

a

b

Оглавление

Введение	5
1 Аналитическая часть	6
1.1 Формализация сцены	6
1.2 Представление объектов сцены	6
1.3 Анализ рассматриваемых алгоритмов	7
1.3.1 Алгоритм, использующий Z-буфер	7
1.3.2 Метод Гуро	7
1.3.3 Метод Фонга	7
1.3.4 Растровая развертка треугольника	8
1.3.5 Обратная трассировка лучей	8
Вывод	9
2 Конструкторская часть	10
2.1 Модель освещения Фонга	10
2.2 Метод Гуро	11
2.3 Метод Фонга	12
2.4 Алгоритм, использующий Z-буфер	12
2.5 Растровая развертка треугольника	14
2.6 Обратная трассировка лучей	14
2.7 Поиск пересечения луча со сферой	16
2.8 Поиск пересечения луча с треугольником	16
2.9 Поиск отраженного луча	17
Вывод	17
3 Технологическая часть	18
3.1 Выбор языка и среды программирования	18
3.2 Замеры времени	18
3.3 Интерфейс программы	19
Вывод	21
4 Экспериментальная часть	22
4.1 Цель эксперимента	22
4.2 Апробация	22
4.3 Описание эксперимента	24
4.3.1 Первый случай	24
4.3.2 Второй случай	26

Вывод	26
Заключение	28
Заключение	29

Введение

В наше время одна из основных задач компьютерной графики состоит в создании и развитии алгоритмов построения реалистичных изображений. Данные алгоритмы используются повсеместно, например: компьютерные игры, дизайн, кинематограф и т.д.

Для построения изображения, максимально похожего на то, что мы видим в реальной жизни, необходимо учитывать такие явления как: преломление, отражение, рассеивание света и многие другие.

На данный момент существует множество алгоритмов для синтеза реалистичного изображения, каждый из которых имеет свои достоинства и недостатки.

Цель данной работы - реализовать и сравнить алгоритм Z-буфера с закраской по Гуро, алгоритм Z-буфера с закраской по Фонгу и алгоритм обратной трассировки лучей.

Для достижения поставленной задачи необходимо:

1. проанализировать рассматриваемы алгоритмы компьютерной графики;
2. реализовать данные алгоритмы;
3. провести сравнительный анализ.

Итогом работы над данным проектом должны стать программа, демонстрирующая работу этих алгоритмов, а также выводы, сделанные на основе проведенного сравнительного анализа.

1 | Аналитическая часть

1.1 Формализация сцены

На сцене могут располагаться некоторые количество геометрических объектов: икосаэдров, параллелипипедов и прямых трехгранных призм. Положение и размер икосаэдра должен задаваться координатами центра описанной окружности и её радиусом, параллелипипеда - также координатами центра, шириной, высотой и глубиной, прямой трехгранной призмы - так же как и параллелипипеда.

Каждый объект обладает характеристиками: цвет, коэффициент рассеивания, коэффициент отражения, коэффициент блеска.

На сцене также могут находиться некоторые количество точечных источников света. Каждый источник задается координатой в пространстве и интенсивность.

1.2 Представление объектов сцены

Необходимо выбрать модель представления объектов на сцене. Для реализации поставленной задачи предпочтительнее будет использовать поверхностные модели, так как для синтеза реалистичного изображения необходимо знать только внешний вид изображаемого объекта.

Для задания поверхностной модели я буду использовать полигональную сетку. Полигональная сетка – представление поверхности геометрического объекта в виде набора состыкованных друг с другом плоских полигонов. В качестве полигонов мою были выбраны треугольники, так как это, в дальнейшем, упростит производимые расчеты.

Существует множество представлений полигональной сетки. Ниже я рассмотрю четыре основных представления: Вершинное, Список граней, «Крылатое» и Веер треугольников.[1]

- Вершинное представление – описывает объект как множество вершин, соединенных с другими вершинами. Дает выигрыш по памяти, но проигрыш в скорости обхода.
- Список граней – представляет объект как множество граней и множество вершин. Требует больше памяти, чем вершинное представление, но позволяет находить соседние грани и вершины за постоянное время.
- «Крылатое» представление – явно представляет вершины, грани и ребра сетки. Требует значительно больше памяти чем список граней и вершинное представление, но позволяет увеличить скорость обхода сетки.
- Веер треугольников – описывает множество соединенных треугольников, которые делят одну центральную вершину. Данное представление требует мало памяти и позволяет производить быстрый обход сетки, но требует больших затрат при изменении.

