Содержание

Введение

Компьютерная графика является неотъемлемой частью жизни человека. Она обладает высоким спросом: её используют сфере развлечений, например, эффекты в кинофильмах и сериалах, анимации в играх; в научных и инженерных дисциплинах, каких как, медицина, геофизика, ядерная физика, картография, для визуализации и лучшего восприятия информации. Компьютерную графику так же можно использовать в школе на уроках физики, информатики, математики для демонстрации различных явлений.

Прогресс компьютерной графики не стоит на месте: со временем развиваются и оптимизируются алгоритмы, которые позволяют получить реалистичное трехмерное изображение. Вместе с прогрессом растут и требования к реалистичности, что приводит к росту как сложности алгоритмов, так и к затратам ресурсов компьютера, таких как оперативная память и время работы алгоритма.

Целью курсовой работы является разработка программного обеспечения визуализации геометрических объектов, расположенных за полупрозрачной сферой.

Для достижения поставленной цели необходимо решить следующие **задачи**:

1) формализовать объекты сцены;
2) выбрать и описать алгоритмы реализации, необходимые для визуализации поставленной задачи;
3) составить требования к программному продукту;
4) реализовать выбранные алгоритмы;
5) исследовать время работы реализации алгоритма от количества потоков.

1 Аналитическая часть

В данном разделе выполняется формализация объектов сцены, а также приводятся существующие методы и алгоритмы для решения данной задачи.

1.1 Формализация объектов сцены

Сцена состоит из следующих объектов:

1) прозрачная сфера — видимый объект, который задается аналитически, коэффициенты преломления и прозрачности задаются пользователем;
2) геометрические объекты — видимые объекты из ограниченного набора, представленного ниже, каждый из которых характеризуется своими параметрами и цветом, который задает пользователь;
3) плоскость, параллельная координатной плоскости Oyz — видимый объект, ограничивающий снизу положение других объектов на сцене;
4) камера — невидимый точечный объект, с помощью которого рассчитывается перспективное отображение объектов сцены;
5) источник света – невидимый точечный объект, который задается тремя координатами положения.

В ограниченный набор объектов входят следующие виды геометрических примитивов:

1) шар;
2) треугольная и четырех угольная пирамиды;
3) параллелепипед;
4) от треугольной до восьмиугольной призмы.

Шар характеризуется координатами центра и радиусом. Остальные объекты — набором точек и связей между ними.

1.2 Обзор и анализ способов задания трехмерной модели

Модель — отображение форм и размеров объекта, чье основное предназначение – правильно отобразить форму и размер конкретного объекта [1].

Используется три вида модели: каркасная, поверхностная и твердотельная.

Каркасная модель. Простейший вид модели, состоит из вершин и ребер, однако она может некорректно передать представление о форме и размерах объекта.

Поверхностная модель. Поверхность в такой модели может задаваться как аналитически, так и другим способом, например как поверхности того или иного вида с помощью, например, полигональной аппроксимации. Ее недостатком является отсутствие информации о том, с какой стороны находится материал, а с какой – пустота.

Твердотельная модель. Данная модель отличается от поверхностной тем, что включает информацию о том, с какой стороны находится материал. Чаще всего для этого используется направление внутренней нормали.

Для решения поставленной задачи выбор был сделан в пользу твердотельной модели, так как каркасная модель не предоставляет полную информацию об объекте, а также необходимо знать с какой стороны объекта находится материал.

1.3 Обзор алгоритмов удаления невидимых ребер и поверхностей

Ниже рассмотрены возможные алгоритмы построения изображений.

Алгоритм Варнока

Данный алгоритм работает в пространстве изображений. В рамках этого алгоритма рассматривается окно и решается пусто оно или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на фрагменты до тех пор, пока содержимое подокна не станет достаточно простым для визуализации или пока его размер не достигнет области в один пиксель [2].

Особенностями этого алгоритма является возможность устранения лестничного эффекта и рекурсивный вызов.

Алгоритм, использующий Z-буфер

Данный алгоритм работает в пространстве изображений, используя буфер кадра для заполнения интенсивности каждого пикселя, здесь вводится некоторый Z-буфер (отдельный буфер глубины каждого пикселя). В процессе работы глубина каждого нового пикселя, который нужно занести в буфер кадра, сравнивается с глубиной пикселя, занесенного в Z-буфер. Ес-

ли сравнение показывает, что новый пиксель расположен ближе к наблюдателю, то новое значение z заносится в буфер и корректируется значение интенсивности [2].

Данный алгоритм прост в реализации, но его недостатком является большой объем требуемой памяти.

Более формальное описание алгоритма Z-буфера [2].

1) Заполнить буфер кадра фоновым значением цвета (интенсивности).
2) Заполнить Z-буфер минимальным значением.
3) Преобразовать каждый многоугольник сцены в растровую форму.
4) Для каждого пикселя(x, y) в многоугольнике вычислить глубину $z(x, y)$.
5) Сравнить глубину $z(x, y)$ со значением Z-буфер(x, y).
6) Если $z(x, y) > Z$=буфер(x, y), то записать цвет (интенсивность) в буфер кадра и заменить значение в Z-буфере.

Алгоритм Робертса

Данный алгоритм работает в объектном пространстве. Требуется, чтобы все изображаемые тела были выпуклыми, иначе они должны быть разбиты на выпуклые части. Тело должно представляться набором пересекающихся плоскостей. Основные этапы алгоритма:

1) подготовка исходных данных;
2) удаление линий, которые экранируются самим телом;
3) удаление линий, которые экранируются другими телами;
4) удаление тех линий пересечения тел, которые экранируются непосредственно самими телами, связанных отношением протыкания

Особенностями этого алгоритма является высокая точность вычислений, но из-за трудоемкости этих вычислений он не может являться достаточно эффективным.

