Государственное образовательное учреждение высшего профессионального образования

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

Построение реалистичного изображения методом трассировки лучей

Студент  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Головнева М. И.

(Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Волкова Л. Л.

(Подпись, дата) (И.О.Фамилия)

Москва, 2018

**Оглавление**

[Введение 4](#_Toc3301955)

[1. Аналитическая часть 5](#_Toc3301956)

[1.1. Методы визуализации изображения 5](#_Toc3301957)

[1.2. Алгоритм прямой трассировки лучей 6](#_Toc3301958)

[1.3. Алгоритм обратной трассировки лучей 8](#_Toc3301959)

[2. Конструкторская часть 10](#_Toc3301960)

[2.1. Описание объектов сцены 10](#_Toc3301961)

[2.2. Разработка алгоритма 11](#_Toc3301962)

[2.3. Поиск пересечения с объектом сцены 13](#_Toc3301963)

[2.3.1. Пересечение со сферой 13](#_Toc3301964)

[2.3.2. Пересечение с полигоном 14](#_Toc3301965)

[2.3.3. Пересечение с цилиндром 15](#_Toc3301966)

[2.4. Вычисление нормалей 15](#_Toc3301967)

[2.4.1. Нормаль к сфере 15](#_Toc3301968)

[2.4.2. Нормаль к полигону 16](#_Toc3301969)

[2.4.3. Нормаль к цилиндру 16](#_Toc3301970)

[2.5. Поиск отраженного луча 17](#_Toc3301971)

[2.6. Поиск преломленного луча 18](#_Toc3301972)

[2.7. Вычисление локального цвета 19](#_Toc3301973)

[3. Технологическая часть 21](#_Toc3301974)

[3.1. Выбор среды разработки и языка программирования 21](#_Toc3301975)

[3.2. Основные классы и структуры данных программы 22](#_Toc3301976)

[3.3. Интерфейс программы 22](#_Toc3301977)

[4. Исследовательская часть 24](#_Toc3301978)

[4.1. Постановка эксперимента 24](#_Toc3301979)

[4.2. Сравнительный анализ на основе экспериментальных данных 24](#_Toc3301980)

[Заключение 28](#_Toc3301981)

[Список литературы 29](#_Toc3301982)

# Введение

В современном обществе роль компьютерных технологий очень велика. Компьютеры буквально пронизывают все сферы деятельности современного человека. Самая важная функция компьютера – обработка информации. Особо можно выделить обработку информации, связанную с изображениями. Компьютерная графика – это область деятельности, в которой компьютеры используются для синтеза изображений и для обработки визуальной информации. С момента начала развития компьютерной графики появилась необходимость в построении реалистичного изображения. Проблема построения реалистичного изображения остается актуальной и по сей день, поскольку является важнейшей областью компьютерной графики и, несмотря на увеличение компьютерных мощностей, построение трехмерного изображения, близкого к реальному, всё ещё остаётся достаточно сложной задачей. Построить такое изображение можно используя различные алгоритмы: метод закраски Гуро, метод закраски Фонга, алгоритм z-буфера, трассировка лучей и так далее.

Целью курсового проекта является изучение и реализация одного из самых популярных методов построения реалистичного изображения – метода трассировки лучей.

Для достижения поставленной цели требуется решить следующие задачи:

1. провести обзор методов построения реалистичного изображения трехмерной сцены;
2. разработать программное обеспечение для моделирования трехмерной сцены, включающей геометрические тела – сферы, кубы, цилиндры;
3. разработать пользовательский интерфейс, предоставляющий возможность изменять положение камеры в пространстве сцены.

# Аналитическая часть

В данном разделе будут рассмотрены методы визуализации изображения и описаны алгоритмы обратной и прямой трассировки лучей.

* 1. **Методы визуализации изображения**

Визуализация изображения — термин в компьютерной графике, обозначающий процесс получения изображения по модели с помощью компьютерной программы. Существует несколько различных методов визуализации трехмерного изображения, и далее будут коротко рассмотрены наиболее часто применяемые [4].

Каркасный метод визуализации (wireframe rendering) не предназначен для создания реалистичных изображений (пример представлен на рис. 1). В каркасных моделях выводятся только края граней объектов сцены, изображаемые прямыми линиями.

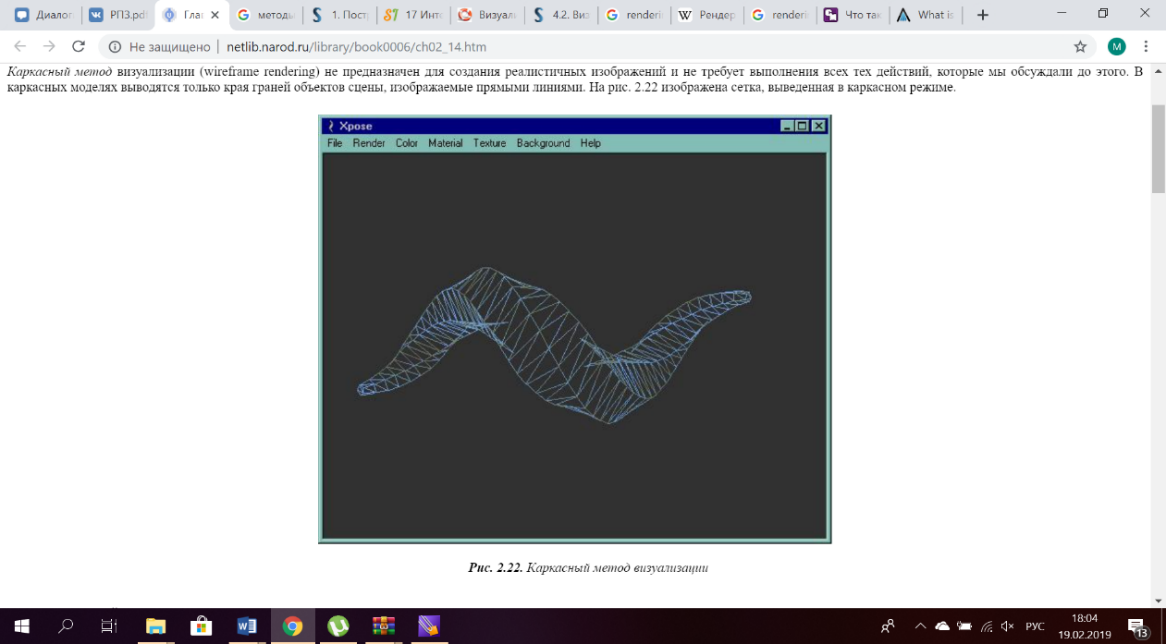


Рисунок 1 – Пример каркасной визуализации изображения

Неосвещенный метод визуализации (unlit rendering) - получил свое название из-за того, что при его применении наличие источников света игнорируется (пример представлен на рис. 2). Грани рисуются с использованием назначенных им цветов и текстур без учета значений освещенности или ориентации грани.