В данной работе мною будет использован список граней, так как я считаю, что в данной работе допустимо пренебречь дополнительными затратами по памяти ради уменьшения затрат на написания программного продукта.

1.3 Анализ рассматриваемых алгоритмов

1.3.1 Алгоритм, использующий Z-буфер

Идея алгоритма, использующего Z-буфер, заключается в наличии двух буферов: буфер кадра и буфер глубины (он же Z-буфер). В первом хранится информация об атрибутах каждого пикселя экрана. Второй используется для хранения Z координаты каждого пикселя.[2]

Перед началом работы данного алгоритма, каждый, находящийся на сцене объект, переводится в растр. Далее необходимо пройти по всем пикселям каждого объекта. В процессе прохода координата Z каждого нового пикселя сравнивается со значением, записанным в Z-буфере. Если сравнение показывает, что Z координата нового пикселя больше записанного, то новый пиксель заносится в буфер кадра. Также производится запись новой координаты Z в соответствующее значение Z-буфера. В ином случае никаких действий не производится.

Основное преимущество этого алгоритма заключается в его простоте. Также он позволяет обрабатывать объекты в произвольном порядке, что позволяет экономить время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма – большой объем требуемой памяти. Другой недостаток заключается в трудоемкости реализации эффектов прозрачности и просвечивания.

1.3.2 Метод Гуро

Метод Гуро – метод закрашивания в трехмерной компьютерной графики (так же еще называют методом затемнения и сглаживания). Данный метод основывается на идеи закрашивания каждого полигона не одним цветом, а плавно изменяющимися оттенками, вычисляемыми путем интерполяции цветов примыкающих граней. Так как в нашем случае каждая грань является треугольником, достаточно вычислить освещенность в каждой вершине и далее путем билинейной интерполяции находить освещенность в каждой точке грани.[3]

К недостаткам метода Гуро относят то, что он хорошо работает только с диффузной моделью отражения.

1.3.3 Метод Фонга

Метод Фонга – также является методом закрашивания (затемнения, сглаживания), основой которого является интерполяция векторов нормалей. В нашем случае объект представляет собой множество плоских граней, соответственно для реализации метода Фонга достаточно найти нормаль в каждой грани, и на её основе вычислять освещенность для каждого пикселя.[4]

Этот метод требует больших вычислительных затрат, чем метод Гуро, но дает в итоге позволяет получить более качественное изображение, так как в данном методе освещенность вычислется для пикселя грани, а не только для её вершин, как в метода Гуро.

1.3.4 Растровая развертка треугольника

Для разложения треугольника в растр изначально производится одновременное разложение в растр двух ребер рассматриваемого треугольника методом цифрового дифференциального анализатора (ЦДА)[5]. При вычислении новых значений точек, принадлежащих какому из отрезков, производится разложение в растр отрезка, полученного из двух найденных точек. Таким образом получается найти все точки принадлежащие рассматриваемому треугольнику.

На данном рисунке точки a , b , c это точки образующие треугольник. Точка ab' - это точка принадлежащая отрезку ab , полученная в результате перевода в растр этого отрезка. Точка ac' - точка, соответственно, принадлежащая отрезку ac . Стрелочками показано направление движение при переводе в растр.