Алгоритм прямой трассировки лучей

Главная идея этого метода заключается в том, что наблюдатель видит любой объект посредством испускаемого неким источником света, который падает на этот объект и затем каким-то путем доходит до наблюдателя. Свет может достичь наблюдателя, отразившись от поверхности, преломившись или пройдя через нее. Если проследить за лучами света, выпущенным источником, то можно убедиться, что немного из них дойдут до наблюдателя. Данный процесс вычислительно неэффективен [2]. В связи с этим он

будет исключен из рассмотрения.

Алгоритм обратной трассировки лучей

Данный алгоритм работает в пространстве изображения. В его основе 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 [2].

Считается, что камера (наблюдатель) находится на положительной части оси Oz, а картинная плоскость ей перпендикулярна. Тогда основные этапы алгоритма будут следующими:

1) в каждый пиксель растра выпускаются лучи из положения наблюдателя;
2) для каждого луча отслеживается траектория для определения ближайшего пересечения с объектом;
3) для определения цвета пикселя из найденной точки пересечения испускаются лучи к каждому источнику света. Если на пути к источнику света луч пересекает другой объект, то свет от данного источника света не учитывается в расчете цвета. Если для луча от наблюдателя не найдено объектов пересечения, то пиксель закрашивается цветом фона;
4) для расчета преломлений и отражений используются физические законы, такие как равенство угла падения и отражения и закон Снеллиуса. Найденная точка пересечения становится новой точкой наблюдения, и алгоритм испускания лучений повторяется до достижения максимальной глубины рекурсии.

На рисунке 1.1 представлена визуализация алгоритма обратной трассировки лучей.

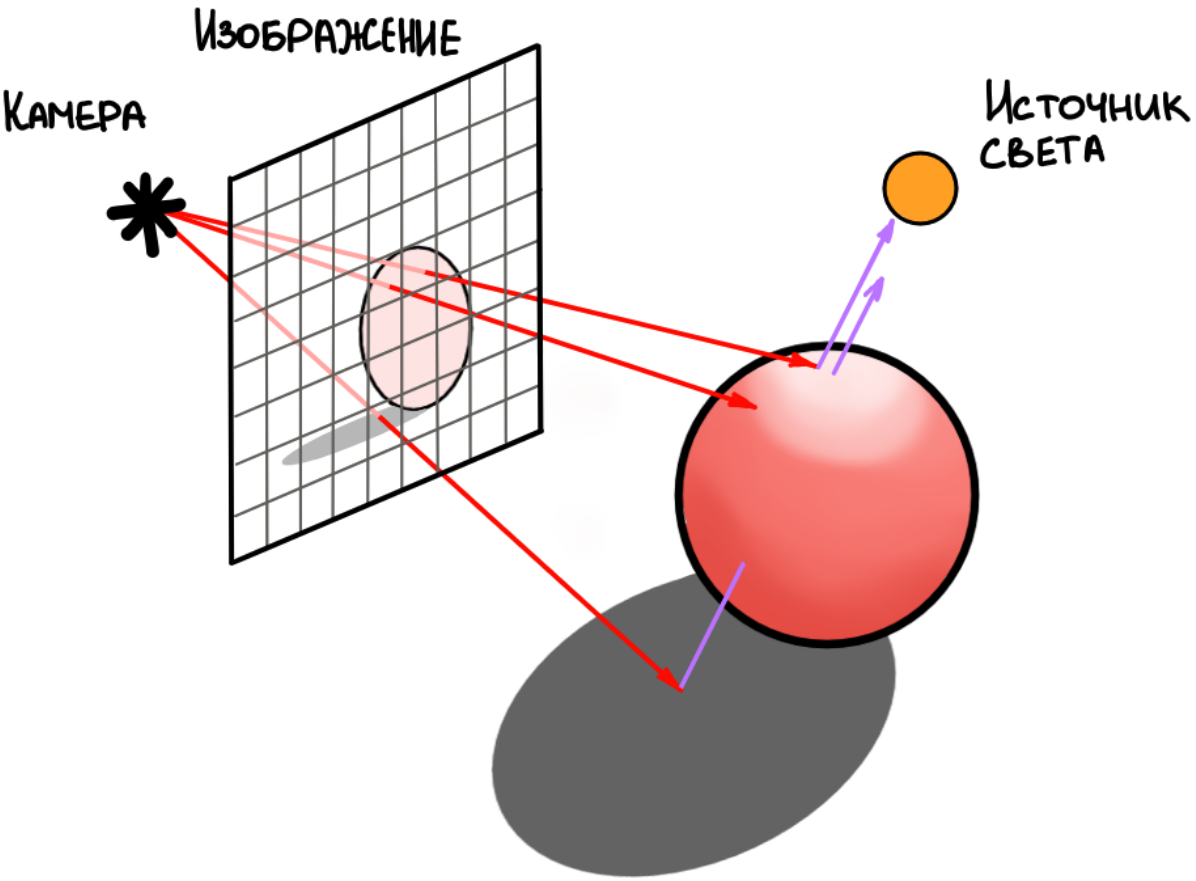

Рисунок 1.1 – Визуализация алгоритма обратной трассировки

Особенностями данного алгоритма является устранение ступенчатости, а также учет эффектов отражения одного объекта от поверхности другого, преломления, прозрачности и затемнения [2].

1.4 Обзор моделей освещения

Модели освещения разделяются на глобальные и локальные.

Локальная модель освещения является самой простой, она рассматривает расчет освещенности самих объектов, не учитывая процессы светового взаимодействия между объектами, кроме этого взаимодействие ограничивается однократным отражением света от непрозрачной поверхности.

Глобальная модель освещения рассматривает трехмерную сцену как единую систему, описывая освещение с учетом взаимного влияния объектов, а так же учитывая многократное отражение и преломление света.

