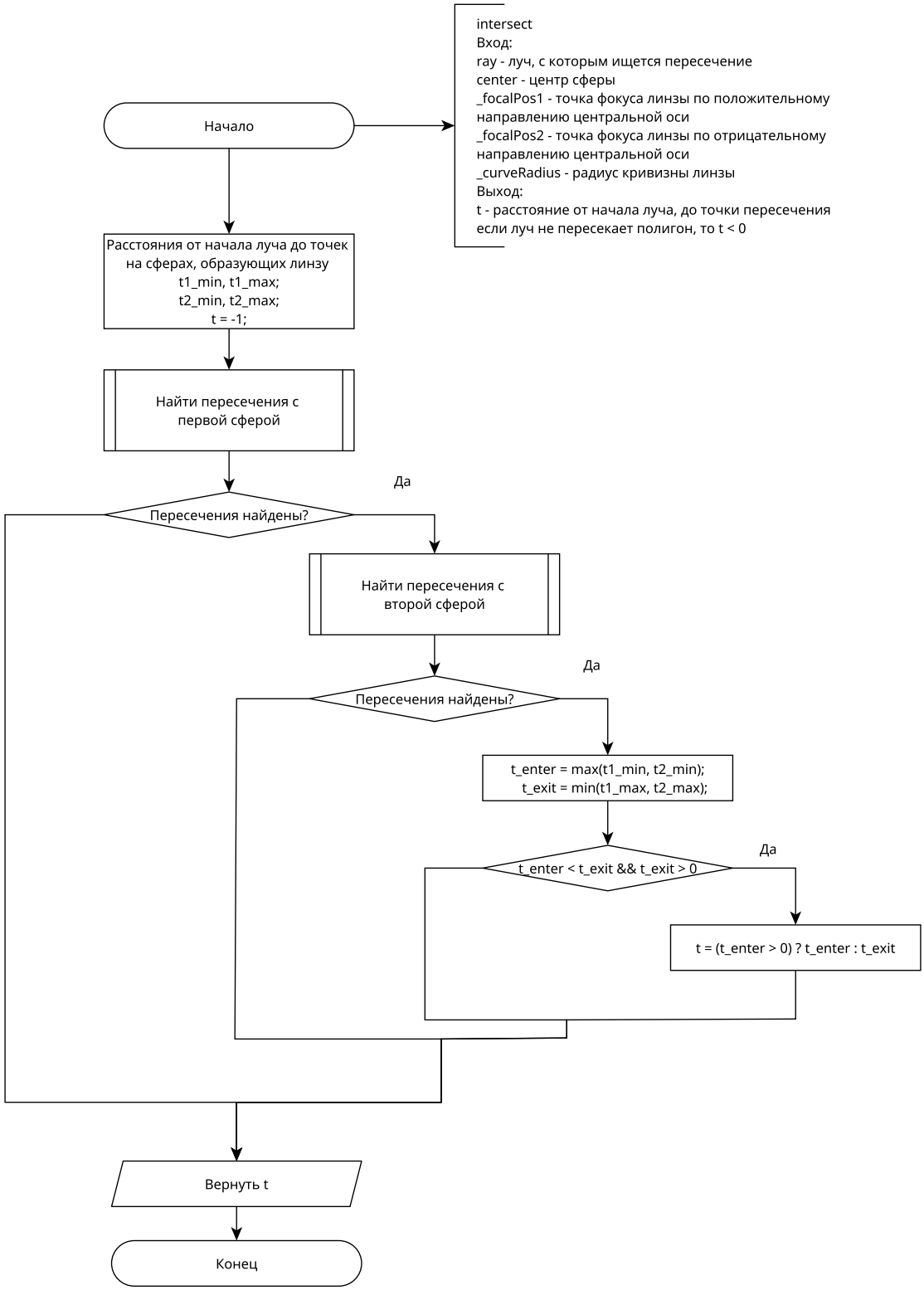


Рисунок 2.6 – Схема алгоритма поиска пересечения с линзой

2.3.4 Обратная трассировка

Алгоритм заключается в трассировке лучей от наблюдателя, пока не превысится глубина рекурсивного поиска или луч не будет пересекаться ни с одним объектом сйены. При пересечении с объектом испускается отраженный и преломленный если соответственно коэффициенты отражения и прозрачности объекта больше 0. Схема алгоритма представлена на рисунках 2.7 - 2.8.

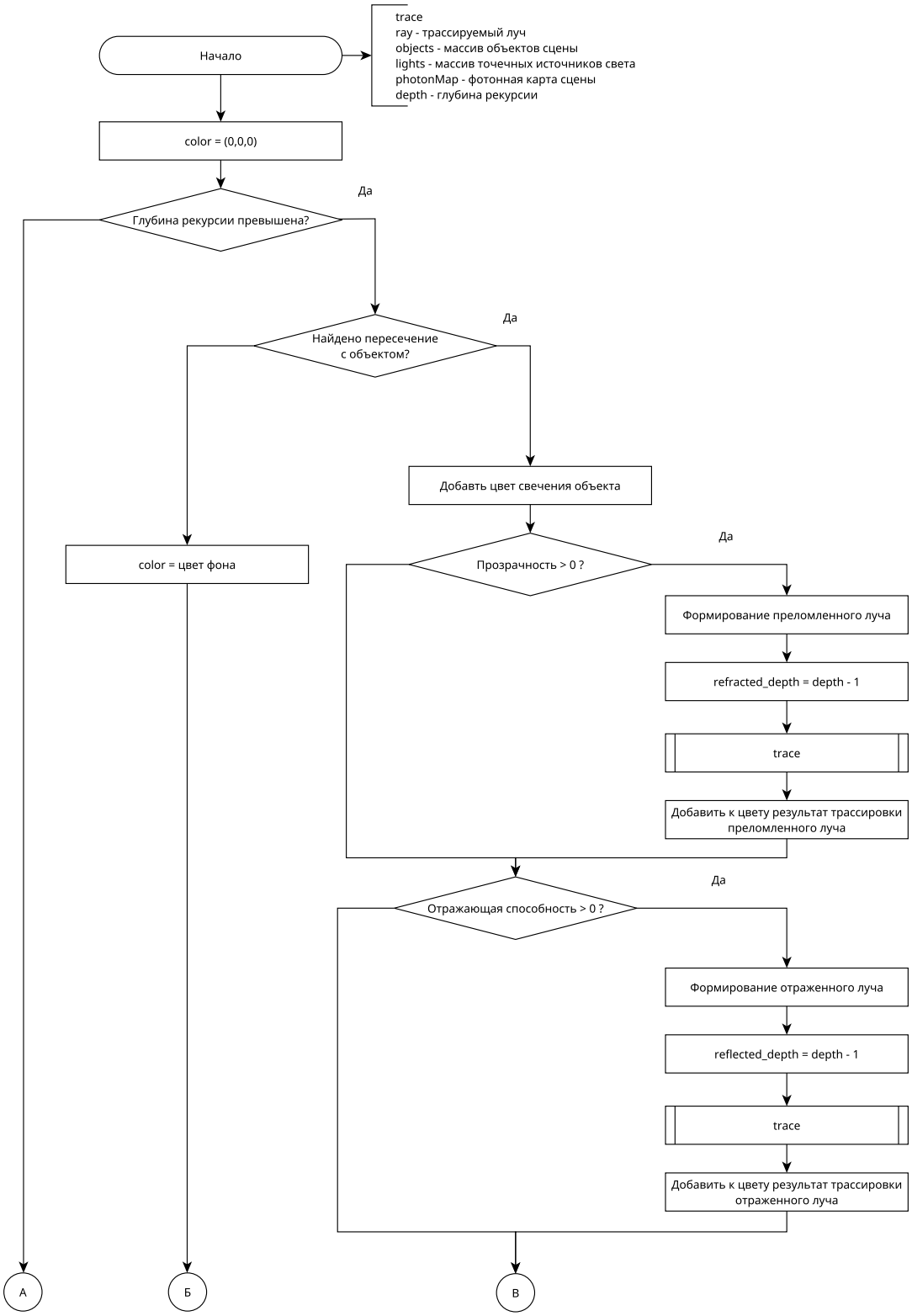


Рисунок 2.7 – Схема алгоритма обратной трассировки (часть 1)