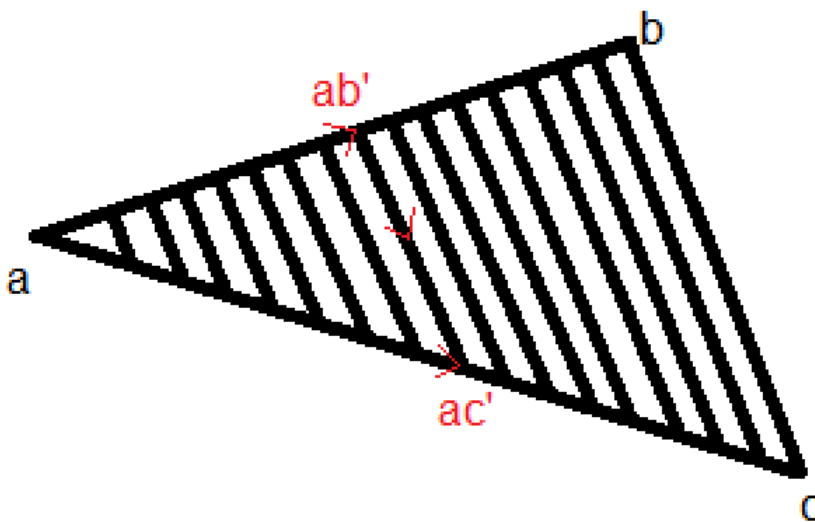


Рисунок. 1.1. Пример разложения треугольника в растр.

1.3.5 Обратная трассировка лучей

Идея обратной трассировки лучей заключается в том, что из виртуального глаза, находящегося на некотором расстоянии от экрана, испускается луч и находится точка его пересечения с объектом сцены. Далее определяется освещенность найденной точки. Делается это путем испускания из данной точки лучей ко всем источникам света и определения их вклада в общую освещенность. Далее из найденной точки пересечения рекурсивно испускается отраженный луч, для поиска объектов, отражающихся в данном. Данный процесс повторяется до тех пор, пока луч не выходит за пределы сцены. Также этот процесс можно ограничить максимальной глубиной рекурсии.[6]

Данный алгоритм позволяет не только закрасивать объекты, но также и определять их видимость, что делает его достаточно универсальным. Достоинства обратной трассировки лучей:

- возможность рендеринга гладких объектов без их предварительной аппроксимации;
- возможность эффективного распараллеливания вычислений.

Основным недостатком этого алгоритма является относительно низкая производительность.

Вывод

В данном разделе осуществлен выбор способа хранения информации об объектах. Выбор пал на список граней, так как он позволяет уменьшить затраты на написание программного продукта из-за простоты своей реализации. На ряду с этим описанны рассматриваемые алгоритмы.

2 | Конструкторская часть

2.1 Модель освещения Фонга

В данной работе было решено использовать модель освещения Фонга[5]. Данная модель учитывает три компоненты освещения: Фоновую (ambient), рассеянную (diffuse) и зеркальную (specular).

Фоновое освещение присутствует в любом уголке сцены. Это происходит из-за многократного отражения лучей света от окружающих объектов.

Рассеянное освещение – тот свет, что отражается от поверхности во всех направлениях.

Зеркальное освещение – часть отраженных лучей, попадающих в глаза наблюдателю.

Для вычисления общей освещенности от одного точечного источника в некоторой точке пространства будет использована формула:

$$I = I_a + K_d * (\bar{n}, \bar{l}) * I_n + K_s * (\bar{V}, \bar{R})^p * I_n \quad (2.1)$$

I_a - интенсивность фонового освещения

K_d - коэффициент рассеянного освещения

K_s - коэффициент зеркального освещения

P - коэффициент блеска

I_n - мощность n-го источника света, рассчитывается как

$\bar{n}, \bar{l}, \bar{V}, \bar{R}$ - единичные вектора. Нормаль, до источника света, до точки наблюдения, отраженного луча.

Если источников больше одного, тогда для вычисления итоговой освещенности необходимо сложить вклад в рассеянное и зеркальное отражение каждого источника.

2.2 Метод Гуро

Описание алгоритма:

1. рассчитать нормаль к каждой грани (так как в данном случае каждая грань является треугольником, нормаль можно вычислить путем векторного произведения двух сторон);
2. определить нормаль в вершинах рассматриваемой грани, путем усреднения нормалей прилегающих граней;
3. закрасить грань цветом, соответствующим билинейной интерполяции значений интенсивности в вершинах.

Рассмотрим билинейную интерполяцию интенсивностей.