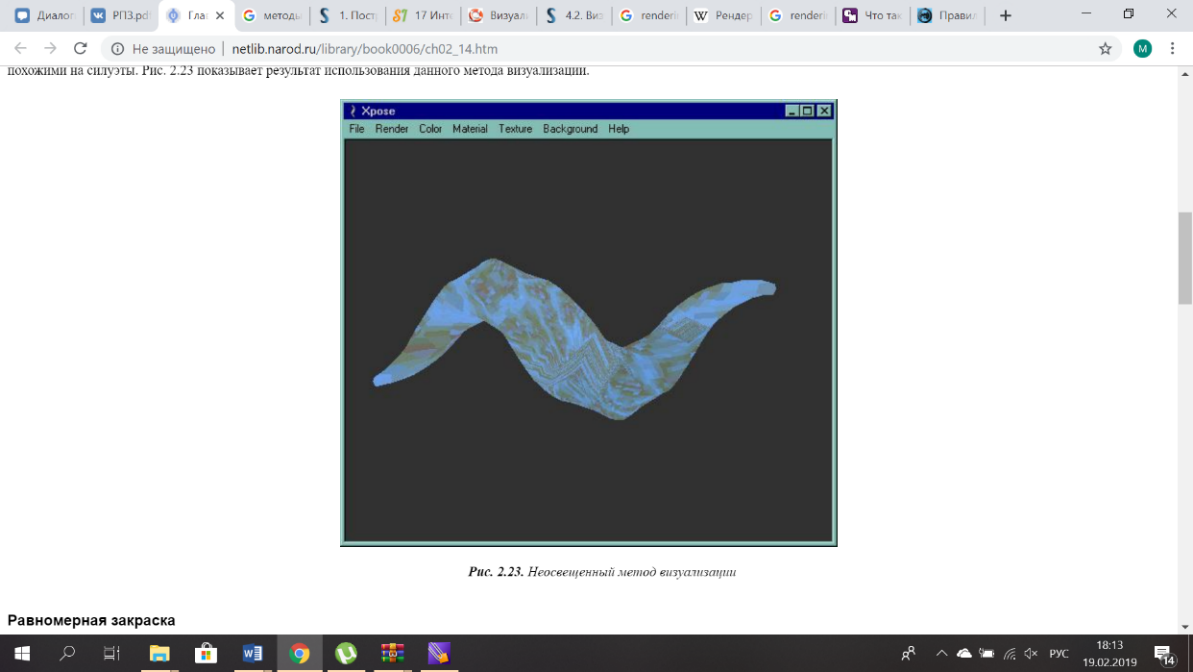


Рисунок 2 – Пример неосвещенного метода визуализации

Закраски (shadings) - позволяют получить более реалистичное изображение, чем каркасный или неосвещенный методы визуализации (пример представлен на рис. 3). При закрасках учитывается освещенность грани. Сюда относятся: равномерная закраска, закраска Гуро, закраска Фонга.

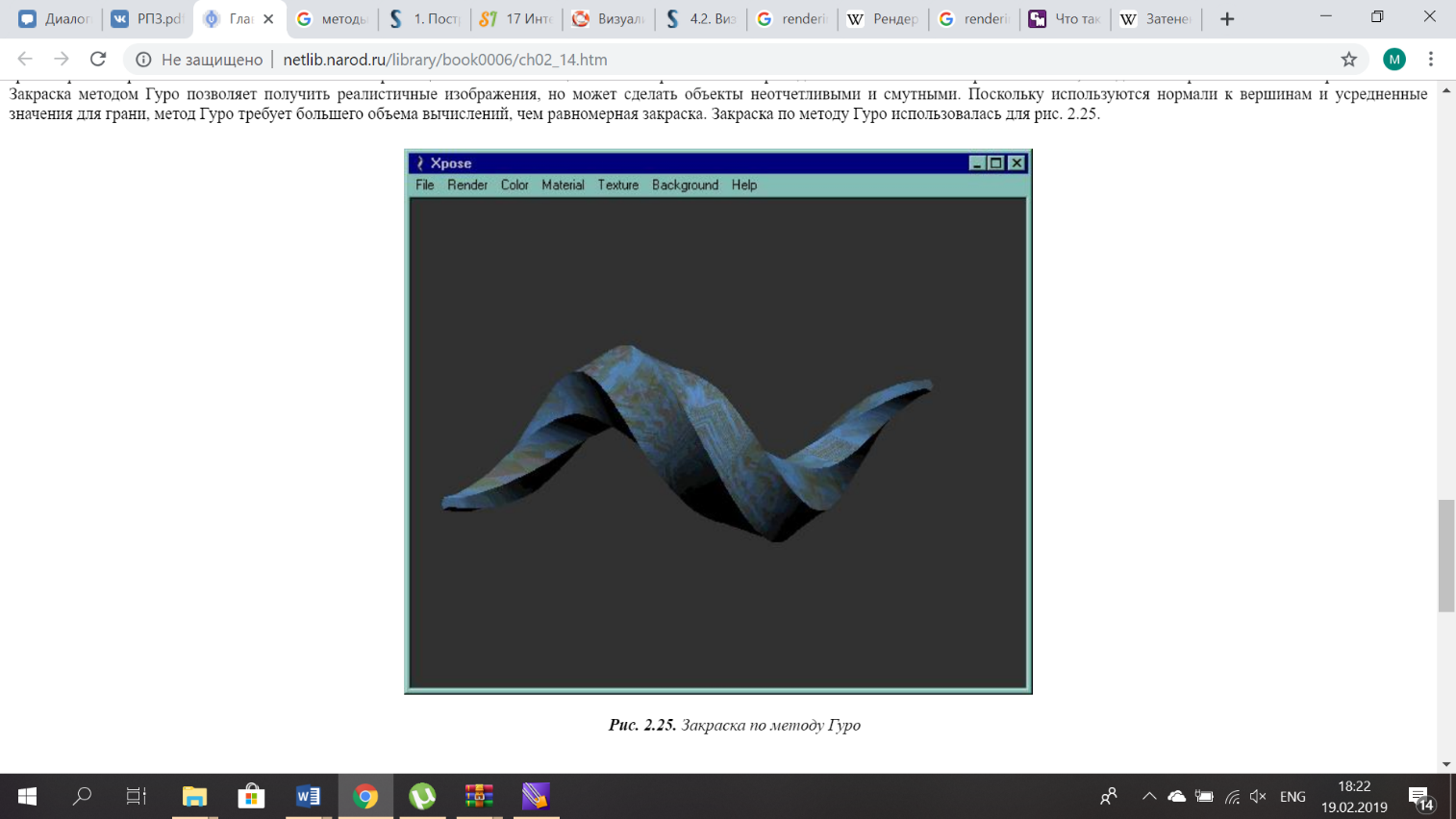


Рисунок 3 – Пример закраски по методу Гуро

Трассировка лучей (ray-tracing) - позволяет получить наиболее реалистичное изображение по сравнению с любым другим методом визуализации (пример представлен на рис. 4). Трассировка лучей известна как способ получения фотореалистичных и даже гиперреалистичных изображений. Алгоритм трассировки лучей автоматически вычисляет тени, отражения и преломления.

