Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«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 курсовому проекту на тему

«Реализация реалистичного изображения аквариума с расположенными внутри источниками света»

Выполнил _____ _____ Орехова Е. О.
Научный
руководитель _____ _____ Кострицкий А. С.

Москва, 2018

Содержание

Введение

Целью курсового проекта является разработка программы, которая генерирует изображение аквариума, наполненного жидкостью, а также содержащего непрозрачные трехмерные объекты и точечные источники света.

В процессе написания курсового проекта должны быть решены следующие задачи:

- Составить модель, основанную на физических законах волновой оптики.
- Изучить алгоритмы компьютерной графики и выбрать наиболее подходящие для решения поставленной задачи.
- Спроектировать архитектуру программы и ее интерфейс.
- Провести исследования разработанной программы.

Результатом работы является программа, демонстрирующая преломление и поглощение лучей света жидкостью. Программа позволяет выводить на экран изображение аквариума с объектами внутри, а также настраивать коэффициенты преломления, поглощения, добавлять или изменять параметры источников света.

1 Аналитический раздел

1.1 Физические явления

Для получения реалистичного изображения необходимо учесть некоторые физические явления, такие как преломление света и поглощение света.

Преломление света

Преломление света — явление, при котором луч света, переходя из одной среды в другую, изменяет направление на границе этих сред.

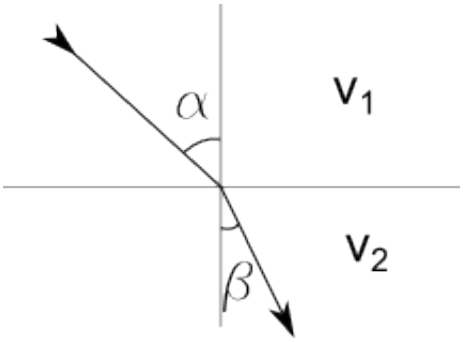


Рисунок 1. Закон преломления света.

Физический смысл относительного показателя преломления (иначе показателя преломления второй среды относительно первой): он показывает во сколько раз скорость света в той среде, из которой луч выходит, больше скорости света в той среде в которую он входит.

$$n = \frac{\sin(\alpha)}{\sin(\beta)} = \frac{v_1}{v_2} = \frac{n_2}{n_1}.$$

Кроме того, каждая среда, через которую проходит луч света, характеризуется абсолютным показателем преломления:

$$n = \frac{c}{v_1}.$$

Абсолютный показатель преломления — это показатель преломления среды относительно вакуума. Он равен отношению скорости света в вакууме к скорости света в данной среде. Среда с меньшим абсолютным показателем преломления называется оптически менее плотной средой.

Идеальное преломление луча на поверхности раздела двух сред с показателями преломления n_1 и n_2 строится по следующим законам преломления:

1. Падающий и преломлённый луч лежат в одной плоскости с вектором нормали, построенным в точке пересечения прямого луча с поверхностью. При этом преломлённый луч и падающий находятся с разной стороны поверхности.

2. Соотношение длин векторов $|\vec{R}| = n|\vec{V}|$, где $n = \frac{n_1}{n_2}$ — относительный коэффициент преломления.

3. Углы падения и преломления удовлетворяют закону Снеллиуса-Декарта:

$$n_1 \sin(\alpha) = n_2 \sin(\gamma).$$

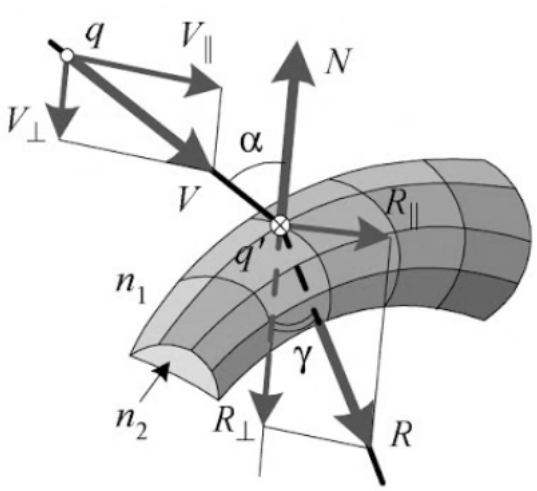


Рисунок 2. Преломление луча на поверхности раздела двух сред.