Модель освещения Ламберта

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 [2].

Интенсивность освещенности в этой модели находится по следующему закону:

$$I_\alpha = I_0 \cdot cos(\alpha), \tag{1}$$

где I_α – уровень освещенности в рассматриваемой точке, I_0 – максимальный уровень освещенности, α – угол между вектором нормали к плоскости и вектором, направленным от рассматриваемой точки к источнику света.

Данная модель одна из самых простых моделей освещения, она часто используется в комбинации других моделей, т.к. практически в любой другой модели освещения можно выделить диффузную составляющую.

Модель освещения Фонга

Модель Фонга (глобальная модель) – классическая модель освещения, которая представляет собой комбинацию диффузной составляющей (модели Ламберта) и зеркальной составляющей и работает таким образом, что кроме равномерного освещения на материале может еще появляться блик. Местонахождение блика на объекте, освещенном по модели Фонга, определяется из закона равенства углов падения и отражения [1]. Ниже представлено более подробное описание составляющих модели.

1) *Ambient* (фоновое освещение) — константа, которая присутствует в любом участке сцены и не зависит от пространственных координат освещаемой точки и источника.
2) $Diffuse$ (рассеянный свет) — рассчитывается по закону косинусов (закону Ламберта):

$$I_d = k_d \cdot cos(L, N) \cdot i_d = k_d \cdot (L \cdot N) \cdot i_d, \tag{2}$$

где I_d – рассеянная составляющая освещенности в точке, k_d – свойство материала воспринимать рассеянное освещение, i_d – мощность рассеянного освещения, L – направление из точки на источник, N – вектор нормали в точке.

3) *Specular* (зеркальная составляющая) – зависит от того, насколько близко находится вектор отраженного луча и вектор к наблюдателю. Падающий, отраженный лучи и нормально к отражающей поверхности лежат в одной плоскости. Нормаль делит угол между лучами

на две равные части. Получается, что отраженная составляющая зависит от угла между направлением на наблюдателя и отраженным лучом, что отражено в формуле (3).

$$I_s = k_s \cdot cos^{\alpha}(R, V) \cdot i_s = k_s \cdot (R \cdot V)^{\alpha} \cdot i_s, \quad (3)$$

где I_s – зеркальная составляющая освещенности в точке, k_s – коэффициент зеркального отражение, i_s – мощность зеркального освещения, R – направление отраженного лука, V – направление на наблюдателя, α – коэффициент блеска, свойство материала.

Тогда, объединяя формулы (2) и (3) и учитывая наличие на сцене других тел, получатся следующее:

$$I = k_{\alpha} \cdot I_{\alpha} + k_d \cdot \sum_j I_j (N \cdot L_j) + k_s \cdot \sum_j I_j (V \cdot R_j)^{\alpha} + k_s \cdot I_s + k_r \cdot I_r, \quad (4)$$

где

- k_{α} – коэффициент фонового освещения,
- k_d – коэффициент диффузного отражения,
- k_s – коэффициент зеркального отражения,
- k_r – коэффициент пропускания,
- N – единичный вектор нормали к поверхности,
- L_j – единичный вектор, направленный к j-му источнику света,
- V – единичный вектор, направленный на наблюдателя,
- R_j – вектор направления отраженного луча L_j,
- I_{α} – интенсивность фонового освещения,
- I_j – интенсивность j-го источника света,
- I_s – интенсивность от зеркально отраженного луча,
- I_r – интенсивность от преломленного луча.

На рисунке 1.2 изображен пример работы модели Фонга.

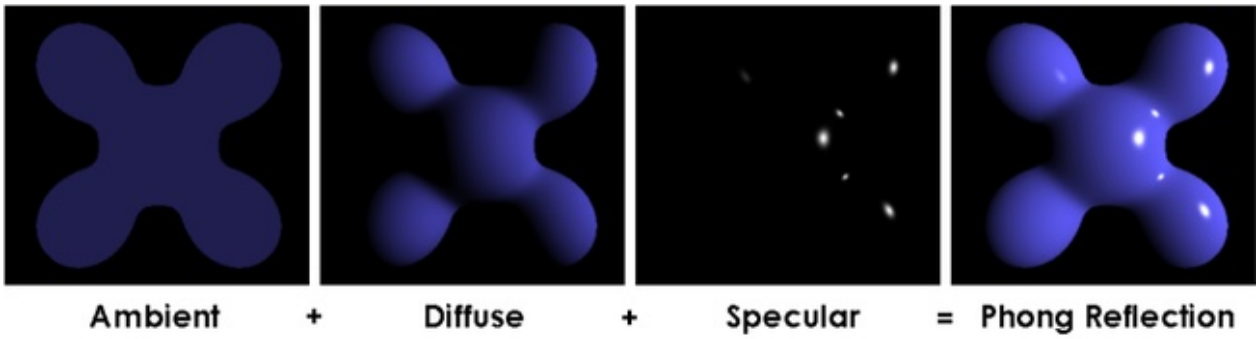

Рисунок 1.2 – Пример работы модели Фонга [3]

1.5 Задача учета прозрачности объекта

В основных моделях освещения и алгоритмах удаления невидимых линий и поверхностей рассматриваются только непрозрачные поверхности и объекты. Однако есть и прозрачные объекты, пропускающие свет.

При переходе из одной среды в другую световой луч преломляется. Преломление рассчитывается по закону Снеллиуса:

$$\eta_1 \cdot sin(\theta) = \eta_2 \cdot sin(\theta') \tag{5}$$

где η_1 и η_2 – показатели преломления двух сред, θ – угол падения, θ' – угол преломления.