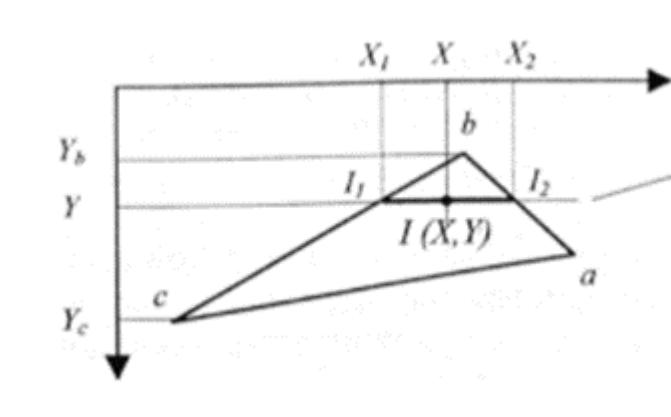


Рисунок 2.1. Пример билинейной интерполяции интенсивностей.

Интенсивность I определяется исходя из пропорции:

$$(I - I_1)/(X - X_1) = (I_2 - I_1)/(X_2 - X_1) \quad (2.2)$$

Отсюда следует:

$$I = I_1 + (I_2 - I_1)(X - X_1)/(X_2 - X_1) \quad (2.3)$$

Значения I_1 , I_2 рассчитываются как:

$$I_1 = I_b + (I_c - I_b)(Y - Y_b)/(Y_c - Y_b) \quad (2.4)$$

$$I_2 = I_b + (I_a - I_b)(Y - Y_b)/(Y_a - Y_b) \quad (2.5)$$

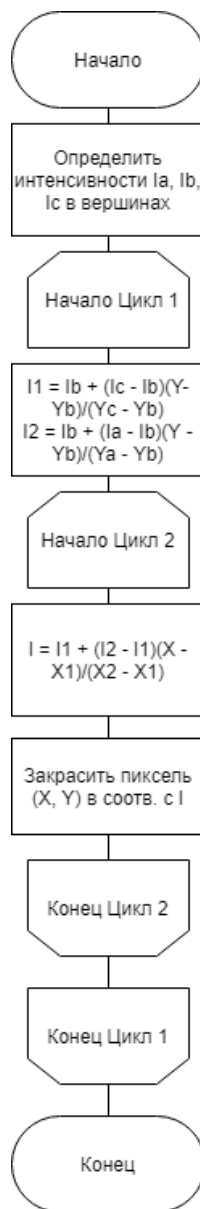


Рисунок 2.2. Схема алгоритма закраши Гуро.

2.3 Метод Фонга

Описание алгоритма:

1. вычисление нормали к рассматриваемой грани;
2. произвести расчет освещенности каждого пикселя грани, на основе вычисленной нормали.

2.4 Алгоритм, использующий Z-буфер

Описание алгоритма:

1. элементам буфера кадра присвоить значение цвета фона;

2. элементам Z-буфера присвоить минимальное значение типа данных;
3. для каждого треугольника, каждого объекта на сцене:
 - (а) выполнить растровую развертку треугольника;
 - (б) для каждого пикселя, принадлежащего грани, сравнить координату Z с соответствующим значением, хранящемся в Z-буфере:
 - i. если больше:
 - А. в буфер кадра записать атрибуты рассматриваемого пикселя (по соответствующим X, Y);
 - В. записать значение координаты Z рассматриваемого пикселя в Z-буфер (по соответствующим X, Y);
4. отобразить полученное изображение.