Вычислим нормальную и тангенциальную составляющие вектора преломленного луча:

$$\vec{R_\perp} = |\vec{R}| \cos(\gamma)\vec{V_\perp} =$$
$$= n|\vec{V}|\sqrt{1 - n^2 \sin^2(\alpha)}\vec{V_\perp} =$$
$$= n|\vec{V}|\sqrt{1 - n^2|\vec{V} \times \vec{N}|^2}\frac{(\vec{V}, \vec{N})}{|(\vec{V}, \vec{N})|}\vec{N} =$$
$$= \mathrm{sgn}((\vec{V}, \vec{N}))n\sqrt{|\vec{V}|^2 - n^2|\vec{V} \times \vec{N}|^2}\vec{N},$$

где sgn — функция знака.

$$\vec{R_{\parallel}} = |\vec{R}|\sin(\gamma)\vec{V_{\parallel}} = n^2|\vec{V}|\sin(\gamma)\vec{V_{\parallel}} = n^2\vec{V_{\parallel}} = n^2(\vec{V} - ((\vec{V},\vec{N}))\vec{N}).$$

Таким образом вектор:

$$\vec{R} = \vec{R_{\perp}} + \vec{R_{\parallel}} =$$
$$= n^2\vec{V} + n(\mathrm{sgn}((\vec{V},\vec{N}))\sqrt{|\vec{V}|^2 - n^2|\vec{V}\times\vec{N}|^2} - n(\vec{V},\vec{N}))\vec{N}. \quad (1)$$

Из равенства (1) следует, что преломленный луч существует, если не отрицательно подкоренное выражение:

$$|\vec{V}|^2 - n^2|\vec{V}\times\vec{N}|^2 > 0 \Rightarrow |\vec{V}| > n|\vec{V}\times\vec{N}| \Rightarrow 1 > n\sin(\alpha). \quad (2)$$

При падении луча на оптически менее плотную среду под углом $\alpha \geqslant \arcsin(\frac{1}{n})$ преломленный луч отсутствует.

Поглощение света

Поглощением (абсорбцией) света называется явление потери энергии световой волной, проходящей через вещество. Свет поглощается в тех случаях, когда проходящая волна затрачивает энергию на различные процессы. Среди них:

- преобразование энергии волны во внутреннюю энергию – при нагревании вещества;
- затраты энергии на вторичное излучение в другом диапазоне частот (фотолюминесценция);
- затраты энергии на ионизацию – при фотохимических реакциях;
- т.п.

Интенсивность волны будет изменяться по закону Бугера:

$$J(x) = J_0 e^{-\alpha x}, \quad (3)$$

где x — толщина поглощающего слоя, J_0 — интенсивность волны на входе в среду, α — коэффициент поглощения, зависящий от длины волны света, химической природы и состояния вещества и не зависящий от интенсивности света при слабых световых потоках.

1.2 Алгоритмы удаления невидимых поверхностей

Для решения поставленной задачи необходимо выбрать алгоритм удаления невидимых линий и поверхностей. Рассмотрим некоторые из них.

Алгоритм Робертса

Алгоритм Робертса относится к алгоритмам, работающим в объектном пространстве, применяется для изображения множества выпуклых многогранников на одной сцене с удалёнными невидимыми линиями. Метод непригоден непосредственно для передачи падающих теней и других сложных визуальных эффектов. Алгоритм Робертса состоит из двух этапов:

1. Необходимо удалить из каждого многогранника те ребра или грани, которые экранируются самим телом. Робертс использовал для этого простой тест: если одна или обе смежные грани обращены своей внешней поверхностью к наблюдателю, то ребро является видимым. Тест этот выполняется вычислением скалярного произведения координат наблюдателя на вектор внешней нормали грани: если результат отрицательный, то грань видима.

2. Каждое из видимых рёбер каждого многогранника сравнивается с каждым из оставшихся многогранников для определения того, какая его часть или части, если таковые есть, экранируются этими телами. Для этого в каждую точку ребра проводится отрезок луча, выходящего из точки расположения наблюдателя. Если отрезок не пересекает ни одного из многогранников, то точка видима. Для решения этой задачи используются параметрические уравнения прямой, содержащей ребро, и луча.

Алгоритм Варнока

В отличие от алгоритма Робертса, Варнок в 1968 г. предложил алгоритм, работающий в пространстве образа. Главная идея алгоритма основана на гипотезе о способе обработки информации глазом и мозгом человека. Эта гипотеза заключается в том, что тратится очень мало времени и усилий на обработку областей, которые содержат мало информации. Большая часть времени и сил уходит на обработку областей с высоким информационным содержимым.