Ни одно вещество не пропускает весь падающий свет, его часть всегда отражается, что видно на рисунке 1.3

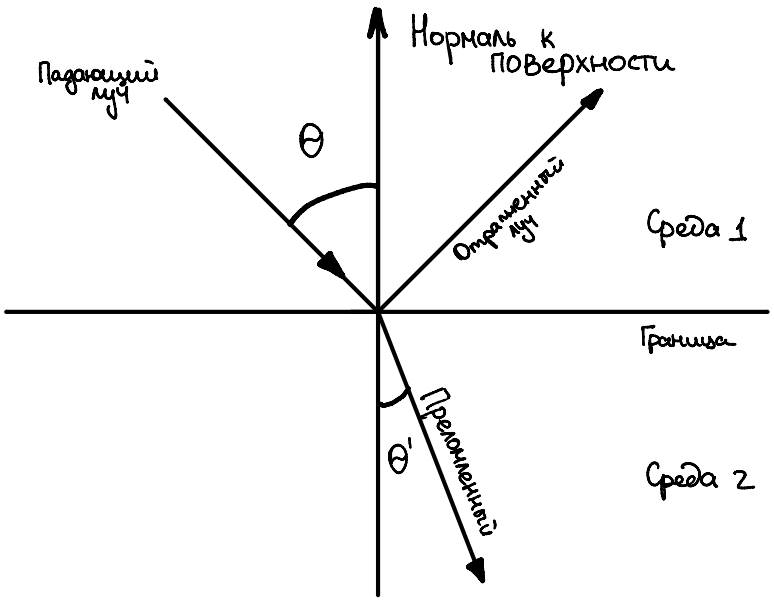

Рисунок 1.3 – Геометрия преломления

Простое преломление пропускание света можно встроить в любой алгоритм удаления невидимых поверхностей, кроме алгоритма с Z-буфером. Прозрачные многоугольники или поверхности поменяются и, если видимая грань прозрачна, то в буфер кадра записывается линейная комбинация двух ближайших поверхностей. При этом интенсивность будет равна:

$$I = t \cdot I_1 + (1 - t) \cdot I_2 \qquad 0 \leq t \leq 1, \tag{6}$$

где I_1 – видимая поверхность, I_2 – поверхность, расположенная непосредственно за ней, t – коэффициент прозрачности I_1 [2].

При $t = 0$ поверхность абсолютно прозрачна, при $t = 1$ непрозрачна.

Для того чтобы включить преломление в модель освещения, нужно при построении видимых поверхностей учитывать не только падающий, но и отраженный и пропущенный свет. Эффективнее всего это выполняется с помощью глобальной модели освещения в сочетании с алгоритмом трассировки лучей [2].

Сравнение рассмотренных алгоритмов

В таблице 1.1 представлено сравнение алгоритмов, описанных ранее. Представленная в таблице оценка эффективности по времени была взята из [1] и [2] источников.

Таблица 1.1 – Сравнение алгоритмов удаления ребер и невидимых поверхностей

Характеристика	Алгоритм или класс алгоритмов			
	В	ЗБ	Р	ОТ
Теоретическая оценка эффективности по времени	$O(w \cdot h \cdot n)$	$O(w \cdot h \cdot n)$	$O(n^2)$	$O(w \cdot h \cdot n)$
Возможность построения сложных сцен	Да	Да	Нет	Да
Учет всех оптических эффектов	Нет	Нет	Нет	Да
Наличие сложных объемных вычислений	Нет	Нет	Да	Нет

В данной таблице были приняты следующие обозначения: «В» – раз-

личные варианты алгоритма Варнока, «ЗБ» – алгоритмы, использующие Z-буфер, «Р» – алгоритм Робертса, «ОТ» – алгоритм обратной трассировки лучей, w и h – ширина и высота изображения растра, соответственно, n – количество объектов на сцене. Было принято решение использовать алгоритм обратной трассировки лучей, так как именно он удовлетворяет всем поставленным задачам.

Вывод

В данном разделе была выполнена формализация объектов сцены и приведены существующие методы и алгоритмы для решения поставленной задачи. Для задания трехмерной модели была выбрана твердотельная модель, а для реализации синтеза изображения — алгоритм обратной трассировки лучей.

2 Конструкторская часть

В данном разделе описываются математические основы метода моделирования и разрабатываются схемы алгоритмов на основе теоретических данных из аналитического раздела.

2.1 Математические основы для реализации алгоритма обратной трассировки лучей

2.1.1 Определение направлений отраженного и преломленного лучей

На рисунке 2.1 представлены направления отраженного и преломленного лучей.

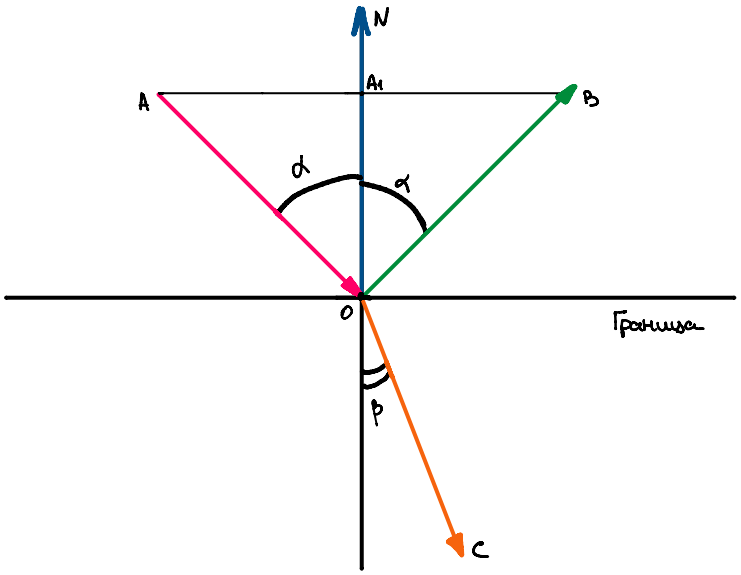

Рисунок 2.1 – Направления лучей

$\overrightarrow{OA_1}$ – проекция $\overrightarrow{AO}$. Тогда, $AO = ON(ON \cdot (-AO))$ и $A_1A = -AO - OA_1 = -AO + ON(ON \cdot AO)$. Тогда $\overrightarrow{OB}$):