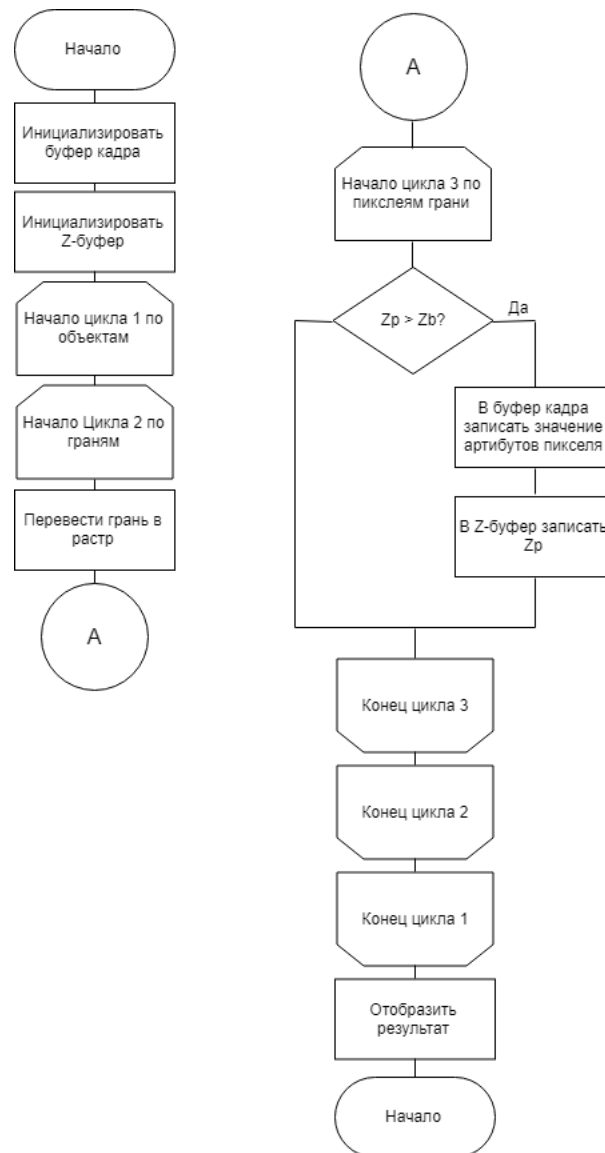


Рисунок 2.3. Схема алгоритма, использующего Z-буфер.

2.5 Растровая развертка треугольника

Пусть треугольник задается тремя точками A, B, C .

Описание алгоритма:

1. найти единичный вектор движения вдоль отрезка AB , как $\frac{\overline{AB}}{|\overline{AB}|}$;
2. найти единичный вектор движения вдоль отрезка AC , как $\frac{\overline{AC}}{|\overline{AC}|}$;
3. инициализировать начальную точку для движения вдоль AB , $T1 = A$;
4. инициализировать начальную точку для движения вдоль AC , $T2 = A$;
5. пока точка $T1 \neq B$ и $T2 \neq C$
 - (a) перевести в растр отрезок $T1T2$;
 - (b) сместить $T1$ на $\frac{\overline{AB}}{|\overline{AB}|}$, $T2$ на $\frac{\overline{AC}}{|\overline{AC}|}$;
6. отобразить полученное изображение.

2.6 Обратная трассировка лучей

Описание алгоритма:

1. для каждого пикселя экрана:
 - (a) задать направление пускаемого луча, как $\frac{\overline{VP}}{|\overline{VP}|}$, V - точка наблюдателя, P - текущая точка экрана
 - (b) для каждого объекта сцены:
 - i. найти пересечения со сферой, описанной вокруг объекта;
 - ii. если пересечение есть, найти точку пересечения с каждой гранью этого объекта;
 - iii. если точка есть:
 - A. рассчитать освещенность найденной точки;
 - B. если не достигнута максимальная глубина рекурсии, рекурсивно пустить отраженный луч и прибавить найденное значение к текущему цвету пикселя;
 - iv. иначе установить цвет пикселя равному цвету фона;
 - (c) закрасить текущую точку экрана соответствуя найденному значению;