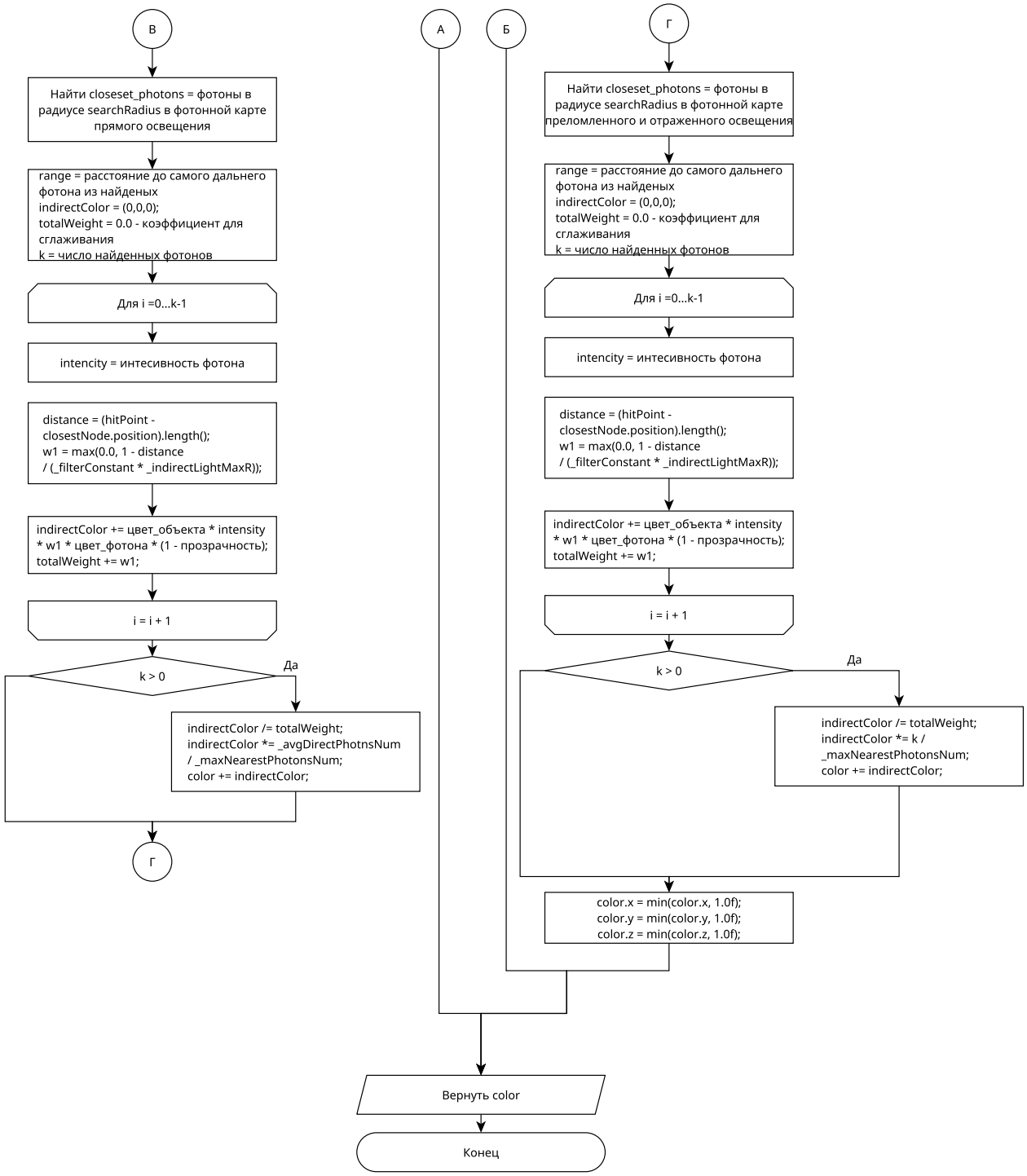


Рисунок 2.8 – Схема алгоритма обратной трассировки (часть 2)

2.3.5 Алгоритм построения фотонной карты

Фотонная карта хранит пересечения фотонов с объектами ввиде K-мерного дерева. При построении карты по списку фотонов список сначала упорядочивается по оси X, разбивается пополам, и происходит рекурсивный вызов функции. На следующем шаге рекурсии разбиение происходит по оси Y, потом Z, посе чего снова по X. Схема алгоритма представлена на рисунке

2.9.

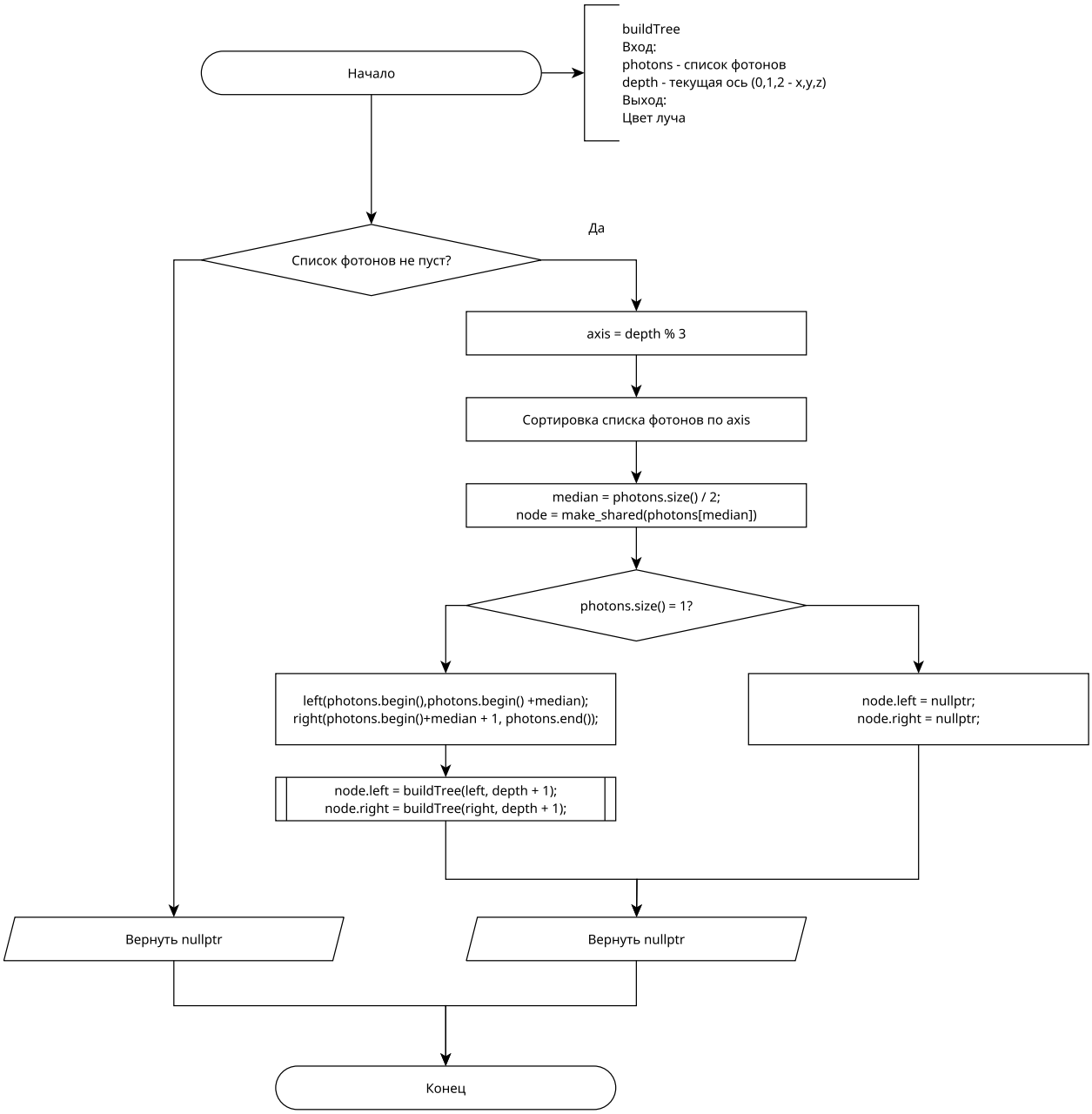


Рисунок 2.9 – Схема алгоритма построения фотонной карты по списку фотонов

3 Технологическая часть

В разделе представлены средства разработки программного обеспечения и детали реализации.