$$OB = -OA - 2ON(ON \cdot AO). \tag{7}$$

Пусть η_1 и η_2 показатели преломления сред 1 и 2, соответственно. При-

меняя закон Снеллиуса можно вычислить преломленный луч:

$$OC = \frac{\eta_1}{\eta_2} AO + (\frac{\eta_1}{\eta_2} cos(\alpha_1) - cos(\alpha_2))ON, \tag{8}$$

где $cos(\alpha_2) = \sqrt{1 - (\frac{\eta_1}{\eta_2})^2 (1 - cos(\alpha_1))^2}$.

2.1.2 Расчет нормали и коэффициентов плоскости по точкам

Уравнение вида

$$Ax + By + Cz + D = 0 \tag{9}$$

называют общим уравнением плоскости [4].

Если есть три точки, не лежащие на одной прямой, тогда существует единственная плоскость, которой эти точки принадлежат. Пусть $M(x, y, z)$ – произвольная точка на плоскости, а точки $M_i(x_i, y_i, z_i), i = 1, 2, 3$ нам известны. Тогда для описания плоскости как множества точек M векторы $\overrightarrow{M_1M_2}, \overrightarrow{M_1M_3}, \overrightarrow{M_1M}$ должны быть компланарны. Записав критерий компланарности векторов, получается следующий вид:

$$\begin{vmatrix} x - x_1 & x_2 - x_1 & x_3 - x_1 \\ y - y_1 & y_2 - y_1 & y_3 - y_1 \\ z - z_1 & z_2 - z_1 & z_3 - z_1 \end{vmatrix} = 0. \tag{10}$$

Тогда из формулы (10) будут получены следующие коэффициенты плоскости:

$$A = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1), \tag{11}$$

$$B = (x_2 - x_1)(z_3 - z_1) - (x_3 - x_1)(z_2 - z_1), \tag{12}$$

$$C = (x_2 = x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1), \tag{13}$$

$$D = -Ax_1 + By_1 - Cz_1. \tag{14}$$

Если после подстановки полученных коэффициентов получается положительное значение, то надо умножить их на -1. Вектор нормали к данному полигону можно представить как $n = \begin{pmatrix} A & B & C \end{pmatrix}^T$.

2.1.3 Поиск пересечения луча со сферой

Луч определяется своим положением p и единичным вектором направления u. Тогда, луч [5] — это множество точек

$$\{p + \alpha u | \alpha \geq 0\}. \tag{15}$$

Сфера задается точкой центра c и радиусом $r > 0$. Прежде всего находится точка q – ближайшая к центру сферы на луче p, следующим образом:

$$q = p + \alpha u, \tag{16}$$

где α – расстояние от q до p. На рисунке 2.2 представлены все обозначения для наглядности.

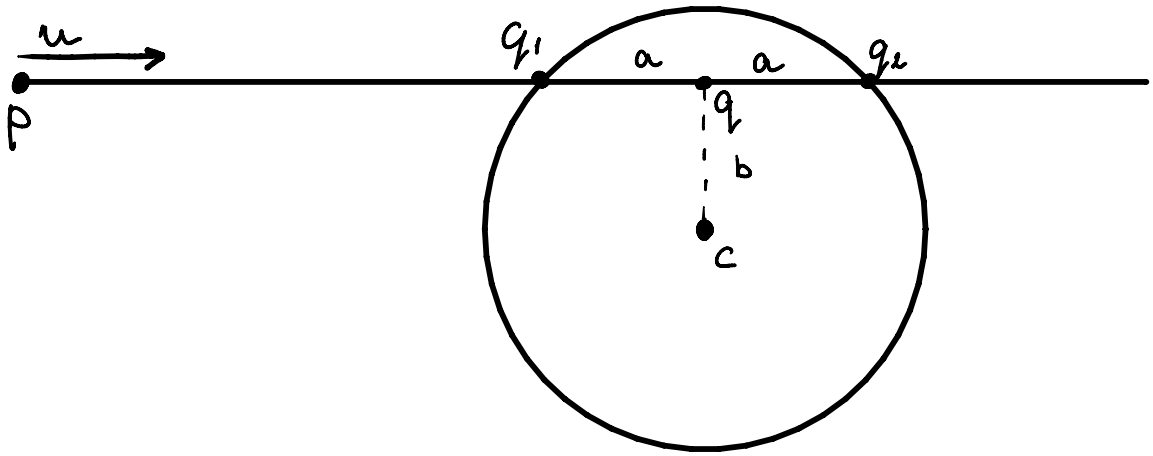

Рисунок 2.2 – Геометрические построения

Из построения получается:

$$(q - c) \perp u \Rightarrow 0 = (q - c) \cdot u = (p + \alpha u)u. \tag{17}$$

Т.к. $u \cdot u = 1$, из правой части формулы (17) можно выразить α:

$$\alpha = -(p - c) \cdot u. \tag{18}$$

Подставляя (18) в (16) можно выразить точку q:

$$q = p - ((p - c) \cdot u) \cdot u. \tag{19}$$

Когда точка q найдена, можно проверить пересекает луч сферу или нет:

$$|q - c|^2 \leq r^2. \tag{20}$$

Если q лежит внутри сферы, то обозначим за $b = |q - c|$ и $a = \sqrt{r^2 - b^2}$. В общем случае луч пересекает сферу в двух точках:

$$q_{1,2} = p + (\alpha \pm a) \cdot u. \tag{21}$$

Если $\alpha \geq a$, то луч пересекается со сферой в точке q_1. Иначе, необходимо проверить $\alpha + a > 0$. Если это неравенство выполняется, то точка лежит внутри сферы, а точка q_2 – точка падения луча на внутреннюю поверхность сферы [5].