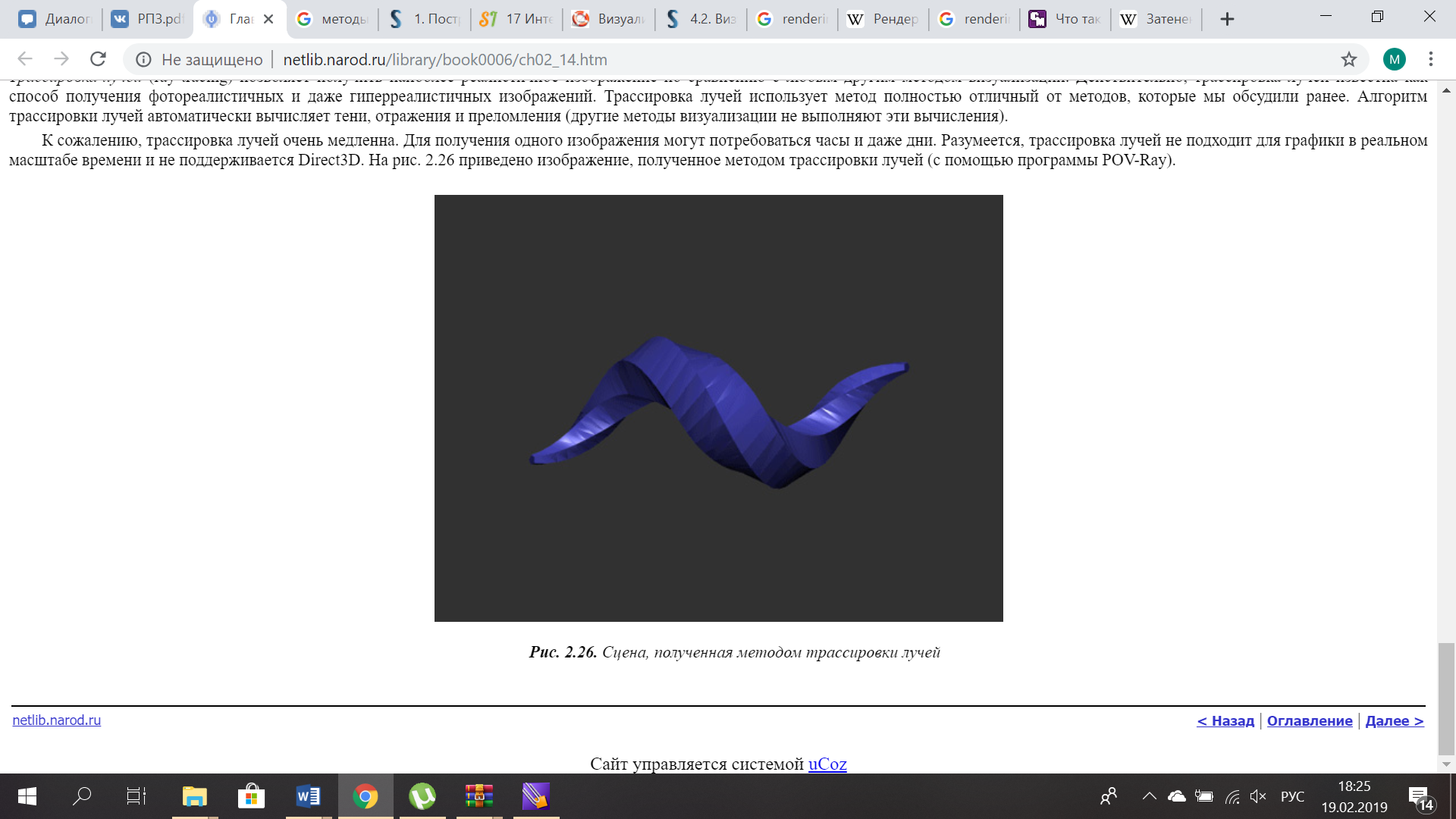


Рисунок 4 – Пример трассировки лучей

* 1. **Алгоритм прямой трассировки лучей**

Алгоритм прямой трассировки лучей основан на реальном взаимодействии света с физическими объектами.

От каждого источника освещения исходит бесчисленное множество лучей во всех направлениях. Назовем их первичными лучами. Некоторые лучи уходят в свободное пространство, а некоторые попадают на другие объекты. Попав на объект, первичные лучи начинают с ним взаимодействовать, отражаться и преломляться, в следствии чего возникают вторичные лучи. Так же, как и в случае с первичными лучами, некоторые из них попадают на другие объекты. Так, многократно отражаясь и преломляясь, отдельные световые лучи приходят в точку наблюдения. Таким образом, формируется изображение сцены. Данный алгоритм демонстрирует нам рис. 5.

Цвет отдельных точек изображения определяется спектром и интенсивностью первичных лучей источников излучения, а так же поглощением световой энергии в объектах, встретившихся на пути соответствующих лучей.

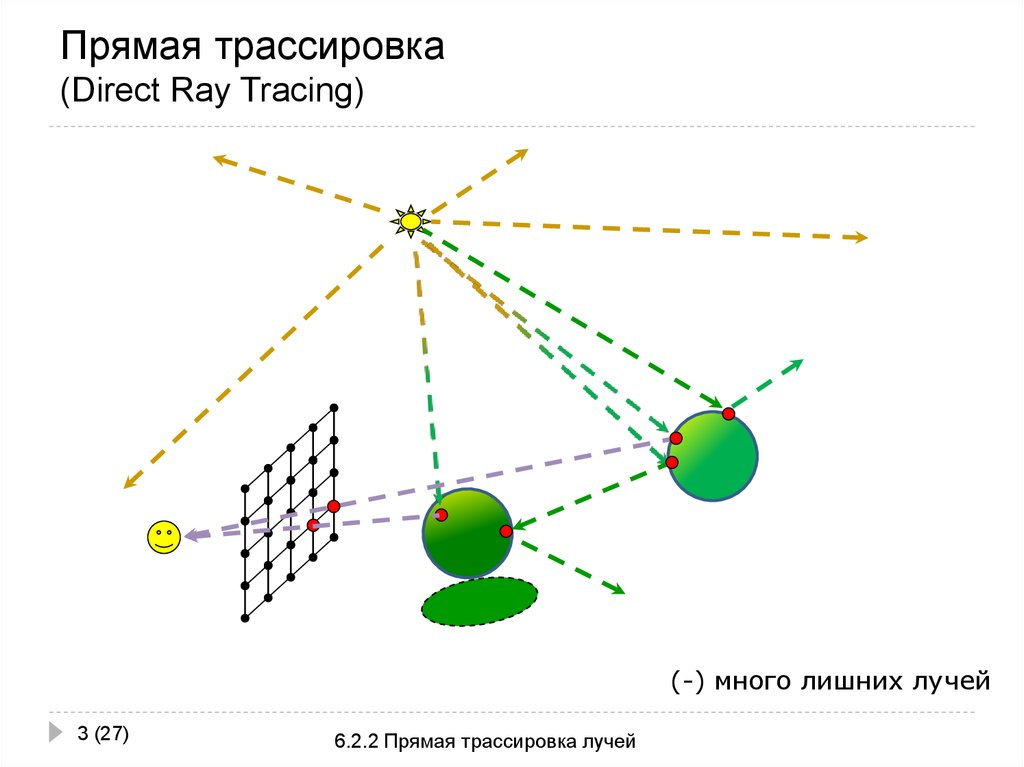


Рисунок 5 – Схема работы алгоритма прямой трассировки лучей

Непосредственная реализация данной модели формирования изображения представляется затруднительной. Начнем с того, что генерация бесконечного множества лучей от источника света уже представляется проблематичным. Даже если это осуществить, то в процессе работы алгоритма в приемник в конечном итоге попадает лишь малая часть лучей, испускаемых источником света. Таким образом, для построения изображения используется лишь малая часть наших вычислений, а большая часть работы алгоритма оказывается проделанной впустую.

Из-за трудоемкости и неэффективности алгоритма сейчас он практически нигде не применяется, а в различных источниках, упоминая трассировку лучей, по умолчанию имеют в виду обратную трассировку [1].

* 1. **Алгоритм обратной трассировки лучей**

Метод обратной трассировки лучей позволяет значительно сократить перебор световых лучей. Согласно этому методу отслеживание лучей производится не от источников света, а в обратном направлении – от точки наблюдения (рис. 6). Так учитываются только те лучи, которые вносят вклад в формирование изображения.

Из точки наблюдения проводится прямая линия через середину пиксела плоскости проецирования. Это будет первичный луч обратной трассировки. Если этот луч попадет в один или несколько объектов сцены, то выбираем ближайшую точку пересечения. Для определения цвета пиксела изображения нужно учитывать свойства объекта, а также то, какое световое излучение приходится на соответствующую точку объекта.

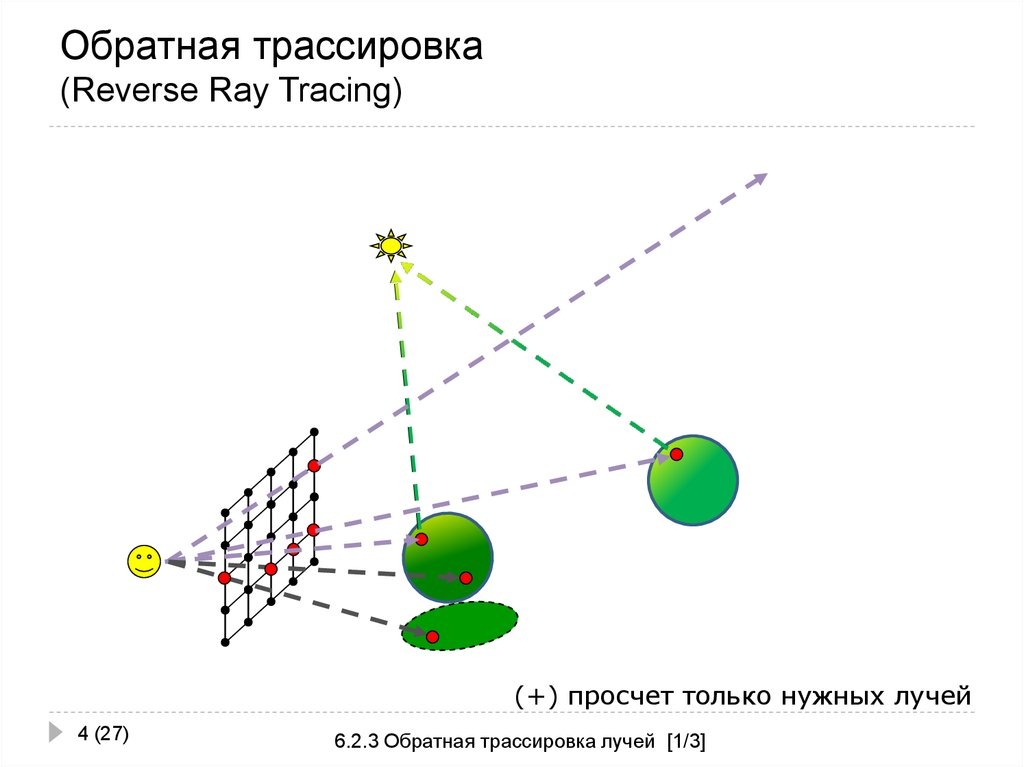


Рисунок 6 – Схема работы алгоритма обратной трассировки лучей

Если объект зеркальный (хотя бы частично), то строим вторичный луч – луч падения, считая лучом отражения предыдущий, первичный трассируемый луч. Для идеального зеркала достаточно затем проследить лишь очередную точку пересечения вторичного луча с некоторым объектом. У идеального зеркала идеально ровная отполированная поверхность, поэтому отраженному лучу соответствует падающий луч. Зеркало может быть затемненным, то есть поглощать часть световой энергии, но все равно остается правило: один луч падает – один отражается.