3.1 Средства реализация

Для реализации программного обеспечения выбран язык C++ [10]. Выбор обуславливается тем, что язык представляет весь необходимый для решения поставленной задачи функционал.

Для создания пользовательского интерфейса ПО был использован фреймворк QT[11]. Фреймворк содержит объекты, позволяющие напрямую работать с пикселями изображения, а так же возможности создания интерактивных пользовательских интерфейсов.

Для сборки программного обеспечения использовался инструмент QMake[11].

В качестве среды разработки была выбрана среда разработки QtCreator[11].

3.2 Используемые классы

Используемые классы представлены на рисунке 3.1. Класс BaseObject является базовым для объектов сцены, класс Camera содержит описание наблюдателя, класс Scene хранит объекты сцены, наблюдателя и фотонные карты прямого и непрямого освещения сцены. В классе Brawer реализованы функции, отвечающие за построения изображения сцены.

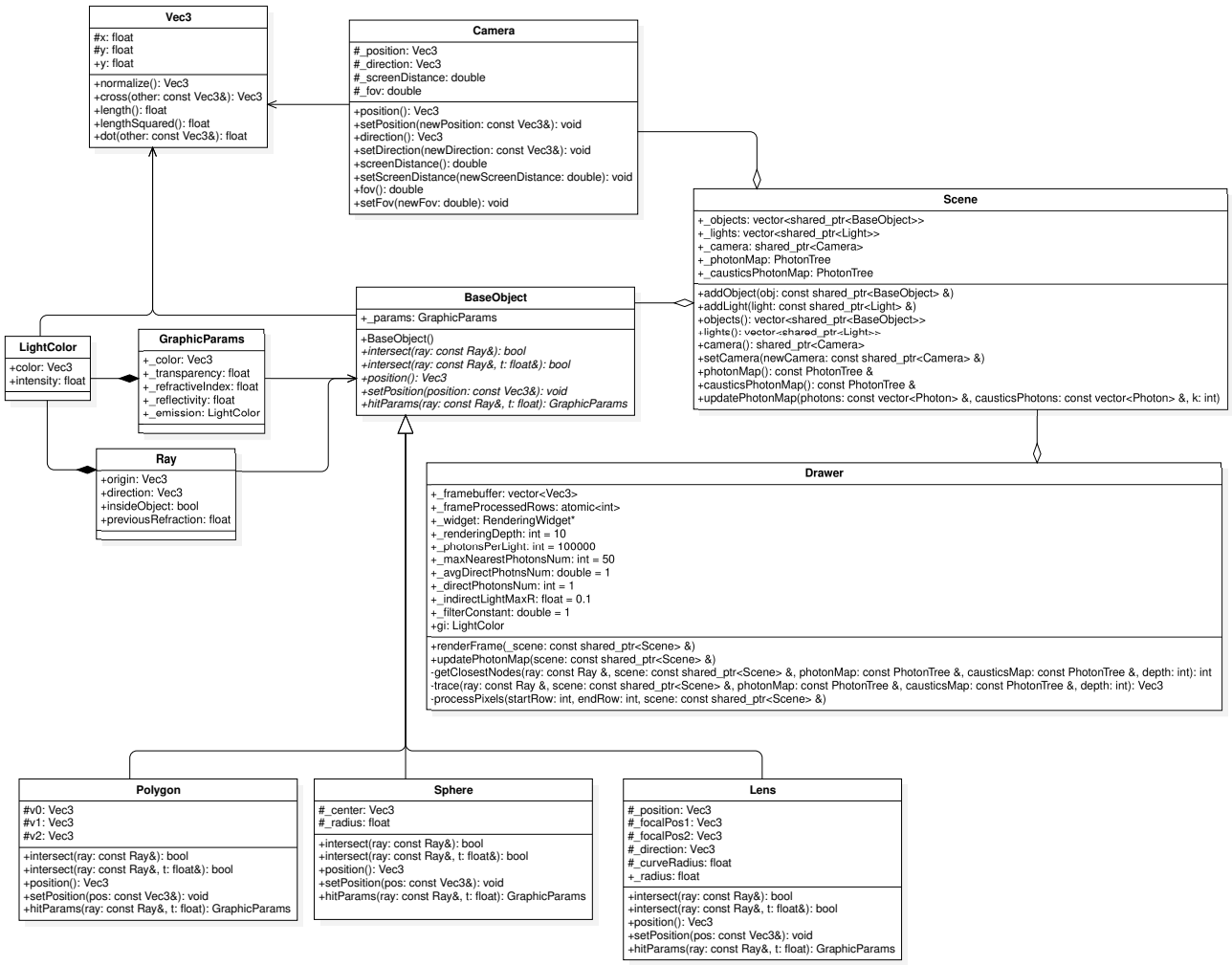


Рисунок 3.1 – UML диаграмма используемых классов

3.3 Пользовательский интерфейс

Интерфейс реализуемого ПО представлен на рисунках 3.2 – 3.7.

На рисунке 3.2 представлены параметры отрисовки, пользователь может задать число фотонов, испускаемых каждым источником света, радиус сбора освещенности и коснтанту фильтра, который используется в алгоритме сбора освещенности от ближайших фотонов, определяя влияния расстояния от точки до фотона на освещенность от этого фотона.

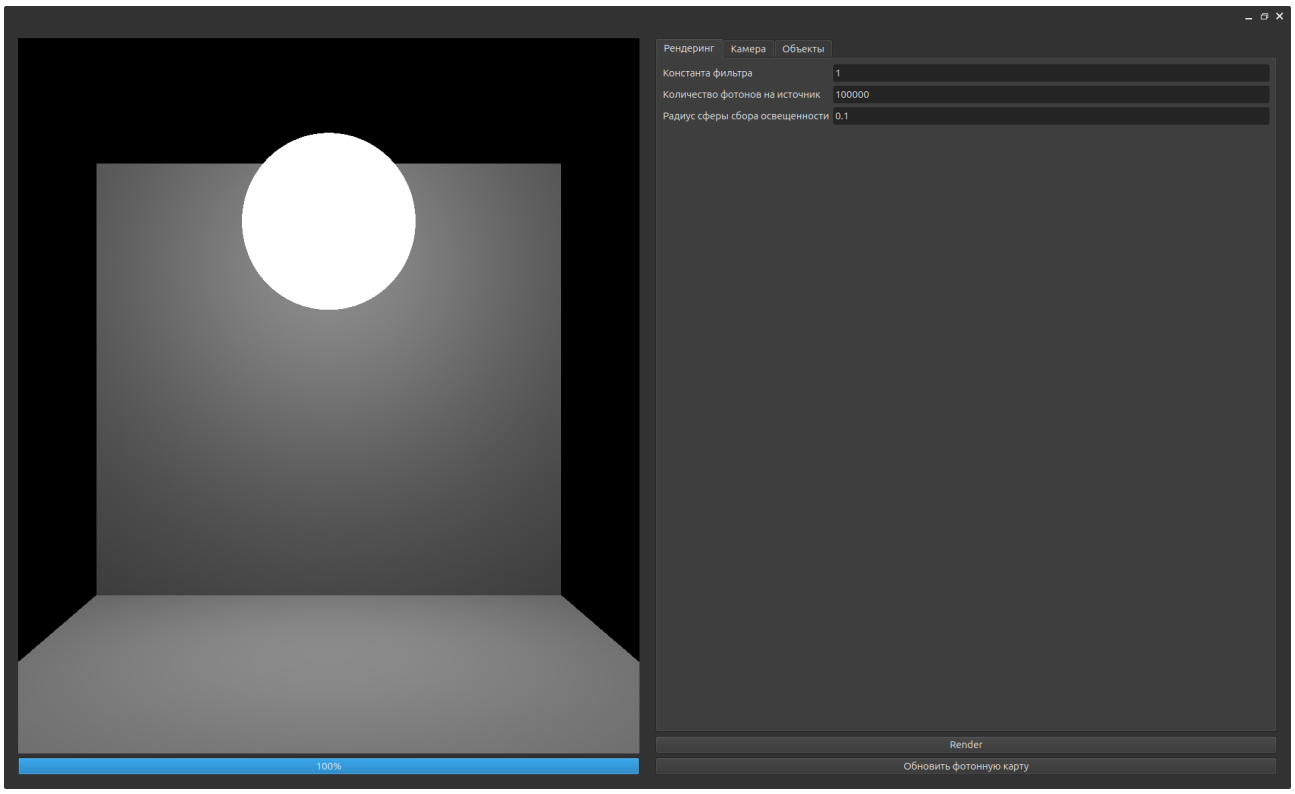


Рисунок 3.2 – Виджет параметров отрисовки

На рисунке 3.3 представлены параметры наблюдателя. Пользователь может задать положения камеры, направление ее взгляда и угол обзора.