2.2 Пересечение луча с многогранником

Есть ситуации, когда можно сразу понять, что многогранник не пересекается лучом, для этого необходимо поместить многогранник в сферическую оболочку и проверить пересечение со сферой по описанному выше алгоритму. В ином случае необходимо осуществить поиск пересечений со всеми гранями многогранника.

Пусть для поиска пересечения луча с гранью многогранника плоскость Г задается вектором нормали n и числом d. Точки x плоскости удовлетворяют уравнению $x \cdot n = d$ [5]. Луч задан точкой положения p и единичным вектором направления u. Пусть

$$q = p + \alpha u, \alpha \in \mathbb{R}. \tag{22}$$

Тогда

$$q \in \Gamma \Leftrightarrow d = q \cdot n = p \cdot n + \alpha u \cdot n. \tag{23}$$

Из формулы (23) получается следующее:

$$\alpha = \frac{d - p \cdot n}{u \cdot n}. \tag{24}$$

Так как плоскость разбивает пространство на два полупространства, пусть полупространство, куда направлена нормаль n будет называться «над» плоскостью, а вторая часть – «под» плоскостью.

Тогда,

$$d - p \cdot n \begin{cases} < 0, & \text{точка } p \text{ находится «над» плоскостью,} \\ > 0, & \text{точка } p \text{ находится «под» плоскостью,} \\ = 0, & \text{луч параллелен плоскости.} \end{cases} \tag{25}$$

Расстояние между p и q обозначается как α. Если $\alpha < 0$, то луч не пересекает плоскость.

2.3 Проектирование алгоритма обратной трассировки лучей

На рисунке 2.3 представлена схема работы алгоритма обратной трассировки лучей.

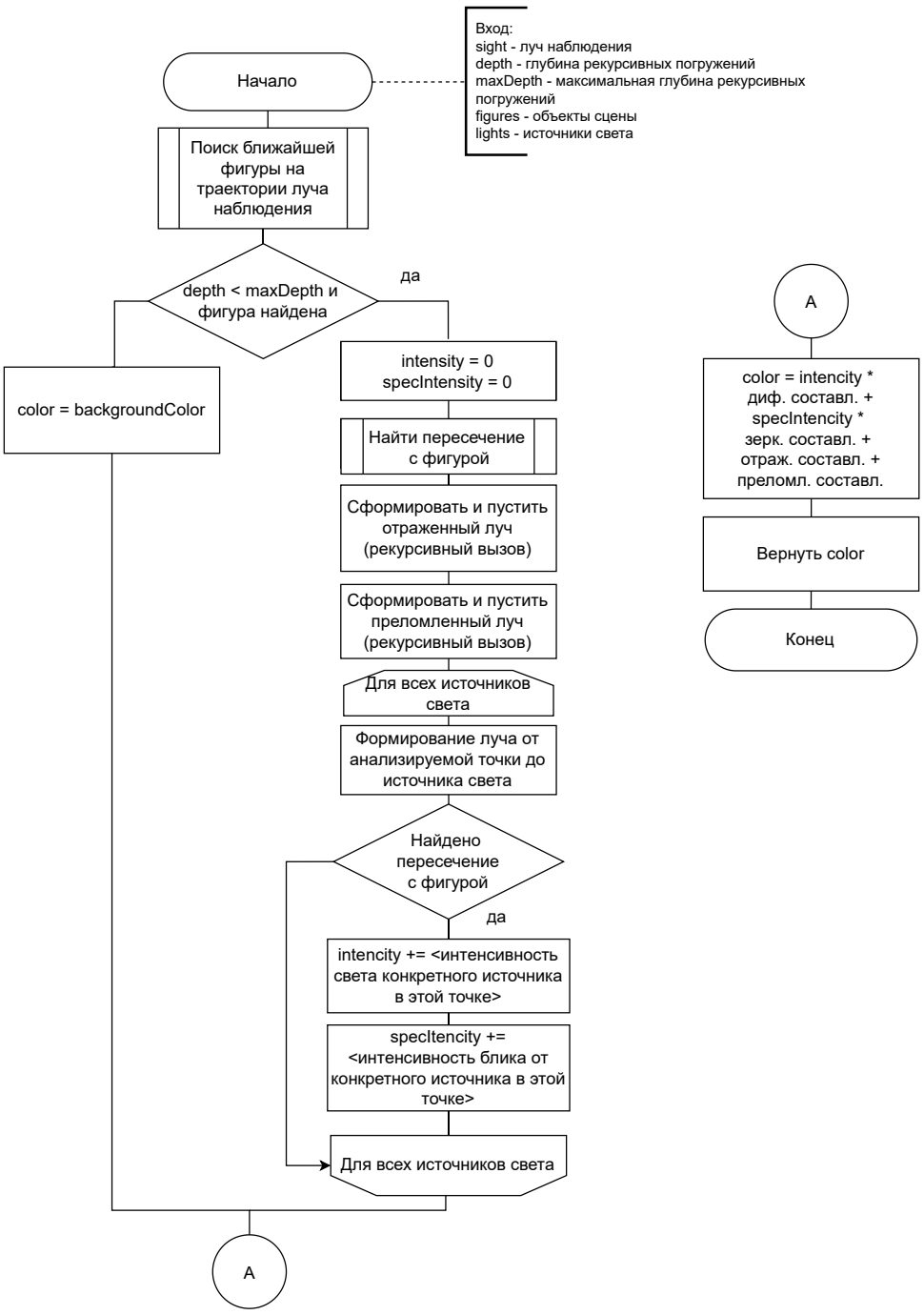

Рисунок 2.3 – Схема работы алгоритма обратной трассировки лучей

Выводы

В данном разделе были описаны математические основы метода моделирования и разработаны схемы алгоритмов на основе теоретических данных из аналитического раздела.