В алгоритме Варнока (и его вариантах) считается, что большие области изображения однородны, т.е. смежные области (пиксели) вдоль обеих осей x и y имеют тенденцию к однородности. Такое свойство называют когерентностью. Алгоритм заключается в следующем: в пространстве изображения рассматривается окно, решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на части до тех пор, пока содержимое окна не станет достаточно простым для визуализации или его размер не достигнет предела разрешения. В последнем случае информация, содержащаяся в окне, усредняется, и результат изображается с одинаковой интенсивностью или цветом.

Алгоритм трассировки лучей

Идея алгоритма была предложена А. Аппелем в 1968 году, впервые алгоритм был реализован в 1971 году.

Аппель предложил трассировать лучи от наблюдателя к объекту. В первой реализации этого метода луч использовался только для обработки скрытых или видимых поверхностей.

В алгоритме каждый луч, выпущенный из камеры, проходит через пиксель растра до сцены. Проверяется пересечение каждого объекта сцены с каждым лучом. Если в сцене рассматривается много объектов, то мы получим большое количество пересечений. Эти пересечения упорядочиваются по глубине. Пересечение с минимальным значением z представляет видимую поверхность для данного пикселя. Свойства этого объекта используются для определения характеристик пикселя.

Процедура определения пересечений является наиболее важной и трудо-

ёмкой задачей в этом алгоритме.

Однако задача формирования изображения не заканчивается нахождением самой точки пересечения: если для решения задачи учитываются эффекты отражения и преломления, необходимо отслеживать дальнейший путь отражённого или преломлённого луча.

Алгоритм, использующий z-буфер

Впервые этот алгоритм был предложен Кэтмулом в 1975 году. Алгоритм работает в пространстве изображения и является обобщением идеи о буфере кадра. Буфер кадра запоминает атрибуты каждого пикселя в пространстве изображения, а Z-буфер запоминает глубину (расстояния от картинной плоскости) в пространстве изображения каждого видимого пикселя. Сцены могут быть любой сложности. Поскольку размеры пространства изображения фиксированы, вычислительная трудоёмкость алгоритма не более чем линейна. Недостатки алгоритма: требуется большой объем памяти; трудно реализовать эффекты, связанные с прозрачностью.

1.3 Выбор алгоритма удаления невидимых линий и поверхностей

Для решения поставленной задачи был выбран алгоритм трассировки лучей, т.к. в этом алгоритме можно реализовать эффект преломления, наблюдая дальнейший путь преломлённого луча.

1.4 Алгоритмы закраски

Модель освещения Ламберта

Модель Ламберта позволяет реализовать диффузное освещение. Метод основан на том, что свет при попадании на поверхность рассеивается равномерно во все стороны. Таким образом, освещённость в точке определяется только плотностью света в точке поверхности, а она линейно зависит от косинуса угла падения. Модель Ламберта является одной из самых простых моделей освещения. Данная модель очень часто используется в комбинации

других моделей, практически в любой другой модели освещения можно выделить диффузную составляющую. Недостатком данной модели является то, что все точки грани будут иметь одинаковую интенсивность.

Модель освещения Фонга

Модель Фонга — классическая модель освещения. Представляет собой сумму диффузной составляющей (модели Ламберта) и зеркальной составляющей, таким образом на материале может появляться блик.

Расчёт освещения по Фонгу требует вычисления цветовой интенсивности трёх компонент освещения: фоновой, рассеянной и глянцевых бликов

$$I = K_a I_a + K_d(\vec{n}, \vec{l}) + K_s(\vec{n}, \vec{h})^p,$$

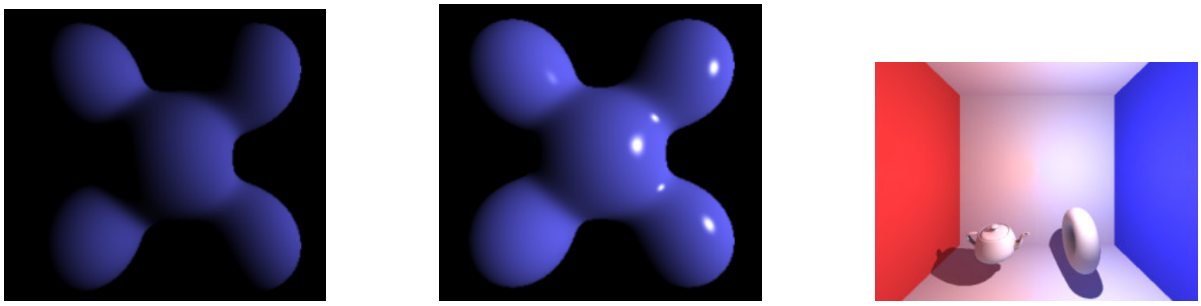
$\vec{n}$ — вектор нормали к поверхности в точке;
$\vec{l}$ — направление проецирования(направление на источник света);
$\vec{h}$ — направление на наблюдателя;
K_a — коэффициент фонового освещения;
K_s — коэффициент зеркального освещения;
K_d — коэффициент диффузного освещения;
p — коэффициент зеркальности объекта.