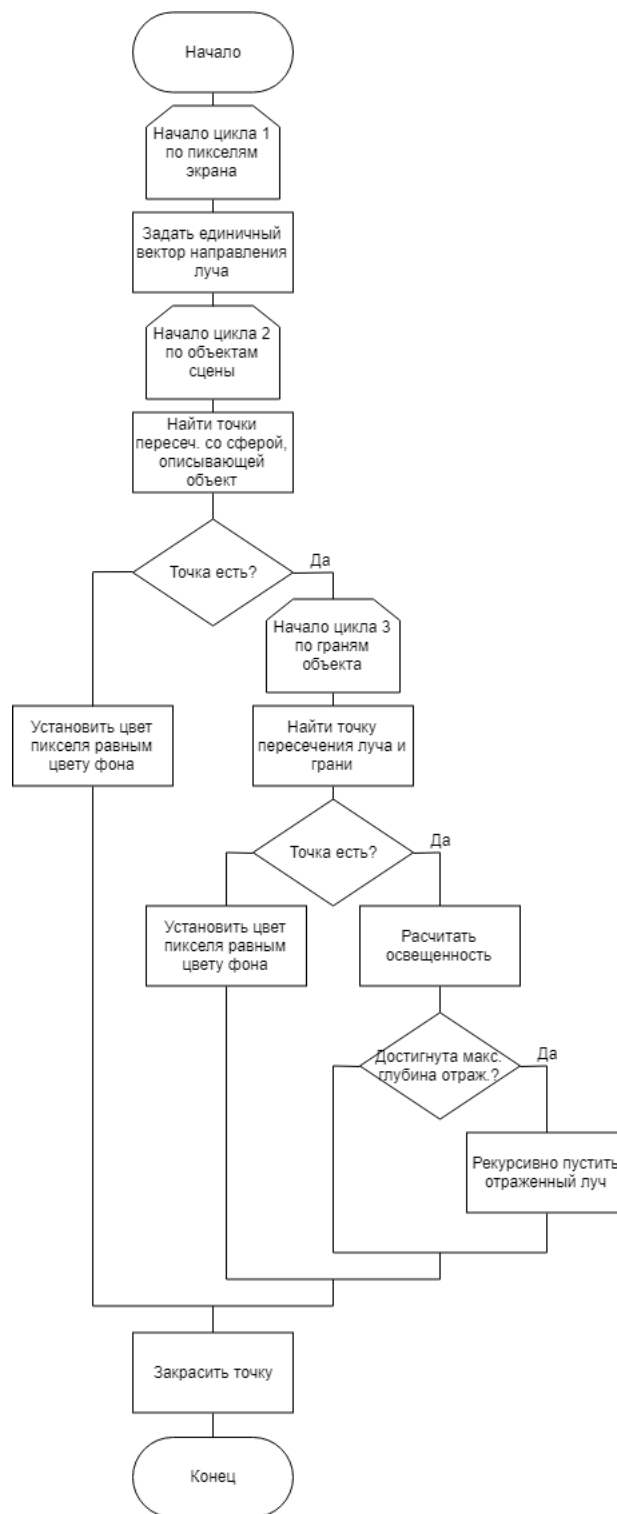


Рисунок 2.5. Схема алгоритма обратной трассировки лучей.

2.7 Поиск пересечения луча со сферой

Луч задается уравнением:

$$\bar{r} = \bar{o} + t * \bar{d} \quad (2.6)$$

$$t \in R, t \geq 0$$

\bar{o} - начальная точка луча

\bar{d} - направление луча

Координаты этого луча можно записать как:

$$X = X_o + t * X_d \quad (2.7)$$

$$Y = Y_o + t * Y_d \quad (2.8)$$

$$Z = Z_o + t * Z_d \quad (2.9)$$

Сфера задается уравнением:

$$(X - X_c)^2 + (Y - Y_c)^2 + (Z - Z_c)^2 = R^2 \quad (2.10)$$

Для того, чтобы найти пересечение луча со сферой необходимо в уравнение сферы подставить соответствующие координаты луча.

После подстановки и несложных преобразований можно получить:

$$t^2 + B * t + C = 0 \quad (2.11)$$

Где,

$$B = 2(X_d * (X_o - X_c) + Y_d * (Y_o - Y_c) + Z_d * (Z_o - Z_c)) \quad (2.12)$$

$$C = (X_o - X_c)^2 + (Y_o - Y_c)^2 + (Z_o - Z_c)^2 - R^2 \quad (2.13)$$

Если дискриминант этого уравнения меньше нуля, тогда луч не пересекается со сферой. Иначе найденные корни будут являться параметрами, которые необходимо подставить в уравнение (2.6) для того, чтобы найти необходимые точки пересечения.

При написании данной работы поиск пересечения со сферой необходим для оптимизации работы обратной трассировки лучей. В этом случае нас интересует только факт того, есть пересечение или нет, соответственно достаточно лишь проверить знак дискриминанта.