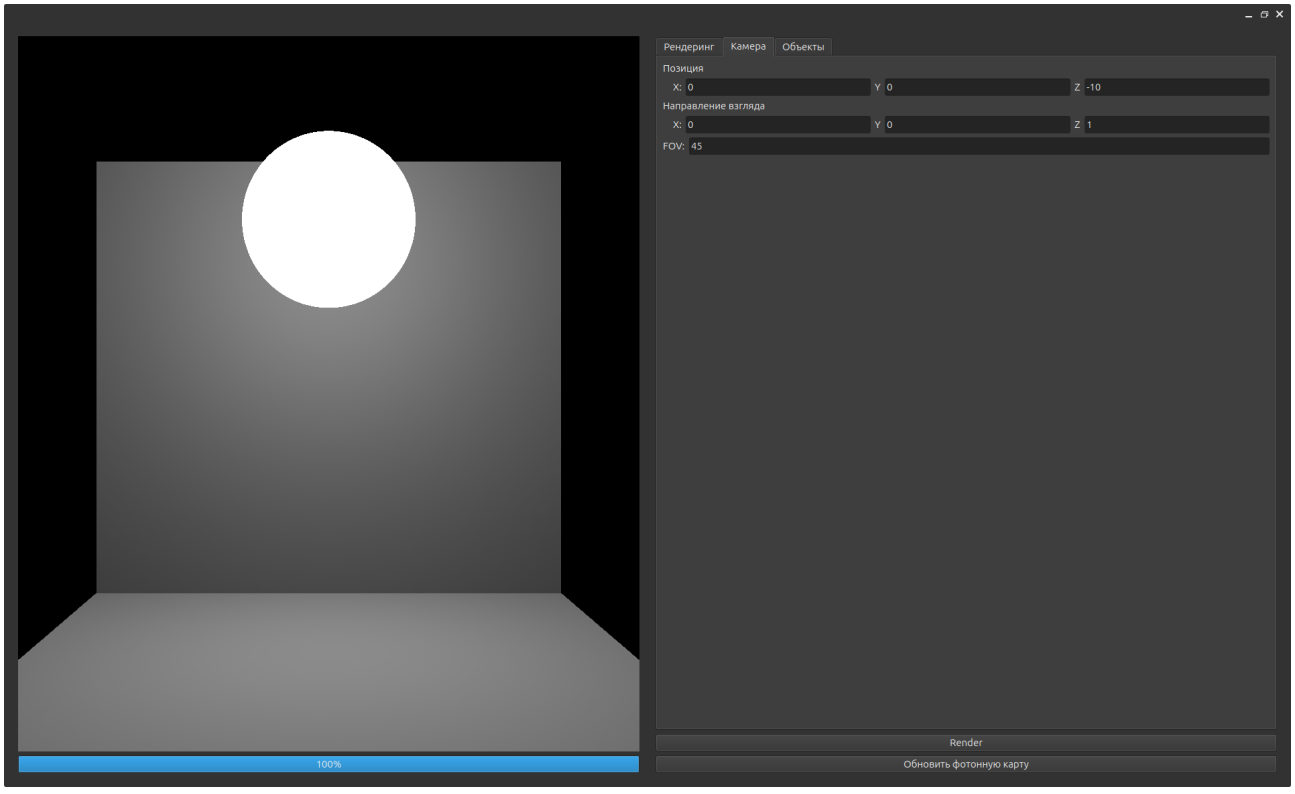


Рисунок 3.3 – Виджет параметров камеры

На рисунке 3.4 представлены параметры сферы на сцене. Пользователь может определить положение, радиус, прозрачность, зеркальность, коэффициент преломления, цвет поверхности сферы, цвет и интенсивность излучения сферы.

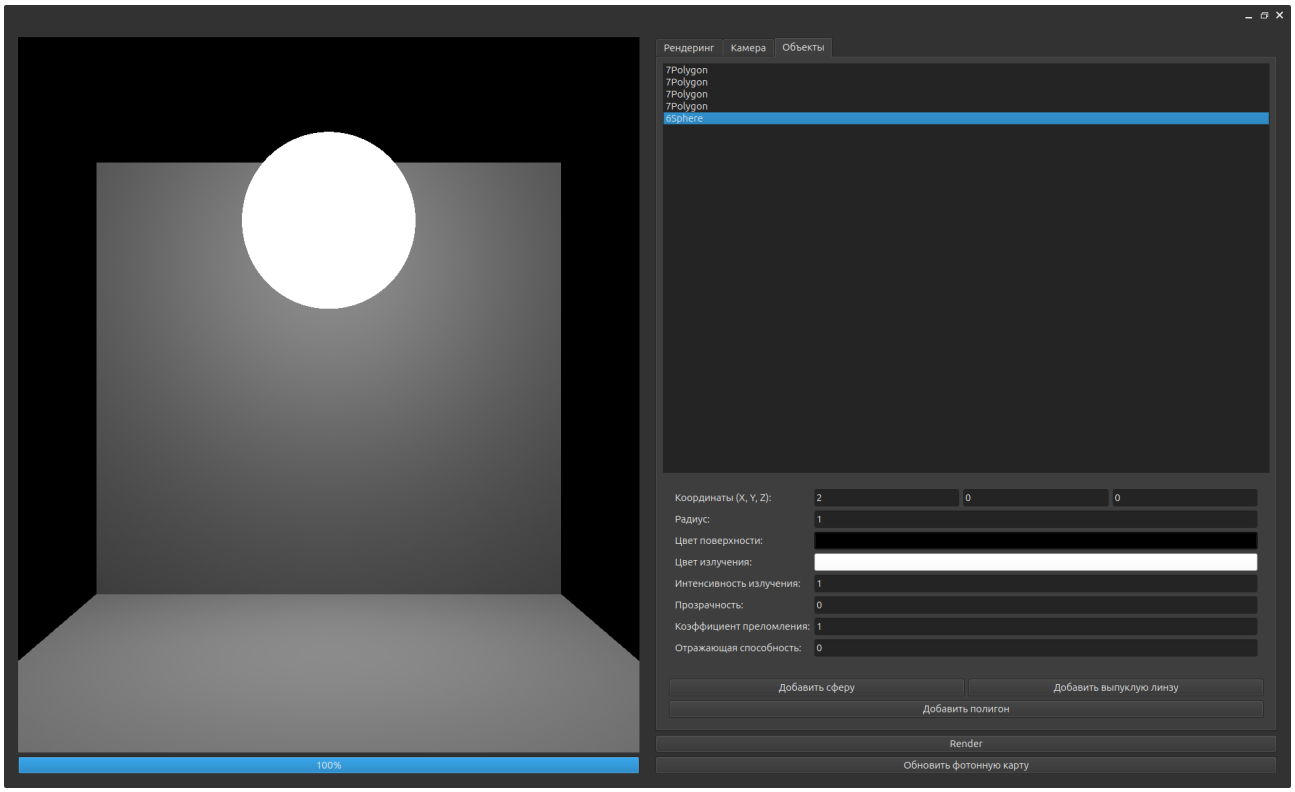


Рисунок 3.4 – Виджет параметров сферы

На рисунке 3.5 представлены параметры треугольного полигона. Есть возможность задать положение трех его точек и графические параметры.