3 Технологическая часть

В данном разделе обосновывается выбор средств реализации и проводится тестирование.

3.1 Выбор технологии программирования

Структурный и объектно-ориентированный подходы являются основными технологиями в программировании. Так как данная задача представляет собой сложную систему, было принято решение об использовании объектно-ориентированного подхода. Кроме того, в выбранном подходе есть принципы инкапсуляции, полиморфизма и наследования, которые позволяют создавать более гибкие и широко расширяемые системы в отличие от структурного подхода.

3.2 Средства реализации

При написании программного продукта было решено задействовать язык C++ [6]. Этот выбор обусловлен следующими факторами:

1) высокая вычислительная производительность, которая позволяет сократить время синтеза изображения;
2) поддержка объектно-ориентированного программирования, что позволяет пользоваться шаблонами и паттернами проектирования;
3) доступность – существует большое количество учебной литературы и статей по C++ на различных языках;
4) по сравнению с C, в C++ больше готового функционала.

Средой разработки была выбрана среда QtCreator [7], что обусловлено следующими причинами:

1) поддержка расширения QtDisign, которая позволяет создавать интерфейс;
2) отладчик, позволяющий выявлять ошибки в коде программного продукта;
3) хорошая совместимость с C++, так как фреймворк Qt написан на этом языке программирования.

3.3 Примеры работы программы

На рисунке 3.1 представлен интерфейс ПО.

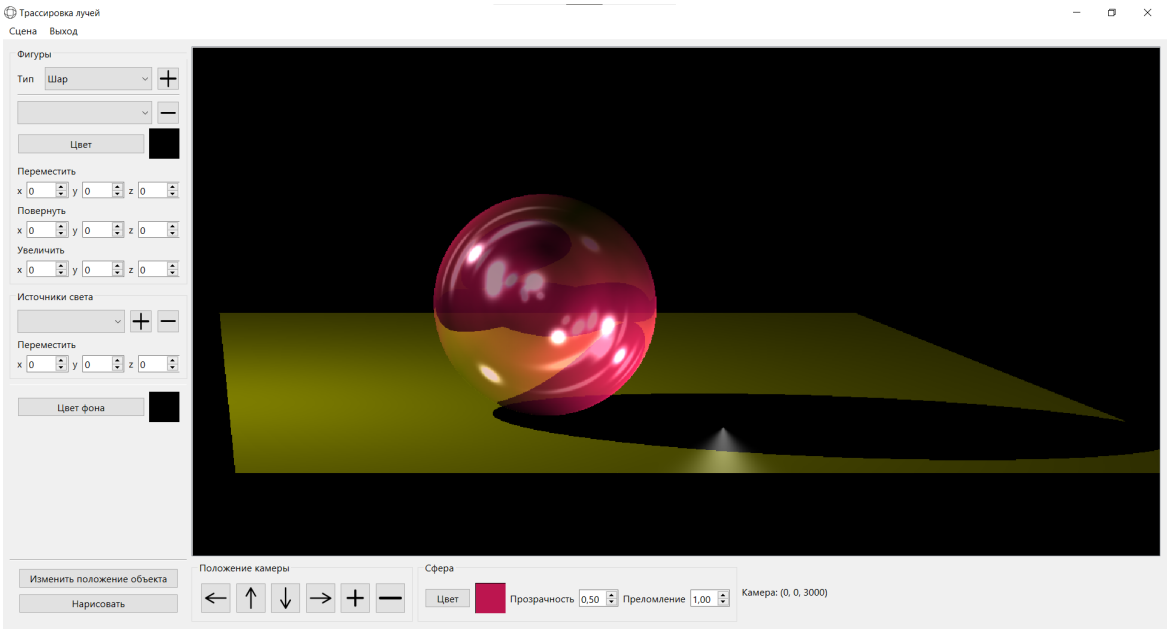

Рисунок 3.1 – Интерфейс ПО

Выводы

В данном разделе были реализованы алгоритмы линейного и параллельного конвейеров и выполнено тестирование этих реализаций.

4 Экспериментальная часть

В данном разделе проводится сравнительный анализ реализованных алгоритмов по затрачиваемому реальному времени.

4.1 Технические характеристики

Технические характеристики устройства, на котором проводился эксперимент:

- операционная система Windows 10 x64 Домашняя;
- оперативная память 15 Гб;
- процессор Intel(R) Core(TM) i5-10300H CPU @ 2.50 ГГц с 8 логическими ядрам [8].

4.2 Замеры реального времени

Для тестирования брались квадратные изображения размером от 128×128 пикселей до 512×512 с шагом 128.

Класс std::chrono::system_clock [9] был использован для замеров затрачиваемого реализациями алгоритмов реального времени.

Для построения графиков использовалась библиотека matplotlib для языка программирования Python [10].

В таблице 4.1 представлен результат замеров времени. На рисунке 4.1 приведен график, отражающий зависимость времени выполнения реализаций алгоритмов от количества потоков и линейного размера квадратного изображения.

Таблица 4.1 – Время выполнения реализации алгоритмов при разном количестве потоков (в мс.)