Если объект прозрачный, то необходимо построить новый луч, такой, который при преломлении давал бы предыдущий трассируемый луч.

Для диффузного отражения интенсивность отраженного света, как известно, пропорциональна косинусу угла между вектором луча от источника света и нормалью.

Когда выясняется, что текущий луч обратной трассировки не пересекает какой-либо объект, а уходит в свободное пространство, то на этом трассировка для этого луча заканчивается.

# Конструкторская часть

В данном разделе приводится описание объектов сцены, а также разработан алгоритм для выполнения поставленной задачи. В качестве алгоритма для построения реалистичного изображения был выбран алгоритм обратной трассировки лучей. Этот метод является наиболее универсальным, так как позволяет выполнять удаление невидимых поверхностей, наложение теней, отражение, преломление. Так же он значительно сокращает перебор лучей, необходимых при прямой трассировке. Реализация прямой трассировки лучей будет описана в технологической части.

* 1. **Описание объектов сцены**

Сцена состоит из геометрических тел и источников освещения.

В работе рассматриваются следующие геометрические тела:

* сфера;
* куб;
* цилиндр.

Каждое геометрическое тело имеет свой цвет, и обладает характеристиками, определяющими их способность к отражению/преломлению света. Сфера задаётся с помощью центра и радиуса. Куб состоит из 12 полигонов, каждый из которых задаётся тремя точками в пространстве. Цилиндр задается с помощью центра, радиуса основания и высоты.

Было решено выбрать модель освещения, описанную в статье [3]. В связи с этим, при построении сцены используются три вида источников света:

* точечный источник;
* направленный источник;
* окружающее освещение.

Точечный источник испускает свет из фиксированной точки в пространстве, называемой его позицией. Свет испускается равномерно во всех направлениях, именно поэтому его так же называют всенаправленным освещением. Следовательно, точечный источник полностью характеризуется его позицией и яркостью.

Направленный источник является аппроксимацией солнечного света. Как и точечные источники, направленный источник имеет яркость, но в отличие от них, у него нет позиции. Вместо неё у него есть направление. Можно воспринимать его как бесконечно удалённый точечный источник, светящий в определённом направлении.

В реальном мире, даже точки, не получающие свет непосредственно от источника освещения, не находятся полностью в темноте. Когда свет падает на объект, часть его поглощается, но остальная часть рассеивается в сцене. Это значит, что свет может поступать не только от источников освещения, но и от других объектов, а значит нужно считать источником освещения каждый объект. Как можно представить, это сильно увеличивает сложность нашей модели, поэтому в данной работе это явление не учитывается. Тогда возникает проблема в том, что если объект не освещен напрямую, он является полностью темным. Эта проблема решается с помощью третьего типа источников освещения – окружающего освещения. Окружающее освещение характеризуется только яркостью. Считается, что оно носит безусловный вклад освещения в каждую точку сцены. Это очень сильное упрощение чрезвычайно сложного взаимодействия между источниками освещения и поверхностями сцены.

* 1. **Разработка алгоритма**

В данной работе алгоритм обратной трассировки лучей реализован рекуррентным образом. Функция трассировки лучей, возвращающая цвет пикселя, рекуррентно вызывает саму себя для нахождения отраженного цвета и цвета света, проходящего сквозь объект. Схема алгоритма трассировки представлена на рис. 7.

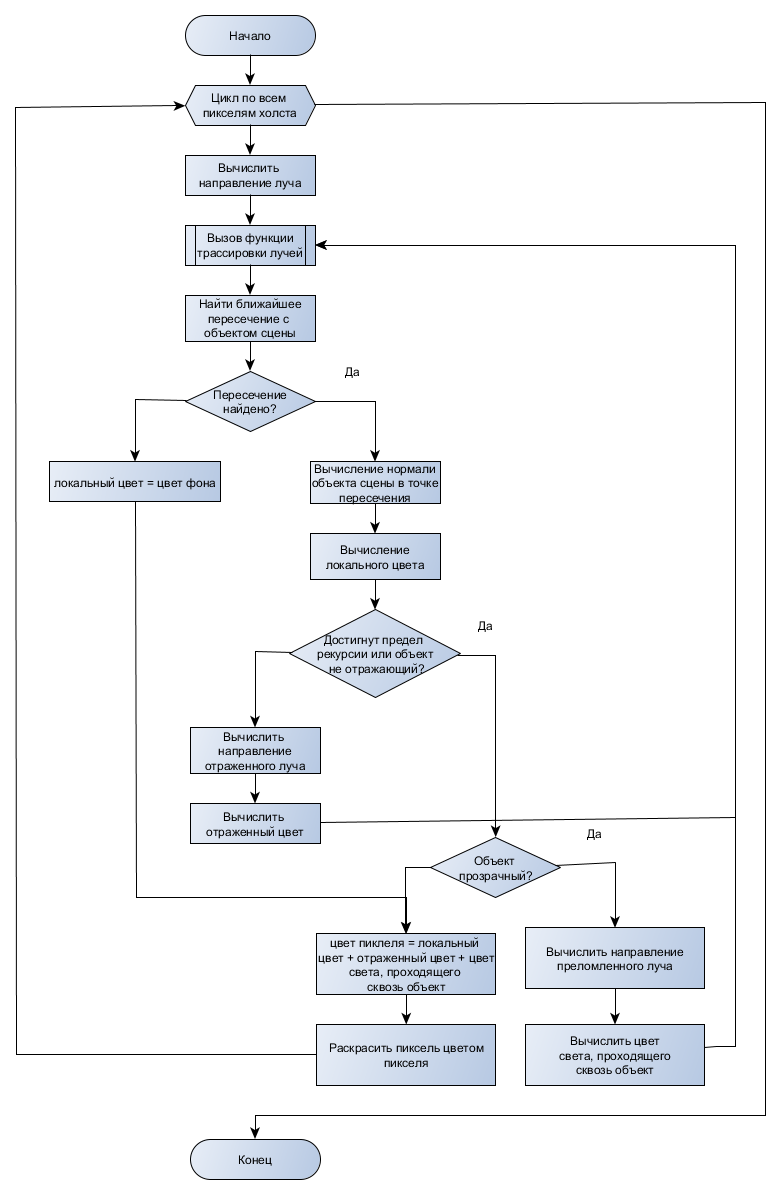
****

Рисунок 7 – Схема алгоритма обратной трассировки лучей

Цвет пикселя складывается из локального цвета, отраженного цвета и цвета, проходящего сквозь объект.

В соответствии с выбранной моделью освещения [3], каждый объект должен обладать тремя коэффициентами: specular, reflective и transparent. Коэффициент specular определяет, будет ли объект более матовым или блестящим, reflective задаёт способность тела к отражению, a transparent – устанавливает насколько прозрачным будет объект.

Локальный цвет зависит от диффузного отражения, от того, является объект матовым или блестящим, и находится ли он в тени.

* 1. **Поиск пересечения с объектом сцены**

Способ определения точки пересечения зависит от того, какой это объект, и каким образом он представлен.

Любая точка луча удовлетворяет уравнению:

R = A + Vt, (1)

где R - радиус-вектор произвольной точки, принадлежащей лучу, A – радиус-вектор начальной точки луча, V – вектор направления луча, t – параметр.

Если направляющий вектор V нормализовать, то параметр t будет численно равен расстоянию от начальной точки луча A до точки R.

Можно записать это уравнение в координатном виде:

,

, (2)

.