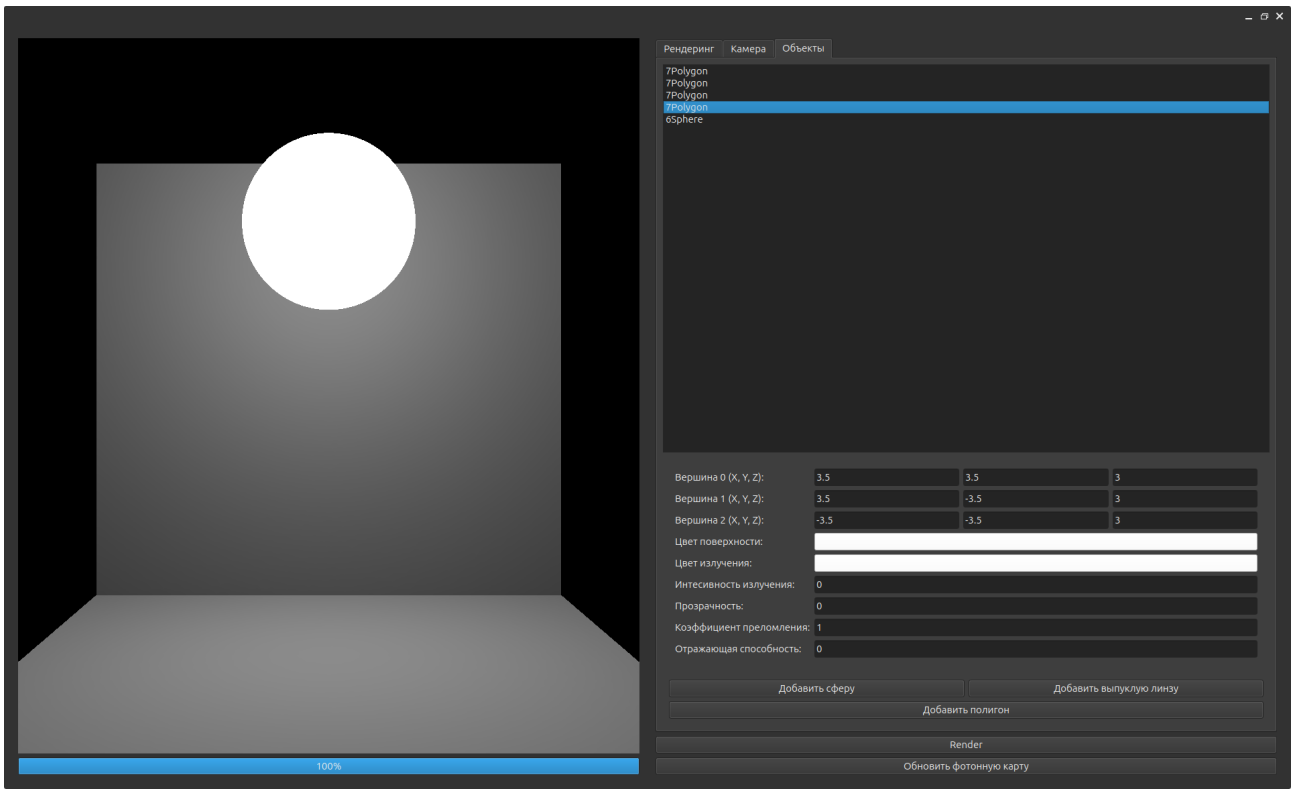


Рисунок 3.5 – Виджет параметров полигона

На рисунке 3.6 представлены параметры линзы, имеется возможность изменить радиус кривизны, общий радиус, графические параметры и положение в пространстве.

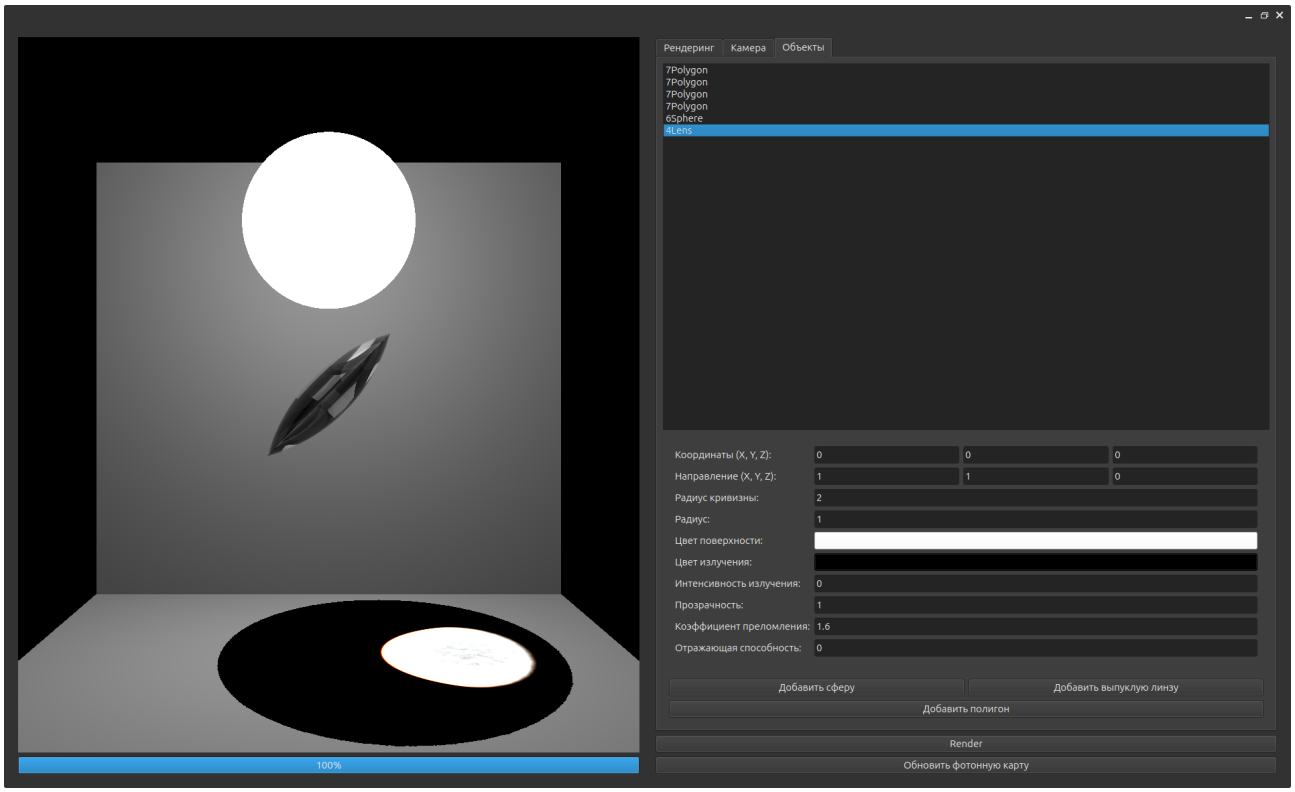


Рисунок 3.6 – Виджет параметров линзы

На рисунке 3.7 представлен пример работы программы с несколькими прозрачными объектами.