Фотонная модель освещения

В фотонной модели освещения источник представляет собой плоскость, излучающую фотоны. По физическим законам определяется интенсивность и направление движения этих фотонов, которые определяют освещённость тех или иных граней. При использовании этой модели получается реалистичное изображение, соответствующее физическим законам.

1.5 Обоснование выбора алгоритма

На рисунке (3) представлены результаты работы различных алгоритмов освещения



(a) (b) (c)

Рисунок 3. Модели освещения: (a) Ламберта; (b) Фонга; (c) Фотонная.

Поскольку в поставленной задаче отсутствует отражение нет смысла использовать модель Фонга. Фотонная модель даёт возможность получить качественное изображение, однако в данной модели предполагаются довольно сложные вычисления, из-за чего ухудшается скорость визуализации. В итоге для решения задачи была выбрана модель освещения Ламберта.

2 Конструкторский раздел

После запуска программы загружаются модели аквариума и жидкости из файла, расположенного в той же директории, что и исполняемый файл. При необходимости загружаются дополнительные объекты и лампочки. После нажатия кнопки «Нарисовать» начинается генерация изображения. Пользователь может изменять коэффициенты поглощения сред; интенсивность внешнего освещения; добавлять или удалять дополнительные объекты; добавлять, удалять, изменять расположение и интенсивность источников света; менять положение камеры.

2.1 Общий алгоритм работы программы

Алгоритм 1. Общий алгоритм работы программы.

```
1:  Начало алгоритма
2:      загрузить модель аквариума
3:      загрузить параметры жидкости из файла
4:      загрузить модели объектов
5:      загрузить параметры освещение
6:      Цикл для каждого вектора «камера – пиксель»:
7:          найти все пересечения
8:          Цикл для каждой точки пересечения
9:              вычислить интенсивность освещения
10:         Конец цикла
11:         вычислить результирующий цвет
12:         поставить пиксель с результирующим цветом
13:     Конец цикла
14:     вывести изображение
15: Конец алгоритма
```

2.2 Подробный алгоритм работы программы

Вычисление векторов камера-пиксель

Камера задаётся 3-мя параметрами – позицией, направлением взгляда и направлением «вверх».

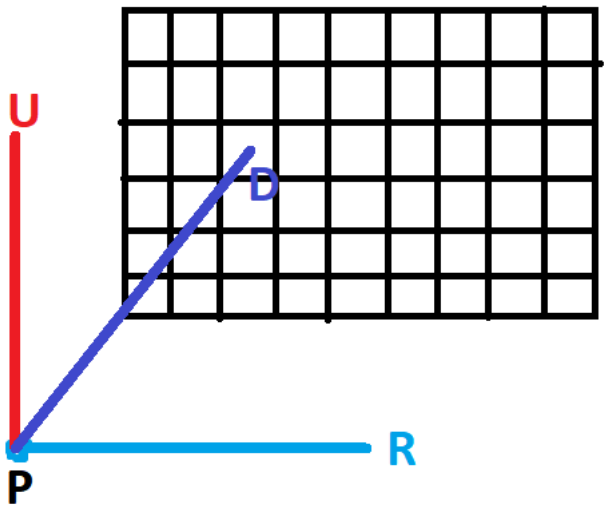


Рисунок 4. Параметры камеры.

На изображении точка P обозначает позицию камеры, вектор $\vec{U}$ — вектор «вверх»,$\vec{D}$ — вектор направления. Задача состоит в том, чтобы каждому пикселю дисплея сопоставить вектор $\vec{PK}$, где K — точка в пространстве, куда направлен вектор, выходящий из позиции камеры и проходящий через пиксель на рамке. $\vec{PK}$ назовём лучом из камеры.