Количество потоков	Линейный размер изображения			
	128	256	384	512
0	50.441	215.668	546.182	827.216
1	52.3	211.573	466.08	836.741
2	41.559	140.251	316.52	557.305
4	26.982	119.199	266.242	436.737
8	19.972	75.809	170.347	310.915
16	14.552	51.035	157.271	229.747
32	13.533	53.764	113.155	192.211
64	14.922	47.623	184.554	184.554
128	20.28	55.28	104.377	187.823

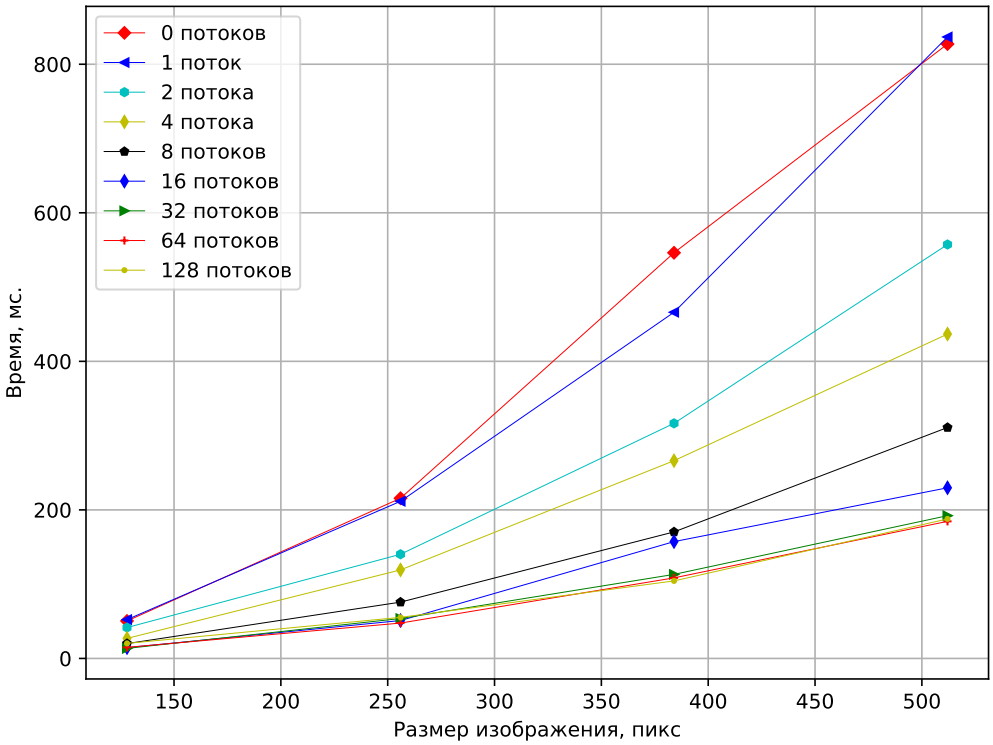

Рисунок 4.1 – Зависимость времени работы реализаций алгоритмов от количества потоков и размера изображения

Выводы

В данном разделе были проведены замеры времени работы реализаций всех трех алгоритмов. Из них можно сделать следующие выводы.

Наибольшая эффективность достигается при 64 потоках. Например, изображение в 512×512 пикселей генерируется быстрее на 64 потоках в 1.7 раз, чем на 8 потоках.

При увеличении размера изображения росло время рендера. Так, при 8 потоках изображение размером 128×128 синтезируется быстрее изображения размером 256×256 в 3.8 раз, так как необходимо обработать больше пикселей.

Заключение

В ходе выполнения курсовой работы поставленная цель была достигнута: было разработано программное обеспечение визуализации геометрических объектов, расположенных за полупрозрачной сферой. Были выполнены все задачи:

— формализованы объекты сцены;
— выбраны и описаны алгоритмы реализации, необходимые для визуализации поставленной задачи;
— составлены требования к программному продукту;
— реализованы выбранные алгоритмы;
— исследовано время работы реализации алгоритма от количества потоков.

Список использованных источников

1. Куров А.В. Курс лекций по компьютерной графике. — 2022.

2. Д. Роджерс, Дж. Адамс. «Математические основы машинной графики» / Дж. Адамс Д. Роджерс. — М.: – Мир, 2001. — С. 603.

3. Zahidi, Usman A. A Radiative Transfer Model-Based Multi-Layered Regression Learning to Estimate Shadow Map in Hyperspectral Images / Usman A. Zahidi, Ayan Chatterjee, Peter W. T. Yuen // *Machine Learning and Knowledge Extraction*. — 2019. — Aug. — Vol. 1, no. 3. — P. 904–927. http://dx.doi.org/10.3390/make1030052.

4. А. Н. Канатников, А. П. Крищенко. Аналитическая геометрия: Учебник для вузов / А. П. Крищенко А. Н. Канатников. — 9-е изд. — М.: Издательство МГТУ им. Н.Э. Баумана, 2019. — С. 376.

5. Ярошевич В.А. Компьютерная графика : практические занятия, лабораторный практикум [Электронный ресурс]. — Режим доступа: http://miet.aha.ru/cg/cg_2015.pdf (дата обращения: 27.10.2022).

6. Документация языка C++ [Электронный ресурс]. — Режим доступа: https://docs.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170 (дата обращения: 27.11.2022).

7. Документация библиотеки Qt [Электронный ресурс]. — Режим доступа: https://doc.qt.io (дата обращения: 27.11.2022).

8. Процессор Intel® Core™ i5-10300H. [Электронный ресурс]. — Режим доступа: https://ark.intel.com/content/www/ru/ru/ark/products/201839/intel-core-i510300h-processor-8m-cache-up-to-4-50-ghz.html (дата обращения: 27.09.2022).

9. Структура system_clock. [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/cpp/standard-library/system-clock-structure?view=msvc-170 (дата обращения: 10.11.2022).

10. Matplotlib: Visualization with Python [Электронный ресурс]. — Режим доступа: `https://matplotlib.org/` (дата обращения: 27.09.2022).