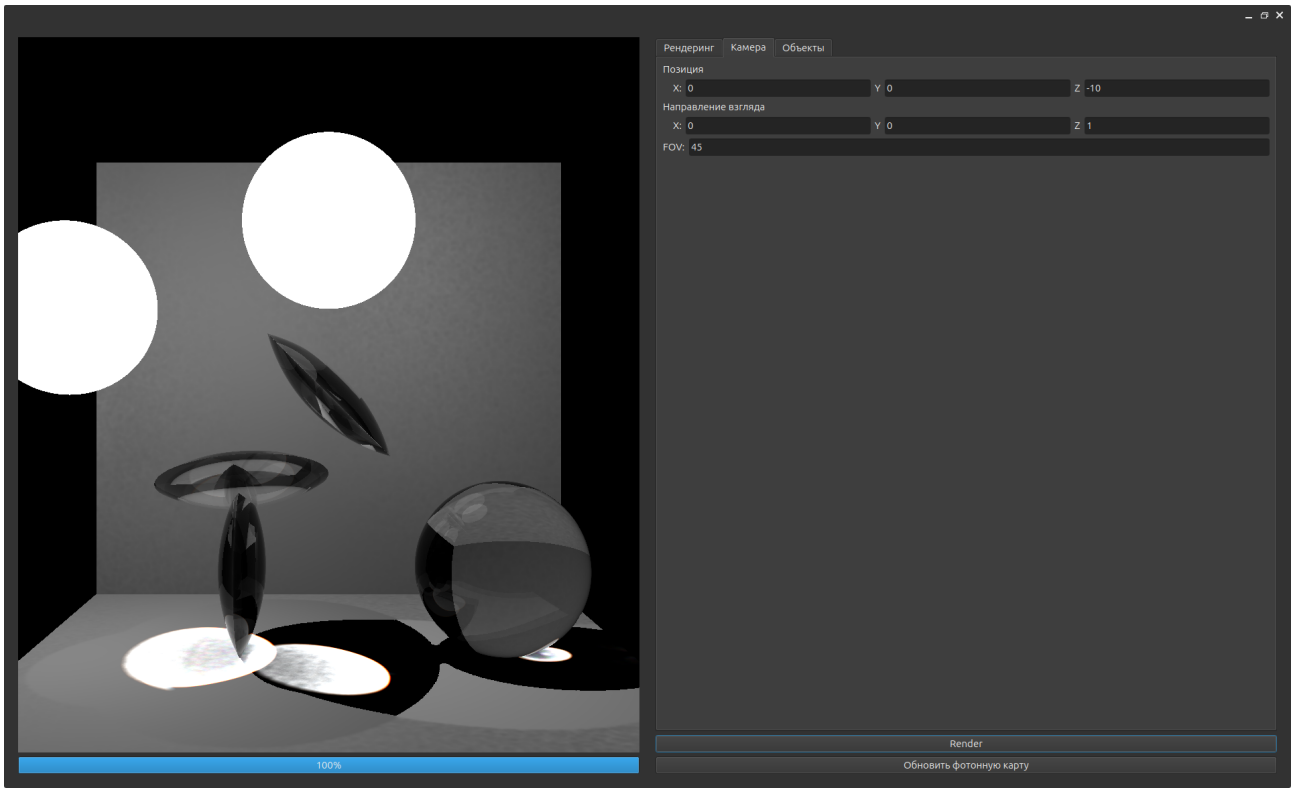


Рисунок 3.7 – Пример работы программы

3.4 Вывод

В разделе были представлены средства разработки программного обеспечения, детали реализации, пользовательский интерфейс. Также были проедемонстрированы примеры работы программы.

4 Исследовательская часть

В данном разделе описан эксперимент по определению влияния числа фотонов, испускаемых источником света, и радиуса сбора освещенности на качество изображения и время его создания.

4.1 Цель эксперимента

Целью эксперимента является определение оптимального числа фотонов для отрисовки заданной сцены.

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялось исследование:

- процессор: Intel Core™ i5-13420H [12] 2.1ГГц;
- память: 16 Гб;
- операционная система: Ubuntu 24.04.1 LTS [13] Linux [14].

Эксперимент проводился на ноутбуке, включенном в сеть электропитания, система работала в производительном режиме без ограничений на нагрузку процессора. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения рабочего стола, окружением рабочего стола и непосредственно системой тестирования.

4.3 Описание эксперимента

Создана сцена состоящая из 2 прозрачных, 1 непрозрачного объекта и источника света. По сцене строилось изображение размером 900x900 пикселей для разного числа фотонов, испускаемых источником: от 100000 до 1100000 с шагом в 250000 фотонов, радиус сбора освещенности: от 0.05 до 0.1 с шагом 0.25. Рассматриваемя сцена представлена на рисунке 4.1.

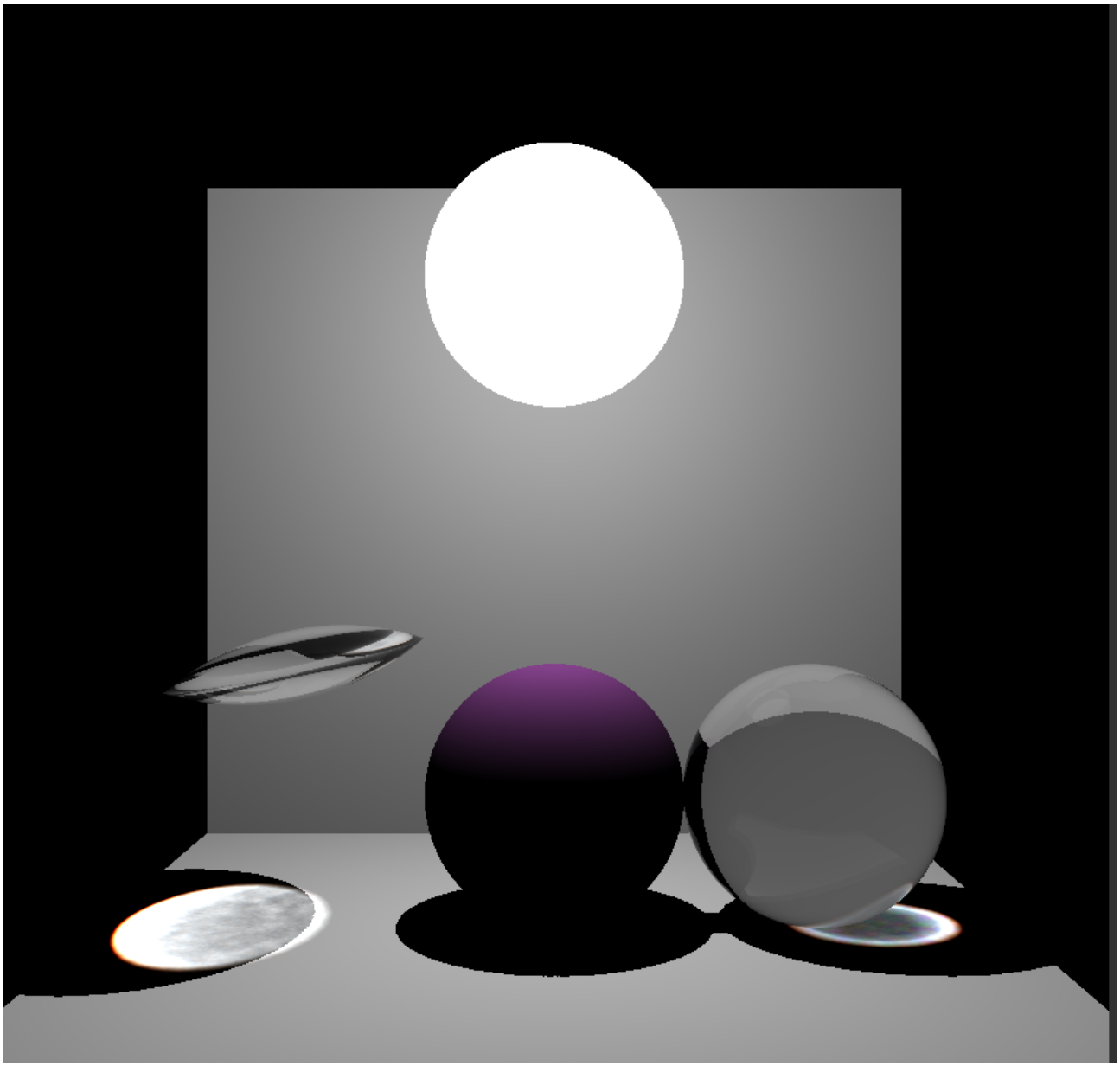


Рисунок 4.1 – Сцена эксперимента

4.4 Зависимость времени построения карты от числа фотонов на источник

В таблице 4.1 представлена зависимость времени построения карты от числа фотонов на источник. Исследование проводилось при различных значениях числа фотонов.

По таблице 4.1 построен график, изображенный на рисунке 4.2.

По результатам видно, что время построения карты линейно зависит от числа фотонов, испускаемых источником.