2.8 Поиск пересечения луча с треугольником

Пусть луч задается так же, как и при поиске пересечения со сферой (2.6).

Тогда для поиска пересечения с треугольником, изначально необходимо найти точку пересечения этого луча с плоскостью, образованной этим треугольником.

Точка \bar{p} лежит в плоскости, если $(\bar{p} - \bar{q}) * \bar{n} = 0$.

\bar{n} - нормаль к плоскости.

Если подставить вместо \bar{r} координаты луча и выразить параметр, то можно получить:

$$t = \frac{(\bar{q} - \bar{o}) * \bar{n}}{\bar{d} * \bar{n}} \quad (2.14)$$

Если $t < 0$, то точки пересечения нет, иначе $\bar{r} = \bar{o} + t * \bar{d}$ искомая точка пересечения.

Далее необходимо определить, принадлежит ли найденная точка, если таковая имеется, треугольнику.

Пусть треугольник задается тремя точками A, B, C , а найденную точку назовем P .

Тогда, если результат трех векторных произведений $\overline{AB} \times \overline{AP}$, $\overline{BC} \times \overline{BP}$ и $\overline{CA} \times \overline{CP}$ будут сонаправлены, то точка P лежит внутри треугольника ABC . Сонаправленность можно проверить путем сравнения знака соответствующих координат полученных векторов.

2.9 Поиск отраженного луча

Отраженный луч ищется как:

$$\bar{r} = \bar{d} - 2\bar{n} * (\bar{d} * \bar{n}) \quad (2.15)$$

\bar{d} - направление падающего луча

\bar{n} - нормаль в точке падения

Вывод

В данном разделе описаны принципы работы всех рассматриваемых алгоритмов, а также необходимые для расчетов математические соотношения.

3 | Технологическая часть

3.1 Выбор языка и среды программирования

Для реализации данного программного продукта был выбран язык C++ [7], так как имеется наибольший опыт работы с данным языком.

Также была выбрана объектно-ориентированная технология из-за её гибкости и из-за того что она предоставляет возможность вносить правки в уже написанный код без необходимости переписывания больших его частей.

Для реализации интерфейса был выбран фреймворк QT [8], так как он предоставляет большой спектр возможностей для создания сложных интерфейсов.

C++ предоставляет доступ к библиотекам для параллельного программирования. В данной работе будут использованы `<thread>` [9] для создания параллельных потоков и `<mutex>` [10] для ограничения доступа к ресурсам.

3.2 Замеры времени

Замер времени работы алгоритмов производился при помощи функций `clock()` из библиотеки `time.h`. [11]

3.3 Интерфейс программы

На Рисунке. 3.1. представлен пользовательский интерфейс программы. Слева - инструментарий для редактирования сцены, справа синтезированное изображение.