Здесь – координаты начальной точки луча в прямоугольной декартовой мировой системе координат, a, b, c – координаты направляющего вектора луча.

* + 1. **Пересечение со сферой**

Для нахождения пересечения со сферой, воспользуемся уравнением сферы:

. (3)

Вместо x, y, z в уравнение (3) нужно подставить соответствующие уравнения луча (2). После раскрытия всех скобок и приведения подобных мы получим квадратное уравнение относительно параметра t:

Если дискриминант квадратного уравнения меньше нуля, то луч и сфера точек пересечения не имеют. В противном случае можно будет вычислить два значения параметра t. Дискриминант может быть равен нулю – это соответствует предельному случаю касания луча поверхности, и мы получим два совпадающих значения параметра t.

В программе при нахождении двух пересечений для визуализации выбирается ближнее из них. Ближнее пересечение определяется путем сравнения найденных параметров t. Ближе к точке наблюдения находится то пересечение, которому соответствует меньший параметр t.

Для нахождения координат точек пересечения луча и поверхности достаточно подставить найденные значения параметра t в уравнение (1).

* + 1. **Пересечение с полигоном**

Для нахождения пересечения луча с полигоном сначала найдём пересечение луча с плоскостью, в которой лежит полигон, а затем определим находится ли точка внутри треугольника.

Запишем уравнение плоскости:

Q(x, y, z) = Ax + By + Cz + D = 0. (4)

Коэффициенты A, B, C совпадают с координатами нормали к этой плоскости. Вычислить нормаль к плоскости можно с помощью векторного произведения двух сторон треугольника.

Выразим свободный член D:

D = –Ax – By – Cz.

Теперь для нахождения свободного члена D достаточно подставить координаты любой точки, принадлежащей треугольнику. В программе подставляется одна из вершин.

Теперь для нахождения точки пересечения подставим уравнение (2) в уравнение (4):

.

Следовательно:

Если знаменатель дроби равен нулю, значит луч параллелен плоскости, в которой лежит треугольник, следовательно, точки пересечения нет.

Далее, необходимо определить, попала ли точка пересечения M внутрь треугольника.

Рассмотрим треугольник ABC. Определим три векторных произведения:

na = AB \* AM,

nb = BC \* BM,

nc= CA \* CM.

Эти вектора будут коллинеарны. Если все три вектора сонаправлены, то точка M лежит внутри треугольника. Сонаправленность определяется равенством знаков соответствующих координат всех трех векторов.

* + 1. **Пересечение с цилиндром**

Для нахождения пересечения с цилиндром, воспользуемся уравнением цилиндра:

Подставим уравнения (2) в уравнение цилиндра (5). После раскрытия всех скобок и приведения подобных, получим квадратное уравнение относительно параметра t:

Если дискриминант квадратного уравнения меньше нуля, то луч и цилиндр точек пересечения не имеют. Если дискриминант равен нулю, то луч касается цилиндра в одной точке. Если дискриминант больше нуля, то можно вычислить две точки пересечения. Как и в случае со сферой, выбирается ближняя из них. Так как цилиндр не бесконечный и задается в программе высотой, найденная точка проходит проверку на то, что она находится внутри цилиндра.

* 1. **Вычисление нормалей**

Для реализации алгоритма обратной трассировки лучей необходимо вычисление нормалей к каждому виду геометрического тела.

* + 1. **Нормаль к сфере**

Для вычисления нормали к точке на сфере необходимо вычесть координаты центра сферы из координаты точки поиска.

где N – вектор нормали, P – точка, к которой надо найти нормаль, O – центр сферы.

Полученный вектор необходимо нормализовать, поделив его на радиус сферы.

* + 1. **Нормаль к полигону**

Вычисление нормали к полигону сводится к операции векторного умножения. Пусть заданы координаты всех вершин треугольника ABC.

Вычислим координаты векторов AB и BC:

Координаты вектора нормали будут вычисляться по формулам:

Полученный вектор необходимо нормализовать, разделив его на его длину.

* + 1. **Нормаль к цилиндру**

Точка P, лежащая на боковой поверхности цилиндра, лежит на окружности, плоскость которой параллельна основаниям цилиндра, радиус равен радиусу цилиндра, а центр О находится на образующей. Координаты вектора нормали N в точке Р равны разности координат точек P и О.

Полученный вектор необходимо нормализовать, разделив его на длину вектора.

* 1. **Поиск отраженного луча**

Для реализации отражения требуется найти направление отраженного луча. Падающий луч, отражённый и нормаль к поверхности в точке касания лежат в одной плоскости, угол падения равен углу отражения.

Будем считать, что задан радиус-вектор S, направленный на источник света, а также известен радиус-вектор нормали N. Требуется найти вектор отражённого луча. Обозначим его R.

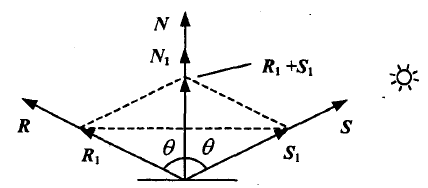


Рисунок 8 – Отражение луча

Рассмотрим единичные вектора R1 и N1. Поскольку векторы нормали, падающего луча и отражённого луча находятся в одной плоскости, то можно записать R1 + S1 = N`, где N` – это вектор, соответствующий диагонали ромба и совпадающий по направлению с нормалью. Длина вектора N` равна 2 cos.

Так как вектор N` по направлению совпадает с N1, то N` = N12 cos или R1+S1=N12 cos.

Отсюда найдём единичный вектор отражённого луча:

R1 = N12 cos - S1= N/|N|2 cos-S/|S|.

Найдём cos, используя скалярное произведение векторов N и S:

cos = N\*S/(|N|\*|S|).

Подставим это значение в выражение для R1:

R1 = N\*2\*N\*S/(|N2|\*|S|) – S/|S|.

Полагая, что искомый вектор отражённого луча будет иметь такую же длину, что и вектор падающего луча, то есть R = |S|R1, получим

R = N\*2\*N\*S/|N2|-S.

Это решение в векторной форме. Запишем координаты вектора:

xR = 2xN(xNxS+yNyS+zNzS)/(xN2+yN2+zN2) – xS,

yR = 2yN(xNxS+yNyS+zNzS)/(xN2+yN2+zN2) – yS,

zR = 2zN(xNxS+yNyS+zNzS)/(xN2+yN2+zN2) – zS.

* 1. **Поиск преломленного луча**

Для определения направления преломлённого луча используется закон Снеллиуса: «Отношение синуса угла падения к синусу угла отражения равно отношению абсолютных коэффициентов преломления сред».

Пусть заданы два единичных вектора: S1 - это радиус-вектор, направленный на источник, N1 – радиус-вектор нормали к границе раздела двух сред. Так же должны быть известны два коэффициента преломления для данных сред n1 и n2.

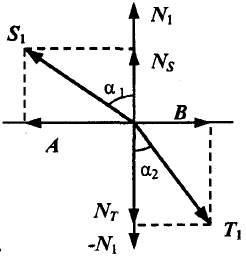


Рисунок 9 – Преломление луча

Требуется найти единичный радиус-вектор преломлённого луча . Искомый вектор равен сумме двух векторов: .

Найдём вначале вектор . Он противоположен по направлению вектору нормали, а его длина равна ||cos = cos (так как по условию задачи – единичный). Отсюда следует, что = -cos.

Необходимо найти cos. Запишем закон преломления sin = sin в виде:

sin = nsin, где n = /.

Воспользуемся тождеством cos2 + sin2 = 1.

Тогда:

cos или cos

Значение cos можно выразить через скалярное произведение единичных векторов и , то есть cos = . Тогда мы можем записать такое выражение для вектора :

Осталось найти выражение для вектора B. Он располагается на одной прямой с вектором A, причем A = – . Учитывая, что равен cos, то:

A = – cos.

Так как cos= , то

A = – ().

Поскольку длина вектора A равна sin, а длина вектора B равна sin, то

Откуда |B| = n|A|. Учитывая взаимное расположение векторов A и B, получим:

B = -nA = n((\*) -).

Теперь мы можем записать искомое выражение для единичного вектора луча преломления :

* 1. **Вычисление локального цвета**

Для вычисления освещённости точки требуется вычислить количество света, вносимое каждым источником и сложить их, чтобы получить одно число, представляющее общее количество полученного точкой освещения. Затем, цвет поверхности в этой точке умножается на это число, чтобы получился правильно освещённый цвет [3].