Таблица 4.1 – Время построения карты в зависимости от числа фотонов

Число фотонов на источник	Время построения (с)
100000	1.794
350000	6.722
600000	11.948
850000	17.718
1100000	23.214

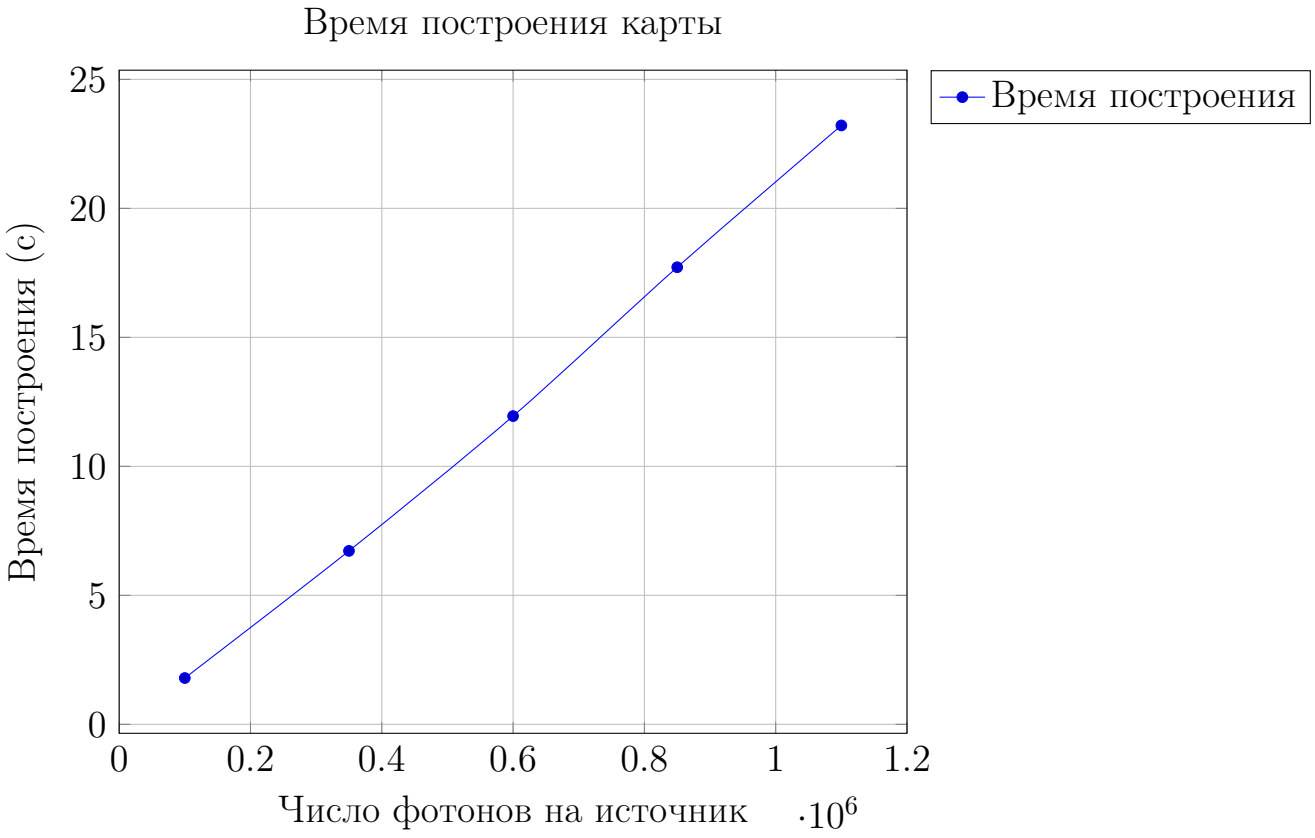


Рисунок 4.2 – График зависимости времени построения карты от числа фотонов на источник

4.5 Зависимость времени отрисовки от радиуса сбора освещенности

В таблице 4.2 представлена зависимость времени отрисовки от радиуса сбора освещенности. Исследование проводилось при различных радиусах и числах фотонов.

Таблица 4.2 – Зависимость времени отрисовки от радиуса сбора освещенности

Радиус сбора	Число фотонов	Время (с)	Изображение
0.025	100000	9.264	Рисунок 4.4a
	350000	16.019	Рисунок 4.4b
	600000	20.838	Рисунок 4.4c
	850000	25.42	Рисунок 4.4d
	1100000	29.201	Рисунок 4.4e
0.050	100000	12.602	Рисунок 4.4f
	350000	25.423	Рисунок 4.4g
	600000	36.297	Рисунок 4.4h
	850000	44.543	Рисунок 4.4i
	1100000	55.177	Рисунок 4.4j
0.075	100000	16.722	Рисунок 4.4k
	350000	37.342	Рисунок 4.4l
	600000	56.473	Рисунок 4.4m
	850000	73.379	Рисунок 4.4n
	1100000	89.379	Рисунок 4.4o
0.100	100000	22.153	Рисунок 4.4p
	350000	53.549	Рисунок 4.4q
	600000	83.571	Рисунок 4.4r
	850000	110.001	Рисунок 4.4s
	1100000	135.275	Рисунок 4.4t

По таблице 4.2 был построен график, изображенный на рисунке 4.3. По результатам можно сделать вывод, что время создания изображения имеет линейную зависимость от числа фотонов, испускаемых источником.

На рисунке 4.4 представлены части изображений с наиболее характерными деффектами для разного радиуса сбора освещенности и числа фотонов.

По результам эксперимента можно сделать вывод, что наиболее детализированными являются изображения с минимальным радиусом сбора. Однако такая настройка требует большего количества фотонов, что можно заметить по черным точкам на частях изображения с прямым освещением. Несмотря на то, что для правильной отрисовки требуется большое количество фотонов, при меньшем радиусе сбора время отрисовки будет все равно меньше, это связано с временем поиска в деревьях фотонных карт, потому что при большем

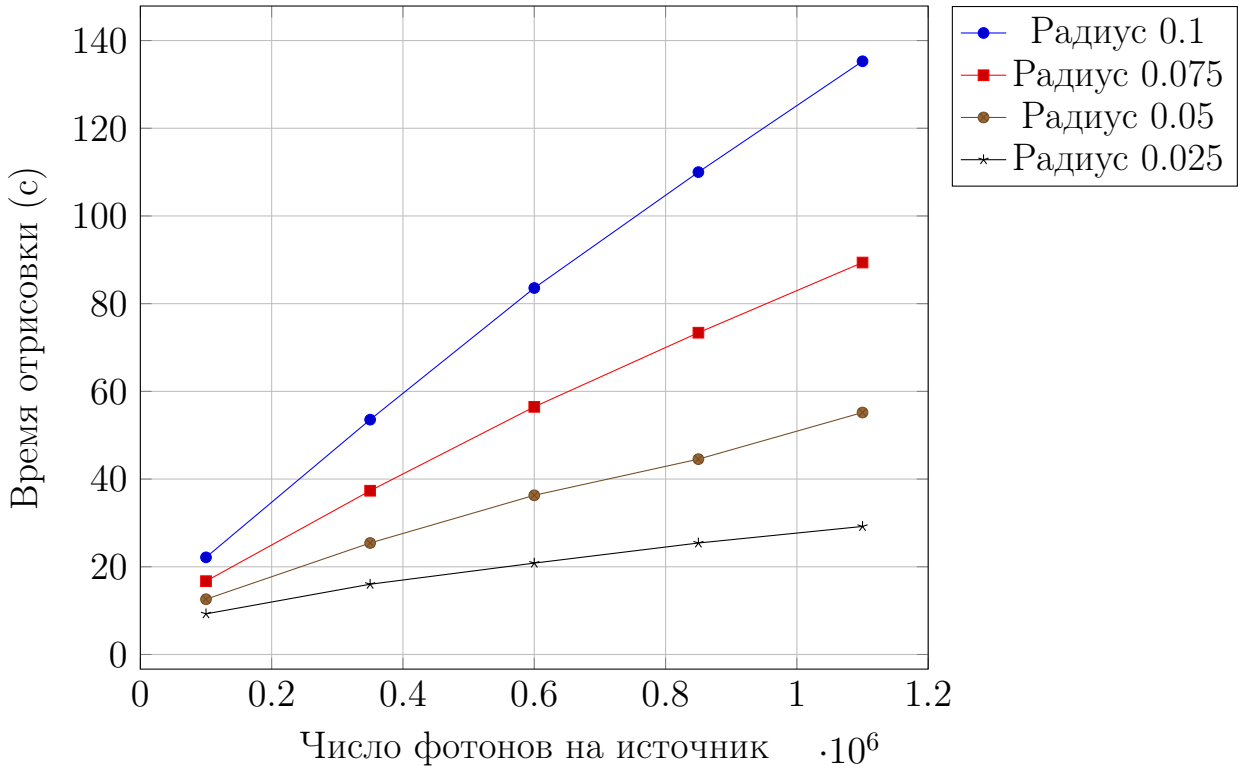


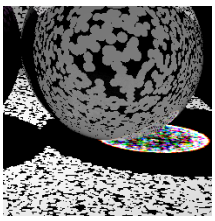
Рисунок 4.3 – График зависимости времени отрисовки от числа фотонов

радиусе большее число фотонов попадает в область поиска. К тому же при большем радиусе сбора освещенности возникают дефекты на краях теней, как, например, «перемычка» между тенями двух сфер, на изображениях, полученных с радиусом 0.05 - 0.1. На изображениях, полученных с радиусом 0.025, такого эффекта не наблюдается. Также явление дисперсии более заметно при меньшем радиусе сбора.