Чтобы найти вектор $\vec{R}$ необходимо векторно умножить вектора $\vec{U}$ и $\vec{D}$, так как $\vec{R}$ перпендикулярен плоскости, в которой находятся $\vec{U}$ и $\vec{D}$.

$$\vec{R} = [\vec{U} \times \vec{D}].$$

Тогда $\vec{PK}$ можно выразить по формуле:

$$\vec{PK} = \vec{D} + \vec{R}(\frac{x}{\text{width}} - 0.5) + \vec{U}(\frac{y}{\text{height}} - 0.5),$$

где x, y — координаты пикселя на экране, а width и width — ширина и высота дисплея соответственно.

Поиск всех пересечений

Алгоритм 2. Поиск всех пересечений.

```
1: Начало алгоритма
2:     глубина рекурсии = 0
3:     До тех пока глубина рекурсии меньше максимальной рекурсии:
4:         найти ближайшее пересечение
5:         Если пересечение есть тогда
6:             поместить точку пересечения в стек
7:             Если пересекаемый объект непрозрачный тогда
8:                 Конец
9:             Иначе вычислить преломлённый луч
10:                Если преломленный луч существует тогда
11:                    увеличить глубину рекурсии на 1
12:                    заменить вектор на преломленный луч
13:                Конец условия
14:            Конец условия
15:        Конец условия
16:    Конец цикла
17: Конец алгоритма
```

Вычисление ближайшего пересечения

Все объекты задаются в виде набора треугольников. Алгоритм поиска ближайшего пересечения будет выглядеть следующим образом:

Алгоритм 3. Алгоритм поиска ближайшего пересечения луча из камеры с треугольником.

```
1: Начало алгоритма
2:     Для каждого треугольника
3:     Найти плоскость, в которой лежит треугольник
4:     Найти точку, в которой луч из камеры пересекает эту плоскость
5:     Узнать, лежит ли эта точка внутри треугольника
6:     Если точка лежит внутри треугольника, и расстояние до этой точки
       меньше расстояния до рассмотренных ранее треугольников тогда
7:         Запомнить треугольник как ближайший
8:     Конец условия
9: Конец алгоритма
```

Разберём подробнее каждый пункт. Пусть треугольник задан точками A, B, C и вектором нормали $\vec{N}$. Камера находится в точке O. Луч из камеры

обозначим как $\vec{D}$. Решим вопрос о пересечении луча и плоскости

$$t = \frac{(\vec{N}, \vec{A}) - (\vec{N}, \vec{O})}{(\vec{N}, \vec{D})}.$$

Если $t < 0$, то луч не пересекает плоскость. Для того, чтобы найти точку пересечения P, надо отложить из точки O t раз вектор $\vec{D}$

$$P = O + t * \vec{D}.$$

Для определения принадлежности точки треугольнику проверяется следующее условие: если точка лежит в треугольнике ABC, то треугольники ABP, ACP, BCP будут просто частями треугольника ABC, и сумма их площадей будет равна площади ABC.

Вычисление преломлённого луча

В Аналитическом разделе были изложены физические принципы преломления.

Алгоритм 4. Алгоритм вычисления преломлённого луча.

1: **Начало алгоритма**
2: Вычислить нормаль к поверхности в точке пересечения
3: **Если** Не выполняется условие (2) **тогда**
4: Преломлённый луч отсутствует. Конец.
5: Найти направляющий вектор по формуле (1)
6: **Конец условия**
7: **Конец алгоритма**

Вычисление интенсивности освещения

Есть точка P для которой надо посчитать интенсивность, пусть есть ещё источник освещения I, а интенсивность освещения сцены (без учёта ламп) равна I_{scene}. Луч из точки I попадает в точку P. x — длина участка IK, y — длина участка KP, где точка K лежит на границе сред. Т.к. каждый из участков имеет свой коэффициент поглощения, то интенсивность света в точке P будет выражаться как:

$$I_P = (I_I e^{-\alpha_1 x}) e^{-\alpha_2 y}.$$

Алгоритм 5. Алгоритм вычисления интенсивности в точке.