Cформулировать уравнение для вычисления полного количества света, полученного точкой P с нормалью  можно следующим образом:

где– яркость окружающего освещения, n  – количество точечных и направленных источников света,  – их яркость,  вектор направления светового луча.

Вектор  известен для направленного источника и вычисляется для точечного источника.

Члены, в которых  < 0 не должны прибавляться к освещённости точки.

Если объект является блестящим, то есть коэффициент specular не равен -1, то к значению прибавляется ещё одно значение:

где – яркость источника света, – направление отраженного луча, «вектор обзора», указывающий из точки P в камеру, s – значение коэффициента specular.

Члены, в которых < 0 не должны прибавляться к освещённости точки.

Учитывая значение можно смоделировать белые блики света на поверхности объекта.

Необходимо так же сделать проверку на то, что объект находится в тени. Если между точкой и источником есть объект, то не нужно добавлять освещение, поступающее от этого источника. Для этого можно снова воспользоваться алгоритмом для вычисления ближайшего пересечения луча с объектом. Однако вместо того, чтобы начинаться с камеры, лучи испускаются из P, а направление луча равно . В случае направленного источника нас интересуют пересечения со всем после P на бесконечное расстояние, в случае точечного источника - начиная с P, но только до источника света (в противном случае, объекты за источником освещения могли бы создавать тени).

# Технологическая часть

В данном разделе будет обоснован выбор среды разработки и языка программирования, а также описаны основные классы программы.

* 1. **Выбор среды разработки и языка программирования**

В настоящее время существует множество языков и сред программирования, многие из которых обладают достаточно высокой эффективностью, удобством и простотой в использовании. Выбор был сделан в пользу среды разработки Microsoft Visual Studio 2017 с использованием языка C#.

Язык C# был выбран по следующим причинам.

* Этот язык предоставляет программисту широкие возможности реализации самых разнообразных алгоритмов, обладая высокой эффективностью и значительным набором стандартных классов и процедур.
* С# является полностью объектно-ориентированным языком, позволяющим использовать множественное наследование, абстрактные и параметризованные классы и т.д. Кроме того, трехмерные объекты, так же как и математические абстракции, естественным образом представляются в виде объектов классов, что позволяет легко и эффективно организовывать их взаимодействие, при этом сохраняя читаемый и легко изменяемый код. Здесь же следует отметить такое немаловажное свойство объектно-ориентированного подхода, как повторное использование кода.

В качестве среды разработки была выбрана Microsoft Visual Studio .NET 2017 по следующим причинам.

* Программа изначально проектировалась для использования в среде Microsoft Visual Studio, поэтому компилятор C#, реализованный в этой среде, полностью поддерживает все особенности операционной системы.
* Данная среда обладает удобным редактором кода и отладчиком, предоставляющими большое количество функций, необходимых для эффективного написания исходного кода, локализации и устранения ошибок в программе.
  1. **Основные классы программы**

Рассмотрим классы, которые потребуются в программном обеспечении.

**class Figure** - родительский класс, который хранит цвет геометрического тела, а также значения коэффициентов specular (определяет, будет ли объект более матовым или блестящим), reflective (задаёт способность тела к отражению) и transparent (устанавливает насколько прозрачным будет объект). От данного класса наследуются class Sphere, class Triangle и class Cilinder.

**class Sphere** – является потомком class Figure и описывает сферу. Хранит координаты центра сферы и радиус.

**class Triangle** - является потомком class Figure и описывает полигон. Хранит координаты трех точек треугольника.

**class Cilinder** - является потомком class Figure и описывает цилиндр. Хранит координаты центра цилиндра, радиус его основания и высоту.

**сlass Light** - используется для описания источника освещения. Хранит тип источника (точечный источник, направленный источник, окружающее освещение), его интенсивность, а также позицию или направление, если они есть.

* 1. **Интерфейс программы**

Интерфейс программы представлен на рис. 9.

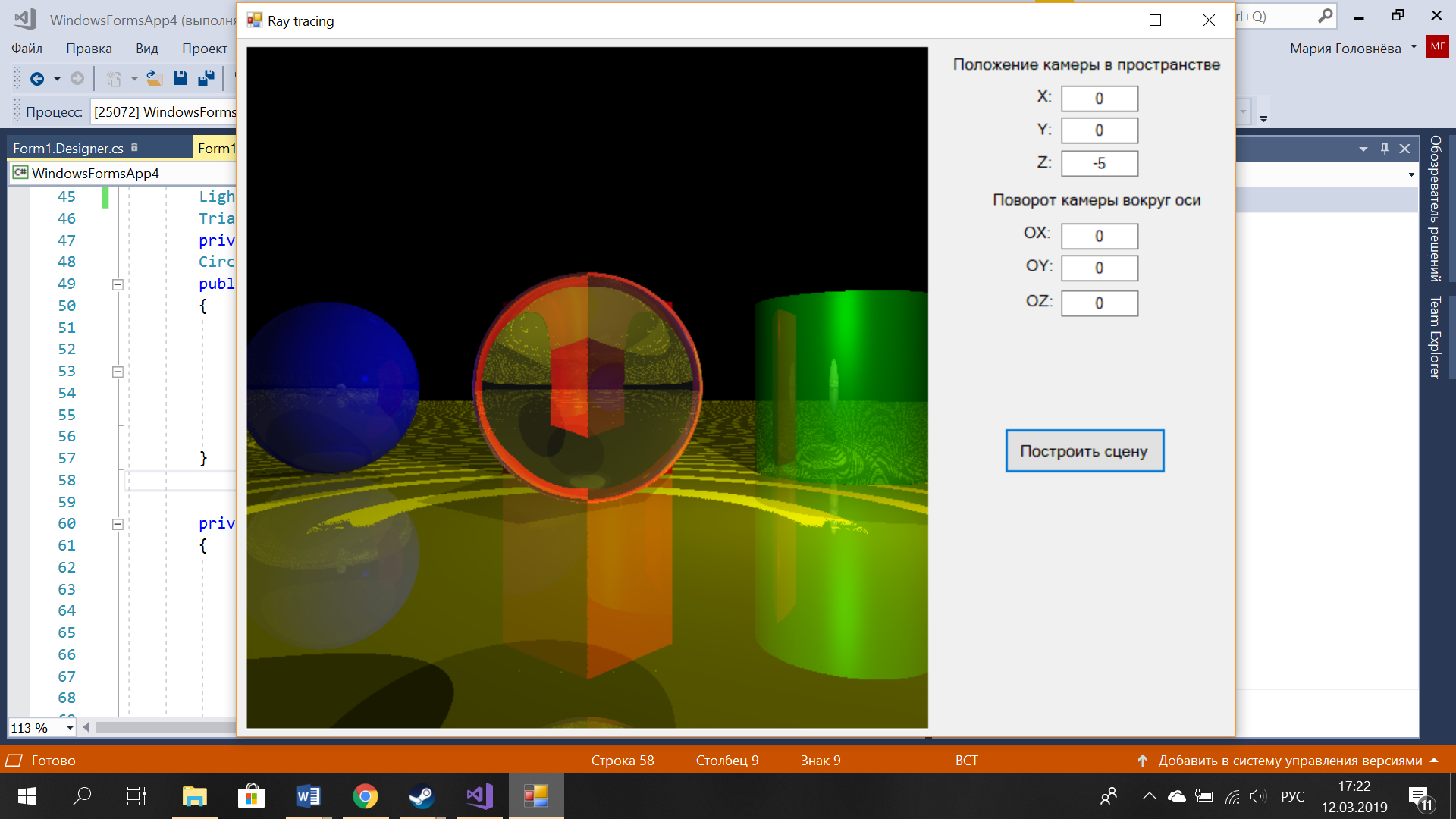


Рисунок 9 – Интерфейс программы

При запуске программы на экран выводится заданная сцена, координаты положения камеры в пространстве по умолчанию равны: {0, 0, -5}, поворот камеры вокруг какой-либо оси отсутствует.

Пользователю предоставляется возможность в соответствующих окнах ввести координаты положения камеры в пространстве, а также углы поворота вокруг каждой координатной оси (в градусах). После нажатия на кнопку «Построить сцену» сцена будет построена в соответствии со значениями, заданными пользователем.

Примеры работы программы можно увидеть на рис. 10 и рис. 11.