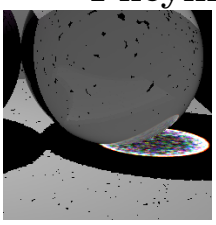
4.6 Вывод

В исследовательской части были описаны характеристики устройства, на котором проводились замеры, а также проведенный эксперимент. Результаты показывают, что для увеличения детализированности освещения необходимо увеличивать число фотонов, при этом уменьшая радиус сбора освещенности. При этом скорость работы возрастает при уменьшении радиуса сбора.

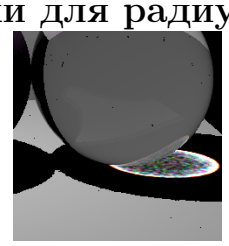
Рисунки для радиуса 0.025



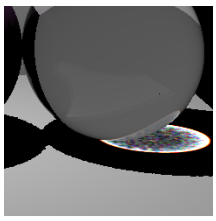
(a) Число фотонов: 100000.



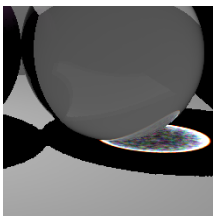
(b) Число фотонов: 350000.



(c) Число фотонов: 600000.

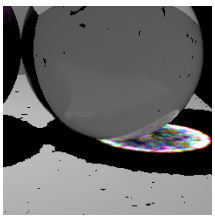


(d) Число фотонов: 850000.

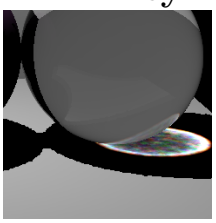


(e) Число фотонов: 1100000.

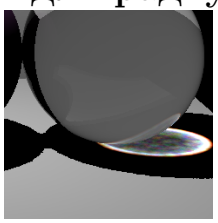
Рисунки для радиуса 0.050



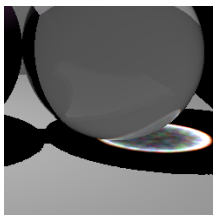
(f) Число фотонов: 100000.



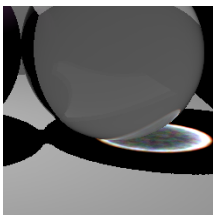
(g) Число фотонов: 350000.



(h) Число фотонов: 600000.

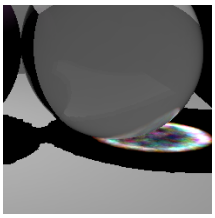


(i) Число фотонов: 850000.

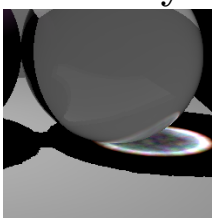


(j) Число фотонов: 1100000.

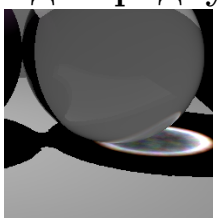
Рисунки для радиуса 0.075



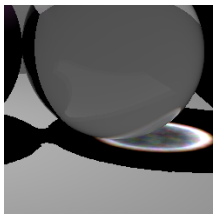
(k) Число фотонов: 100000.



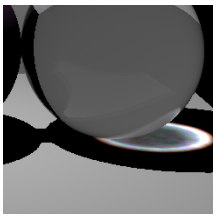
(l) Число фотонов: 350000.



(m) Число фотонов: 600000.



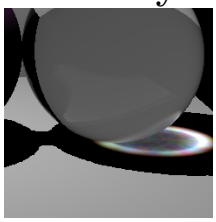
(n) Число фотонов: 850000.



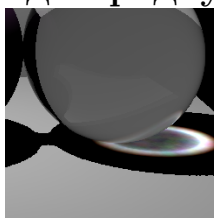
(o) Число фотонов: 1100000.

Рисунки для радиуса 0.100

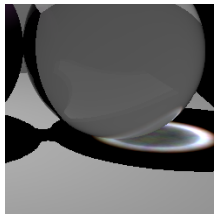
(p) Число фотонов: 100000.



(q) Число фотонов: 350000.



(r) Число фотонов: 600000.



(s) Число фотонов: 850000.

(t) Число фотонов: 1100000.

Рисунок 4.4 – Результаты для различных радиусов сбора и количества фотонов.

ЗАКЛЮЧЕНИЕ

В рамках курсовой работы было создано программное обеспечение, позволяющее визуализировать преломление света через оптические линзы. Разработанная система предоставляет пользователям возможность задавать материалы и другие параметры объектов, а также изменять положение точки наблюдения и характеристики источников света в процессе работы. В ходе выполнения проекта были успешно решены следующие задачи:

- рассмотрены физические основы явления;
- проанализированы и выбраны модели представления трехмерных объектов;
- выбран оптимальный алгоритм для решения поставленной задачи;
- спроектирована архитектура программы и пользовательского интерфейса;
- реализованы необходимые структуры данных и алгоритмы;
- описана структура разработанного программного обеспечения;
- продемонстрирована работоспособность программы;
- проведено исследование производительности созданного ПО.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *А.М. П.* Физическая энциклопедия. Т. 1. — Рипол Классик, 1988.
2. *И.В. С.* Курс общей физики. Том 4. Волны, оптика: Учебное пособие. — 5-е изд. — СПб.: Издательство Лань¨, 2021.
3. *Лебедева И.* Обзор методов расчета освещенности сцены в рамках решения проблемы реалистической визуализации трехмерных объектов // Вестник МГСУ. — 2011. — № 2—2. — С. 367—371.
4. *Жданов А.*, *Жданов Д.*, *Бирюков Е.* Реалистичный рендеринг на основе прямых и обратных фотонных карт // Препринты ИПМ им. МВ Келдыша. — 2020. — № 77. — С. 1—22.
5. *Цапко И. В.*, *Цапко С. Г.* Алгоритмы и методы обработки информации в задачах трехмерного сканирования объектов // Известия Томского политехнического университета. Инжиниринг георесурсов. — 2010. — Т. 317, № 5.
6. *Камалидинова Э.*, *Рожина И.* Программы построения трехмерных графических изображений // Актуальные вопросы преподавания математики, информатики и информационных технологий. — 2015. — № 1. — С. 82—90.
7. Построение реалистичных изображений при помощи алгоритма Варнока и сравнение эффективности различных его реализаций: выпускная бакалаврская работа по направлению подготовки: 01.03. 02-Прикладная математика и информатика / В. А. Провкин [и др.]. — 2017.
8. *Афанасьев В.* Задача обратной трассировки лучей в расширенной постановке // Программные продукты и системы. — 2006. — № 3. — С. 33—38.
9. *McHenry K.*, *Bajcsy P.* An Overview of 3D Data Content, File Formats and Viewers. — 2008.
10. Standard C++ [Электронный ресурс]. — Режим доступа: `https://isocpp.org/` (дата обращения: 01.12.2024).

11. Qt | Cross-platform software development for embedded & desktop [Электронный ресурс]. — Режим доступа: https://www.qt.io/ (дата обращения: 01.12.2024).

12. Core i5-13420H: технические характеристики и тесты [Электронный ресурс]. — Режим доступа: https://technical.city/ru/cpu/Core-i5-13420H (дата обращения: 12.11.2024).

13. Ubuntu 24.04.1 (Noble Numbat) [Электронный ресурс]. — Режим доступа: https://releases.ubuntu.com/24.04/ (дата обращения: 01.12.2024).

14. Linux [Электронный ресурс]. — Режим доступа: https://www.linux.org (дата обращения: 01.12.2024).