1: **Начало алгоритма**
2: $I_P = I_{\text{scene}}$
3: **Цикл** каждого источника света
4: трассируем луч из источника света в точку
5: Вычисляем пересечение с объектами
6: **Если** есть пересечение **тогда**
7: **Если** объект непрозрачный **тогда**
8: Интенсивность от этого источника равна 0
9: **Иначе**Уменьшить интенсивность от этого источника света по формуле (3)
10: **Конец условия**
11: **Иначе**Уменьшить интенсивность от этого источника света по формуле (3)
12: **Конец условия**
13: Прибавить результат к I_P
14: **Конец цикла**
15: Вернуть I_P
16: **Конец алгоритма**

Вычисление результирующего цвета

Так как в сцене присутствуют как прозрачные, так и непрозрачные объекты, то недостаточно просто взять цвет ближайшего к камере объекта. Для решения поставленной задачи необходимо уметь смешивать цвета, при этом нужно учитывать поглощение среды.

Для того, чтобы вычислить цвет результирующего пикселя, необходимо заполнить следующую таблицу

1

, пусть интенсивность света в точке A равна I_A, в точке B — I_B :

Таблица 1. Таблица вычисления цвета

Предыдущий цвет	Интенсивность точки
Colour1	I_A
Colour1*$e^{-\alpha * x}$+Colour2	$I_A * e^{-\alpha * x} + I_B$
...	...

После заполнение таблицы остаётся только умножить последнюю строчку первого столбца на последнюю строчку второго столбца. В итоге мы получаем значение результирующего цвета с определённой интенсивностью.

3 Технологический раздел

3.1 Выбор и обоснование языка программирования

В качестве языка программирования был выбран C#. Этот язык позволяет разрабатывать приложения, имеющие графический интерфейс. C#поддерживает объектно-ориентированный подход к разработке, а следовательно позволяет представлять объекты на сцене как объекты в языке, что упрощает разработку, а также поиск ошибок.

3.2 Схемы классов

В данном подразделе представлены основные классы приложения

Общая схема классов

Классовая диаграмма показана на рисунке (6)

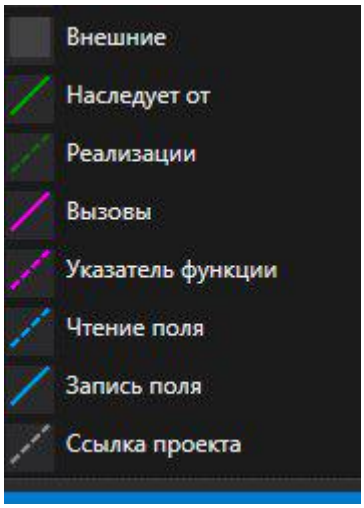


Рисунок 5. Обозначения для схемы классов.

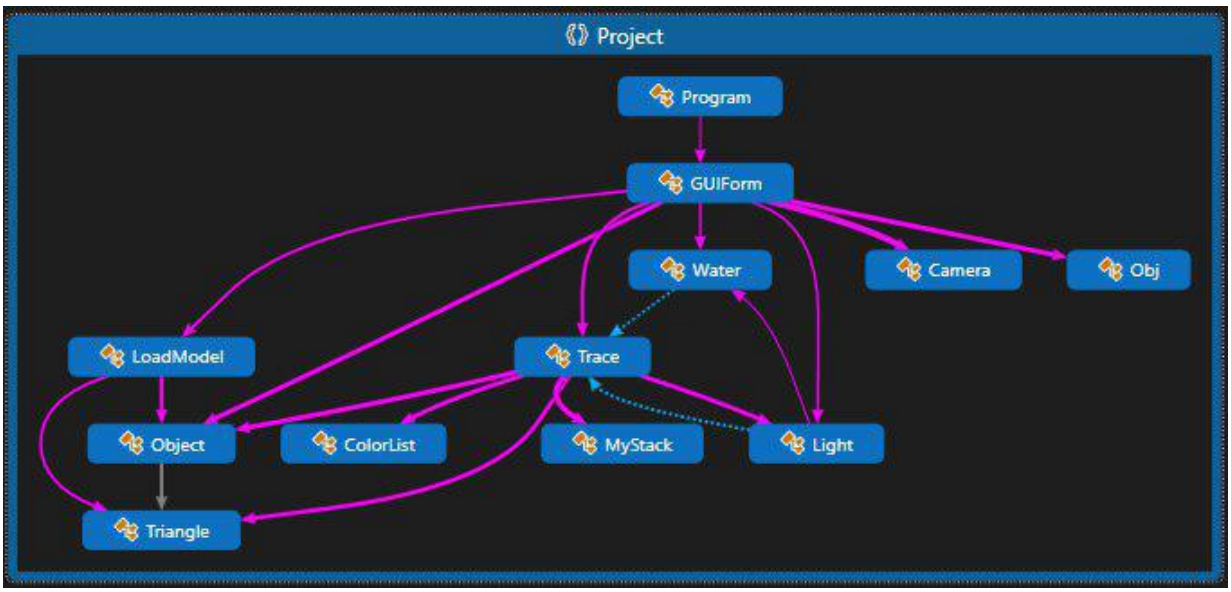


Рисунок 6. Общая схема классов.