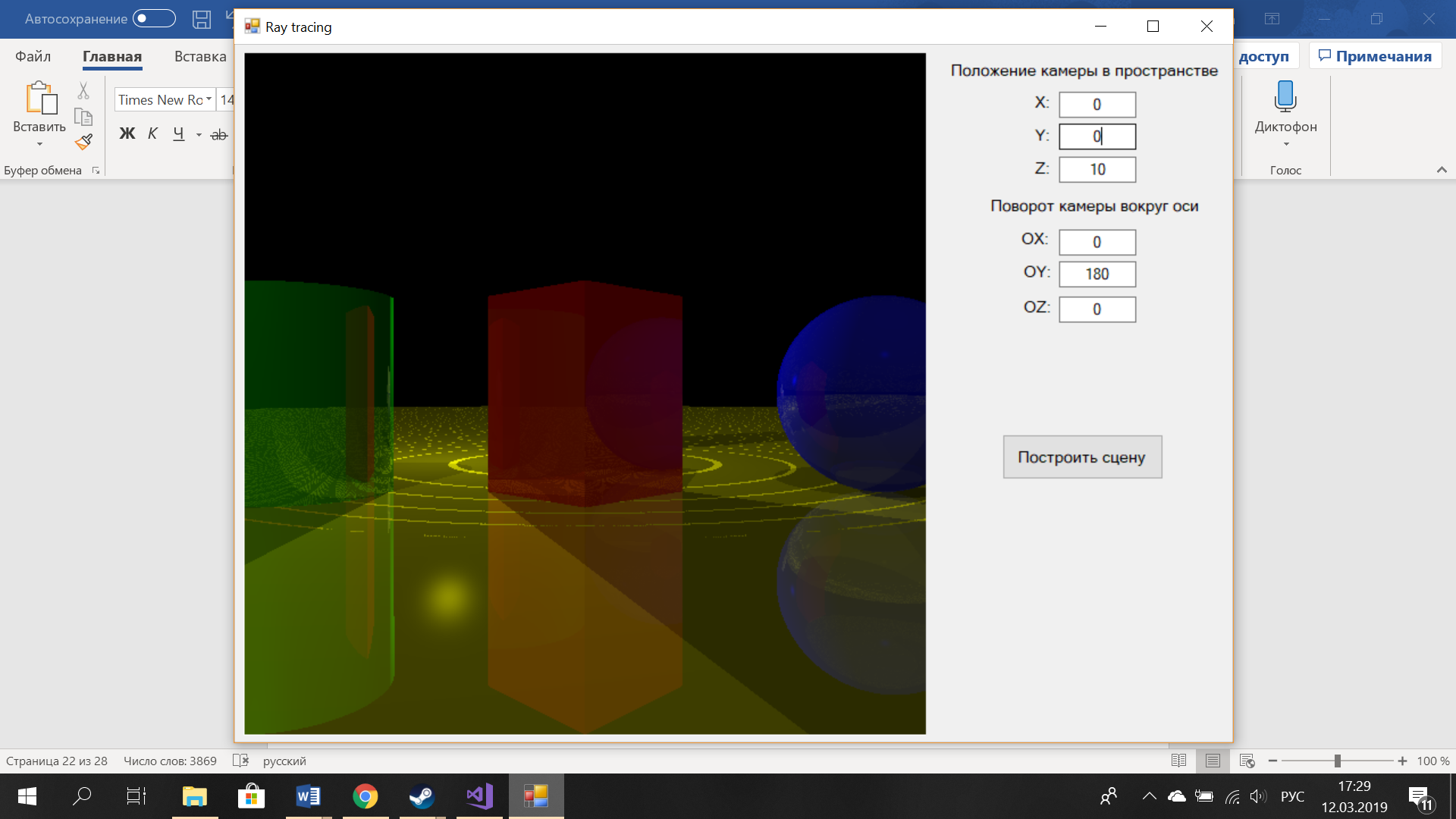


Рисунок 10 – Пример работы программы



Рисунок 11 – Пример работы программы

# Исследовательская часть

В данном разделе будут поставлены, реализованы и проанализированы эксперименты.

* 1. **Постановка эксперимента**

В данной работе предметом для исследования являются алгоритмы прямой и обратной трассировки лучей.

Во-первых, в качестве эксперимента была предпринята попытка реализации алгоритма прямой трассировки лучей.

Во-вторых, так как алгоритм обратной трассировки лучей является далеко не самым быстрым, то естественным было бы исследовать время генерации изображения в зависимости от различных факторов. Была исследована зависимость скорости генерации сцены от эффекта. Сцена состоит из одного геометрического тела и фона. Свойства геометрического тела менялись в каждом опыте, но его размер и положение оставались постоянными. В итоге было проведено четыре опыта для каждого вида геометрического тела:

1. без эффектов;
2. эффект отражения;
3. эффект преломления;
4. эффекты отражения и преломления.

Каждый опыт производился 50 раз, затем выбиралось среднее время.

* 1. **Сравнительный анализ на основе экспериментальных данных**

Реализация алгоритма прямой трассировки лучей производилась без учета эффектов взаимодействия света с геометрическим объектом, т.е. использовались только функции поиска пересечения с геометрическим объектом, и возвращался цвет объекта или фона в зависимости от результата. Реализация функций та же, что и для обратной трассировки лучей, однако лучи испускаются из точечного источника света в произвольном направлении.

Результаты эксперимента представлены на рис. 10.

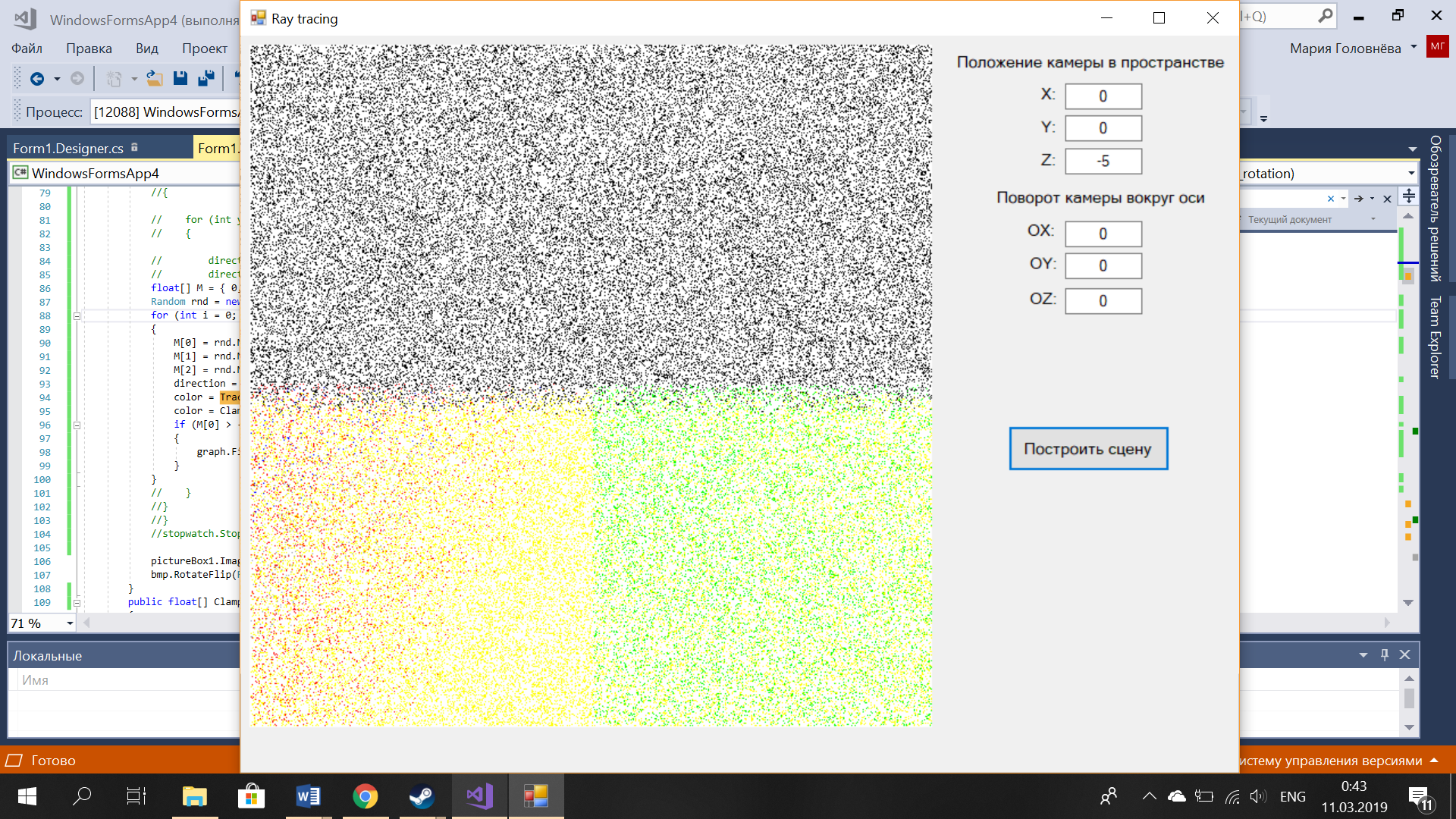


Рисунок 10 – Пример работы алгоритма прямой трассировки лучей

При построении сцены на Рисунке 10 был сгенерирован 1000000 световых лучей в различных направлениях. Время построения сцены составило ~20 секунд.