Назначение классов

Описание для основных классов приложения

GUIForm — класс пользовательского интерфейса.

Water — класс для описания жидкости

Camera — класс для описания камеры

LoadModel — загрузка объектов из файлов

Trace — класс, выполняющий трассировку луча

Light — класс, для описания источников света

Object — класс, для описания всех остальных объектов

3.3 Интерфейс программы

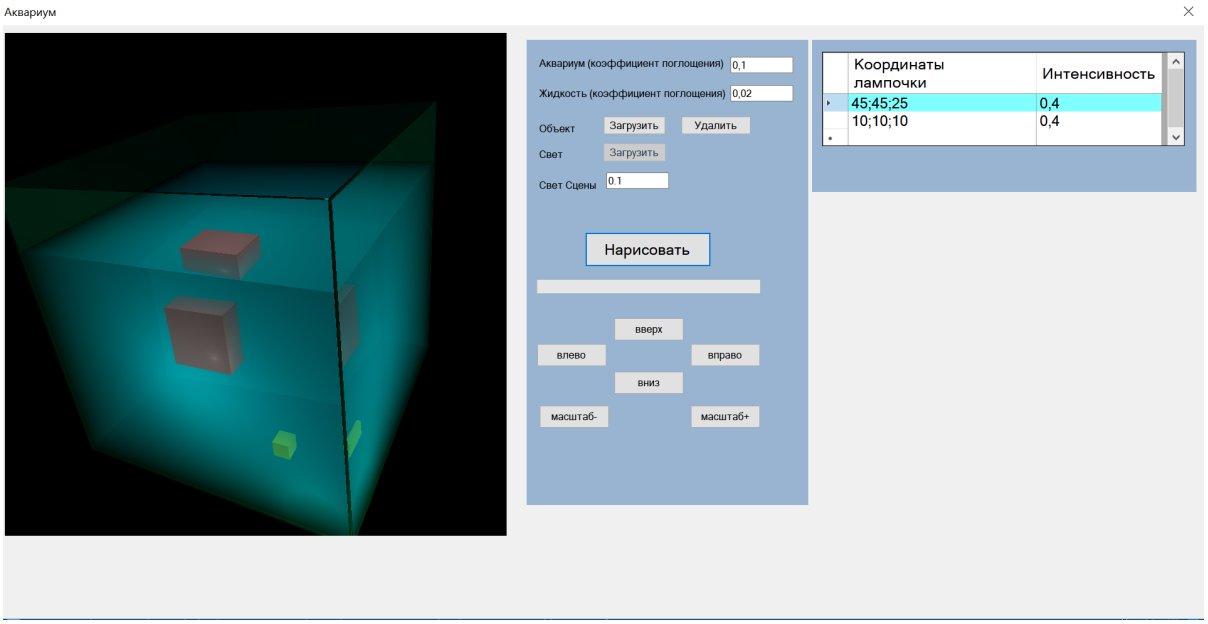


Рисунок 7. Интерфейс программы.

4 Исследовательский раздел

Было измерено время построения изображения в зависимости от количества треугольников (в этом эксперименте источники света отсутствовали).

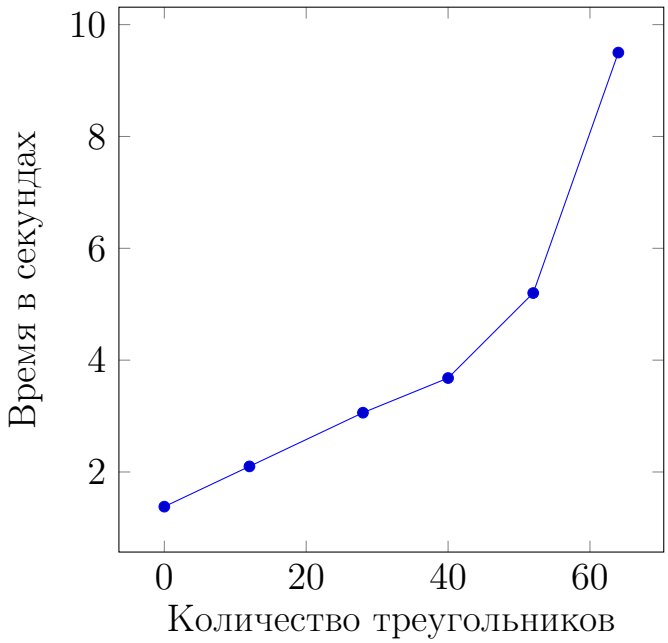


Рисунок 8. Время построения в зависимости от количества треугольников.

Также было измерено время построения изображения с учётом освещения.

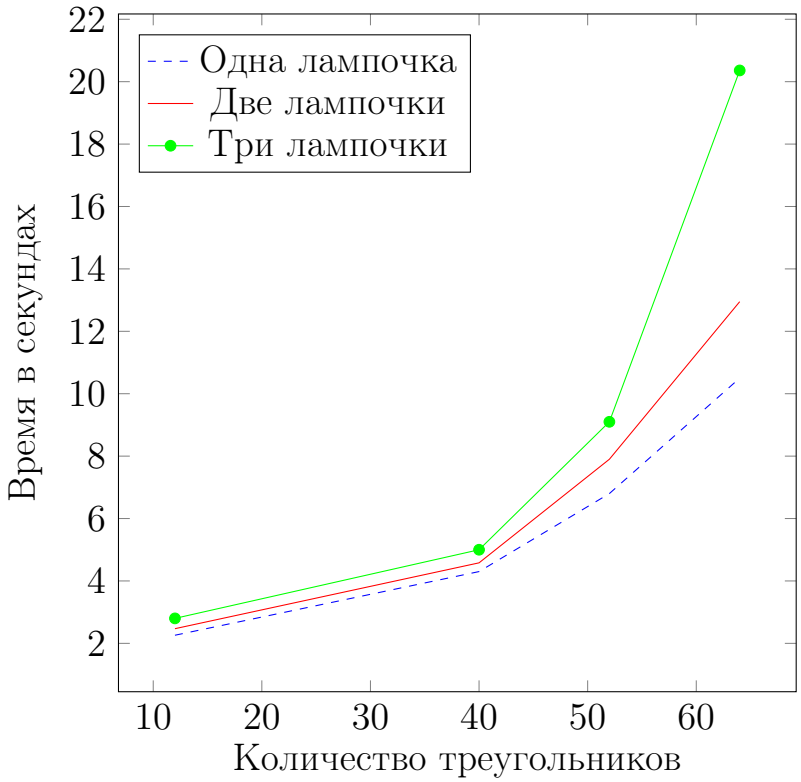


Рисунок 9. Время построения при разном количестве ламп.

Исходя из полученных данных, следует вывод, что время генерации изображения растёт по экспоненте и зависит от количества треугольников. Оптимальное время построения изображения (4-5 секунд) достигается при 45-55 треугольниках.

Технические характеристики ЭВМ, на которой были проведены тесты:

1. Процессор: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
2. ОЗУ: 6 Гб DDR3

Заключение

При написании проекта были рассмотрены и проанализированы алгоритмы генерации реалистичного изображения, проанализированы их достоинства, недостатки, а также возможность использования для решения поставленной задачи. Для реализации данной задачи были выбраны советующие алгоритмы. Проведены экспериментальные исследования, по материалам которых подготовлена расчетно-пояснительная записка.

Разработанная программа позволяет получать на экране дисплея реалистичную модель аквариума, наполненного жидкостью, и дополнительных объектов, расположенных внутри жидкости. Пользователь может добавлять или удалять дополнительные объекты, а также добавлять, удалять или изменять положение и интенсивность точечных источников света, расположенных внутри жидкости.

Список литературы

1. Роджерс, Д. Алгоритмические основы машинной графики: Пер. с англ. / Д. Роджерс —М.: Мир, 1989. —512 с., ил. —ISBN 5-03-000476-9

2. Никулин, Е.А. Компьютерная геометрия и алгоритмы машинной графики. СПб.: БХВ—Петербург, 2003. —560 с.: ил. —ISBN 5-94157-264-6

3. Преломление света. Волновая оптика [Электронный ресурс]. — Режим доступа : http://class-fizika.narod.ru/voln3.htm

4. Трёхмерная графика с нуля [Электронный ресурс]. — Режим доступа: https://habrahabr.ru/post/342510/

5. Поглощение света [Электронный ресурс]. — Режим доступа: https://www.booksite.ru/fulltext/1/001/008/090/168.htm