Таким образом, проблема алгоритма прямой трассировки лучей заключается не только в его медленной работе, но и в том, что в конечном итоге в камеру попадает лишь малая часть лучей, испускаемых источником света, что не дает нам получить желаемое изображение, а также большая часть вычислений оказывается проделанной впустую.

Для сравнения, стоит показать, что при построении сцены на Рисунке 11 генерируется 360000 световых лучей (по одному лучу на каждый пиксель холста). Время построения сцены – 18,194 секунды.

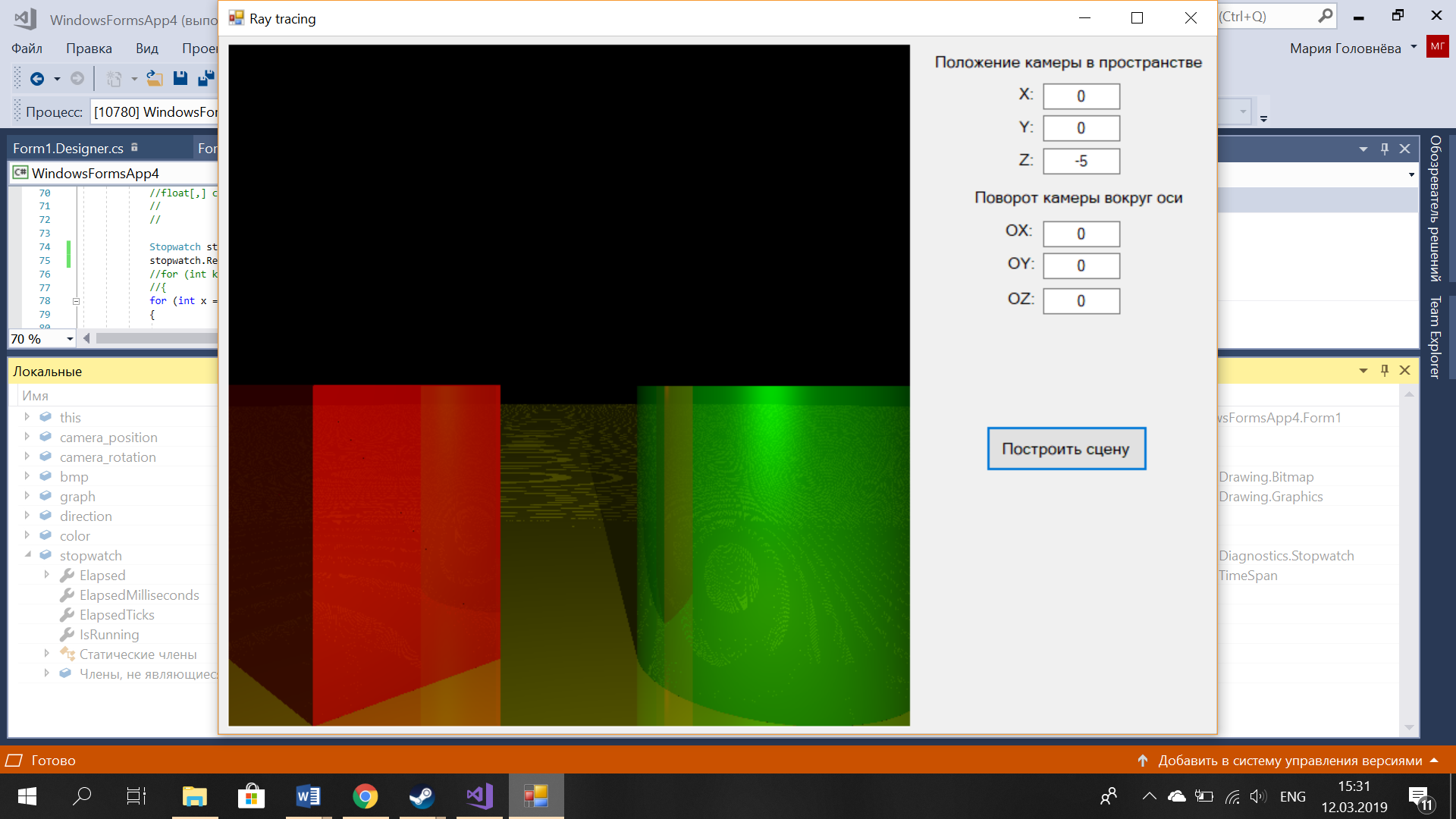


Рисунок 11 – Пример работы алгоритма обратной трассировки лучей

Результаты второго эксперимента представлены в Таблице 1 и на Рисунке 12.

Из результата эксперимента следует, что трассировка лучей является крайне медленным алгоритмом. Самым продолжительным оказалось использование обоих эффектов (преломления и отражения), что является логичным, так как производится больше расчетов. Реализация эффекта преломления оказалась медленнее, чем реализация эффекта отражения.

Таблица 1 – Результаты второго эксперимента

|  |  |  |  |
| --- | --- | --- | --- |
| № опыта | Время построения сферы, мс | Время построения куба, мс | Время построения цилиндра, мс |
| 1 | 2264 | 11102,5 | 2042 |
| 2 | 2299,5 | 11817,8 | 2094,4 |
| 3 | 2587,5 | 11891,3 | 2211,5 |
| 4 | 2800,6 | 12144,6 | 2587,5 |

Если сравнивать геометрические тела, то в разы медленнее оказалось построение куба, состоящего из 12 полигонов.

Рисунок 12 – Время генерации изображения с различными эффектами

# Заключение

В рамках данной работы решены следующие задачи:

1. проведен обзор методов построения реалистичного изображения трехмерной сцены;
2. разработано программное обеспечение для моделирования трехмерной сцены, включающей геометрические тела – сферы, кубы, цилиндры;
3. разработан пользовательский интерфейс, предоставляющий возможность изменять положение камеры в пространстве сцены.

В данной работе были продемонстрированы результаты роботы алгоритмов генерации реалистичных изображений – обратной и прямой трассировки лучей. Только алгоритм обратной трассировки лучей позволил выполнить поставленную задачу.

Был проведен эксперимент, по результатам которого можно сказать, что алгоритм работает крайне медленно.

На сегодняшний день в реальном времени алгоритм обратной трассировки лучей используют лишь в исследовательских целях на сверхмощных компьютерах. Так же считается, что алгоритм трассировки лучей актуален только для изображений искусственных объектов с геометрически простыми формами. Однако алгоритм является интуитивно понятным, так как работает по законам реального мира - по законам физики.

Трассировка лучей является универсальным методом и позволяет строить изображение высокого качества. Именно из-за этих качеств алгоритм достаточно популярен и методы его оптимизации совершенствуются постоянно.

# Список литературы

1. Д. Роджерс. Алгоритмические основы машинной графики./Д. Роджерс. – М.:Мир. 1989.
2. Д. Роджерс. Математические основы машинной графики./Д. Роджерс. – М.:Мир. 2001.
3. Трёхмерная графика с нуля. Часть 1: трассировка лучей [Электронный ресурс. – Режим доступа <https://habr.com/ru/post/342510/>].
4. Методы визуализации в компьютерной графике [Электронный ресурс. – Режим доступа http://csaa.ru/metody-vizualizacii-v-kompjuternoj-grafiki/].