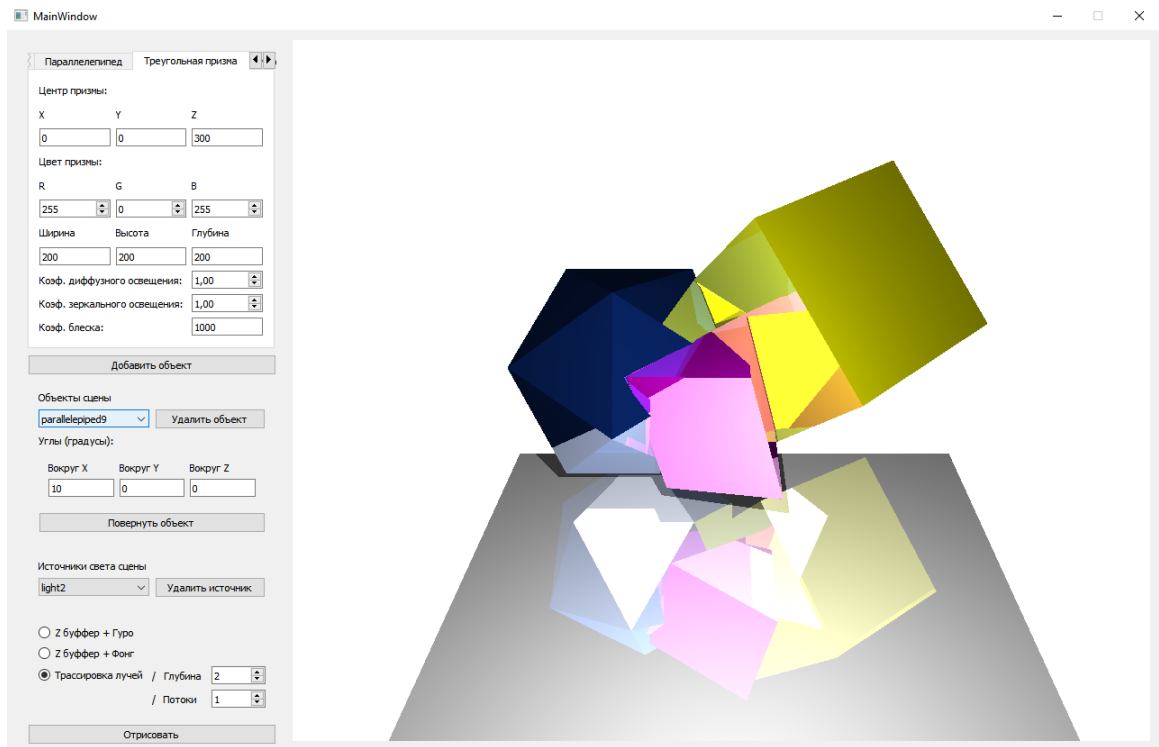


Рисунок. 3.1. Интерфейс программы

На Рисунке. 3.2. - Рисунок. 3.4. изображены окна добавления объектов на сцену. Каждое окно имеет поля для ввода:

1. центра фигуры;
2. цвета фигуры;
3. коэффициента диффузного и зеркально освещения;
4. коэффициент блеска;
5. параметры, необходимые для задания размеров объекта (Радиус для икосаэдра, Ширина, Высота и Глубина для параллелепипеда и прямой треугольной призмы)

На Рисунке. 3.5. изображено окно добавления источника света. Данное окно обладает полями для ввода координаты источника и его интенсивности.

Переключение между окнами происходит при помощи стрелок, расположенных в правом верхнем углу.

Рисунок. 3.2. Окно добавление икосаэдра

Рисунок. 3.4. Окно добавление трехгранной прямой призмы

Рисунок. 3.3. Окно добавление параллелепипеда

Рисунок. 3.5. Окно добавление источника света

На Рисунок. 3.6. изображены поля, предоставляющие доступ для изменения и удаления объектов, уже находящихся на сцене.

Все созданные объекты добавляются в список, далее можно выбрать объект из этого списка и удалить или повращать его.

Источники света добавляются в отдельный список, их можно только удалять.

Объекты сцены

paralleliped9 ▼ Удалить объект

Углы (градусы):

Вокруг X Вокруг Y Вокруг Z

10 0 0

Повернуть объект

Источники света сцены

light2 ▼ Удалить источник

Рисунок. 3.6. Поля для изменения созданных объектов

На Рисунке. 3.7 изображены поля для выбора способа синтеза изображения.

Поле Глубина отвечает за максимальную глубину отражения при поиске отраженных лучей для алгоритма обратной трассировки лучей.

Поле Потоки отвечает за то, на сколько потоков будет делиться процесс при синтезе изображения.

☐ Z буфер + Гуро

☐ Z буфер + Фонг

☒ Трассировка лучей / Глубина 2

/ Потоки 1

Отрисовать

Рисунок. 3.7. Поля для выбора способа синтеза изображения

Вывод

В данном разделе обоснован выбор языка программирования, описаны выбранные средства для создания интерфейса, а так как же описано строение пользовательского интерфейса.

4 | Экспериментальная часть

4.1 Цель эксперимента

Целью эксперимента является временное сравнение работы реализованных алгоритмов.

4.2 Апробация

На Рисунке 4.1. представлена сцена, содержащая куб, икосаэдр, трехгранную призму и один точечный источник. Это изображение сгенерировано обратной трассировкой лучей.

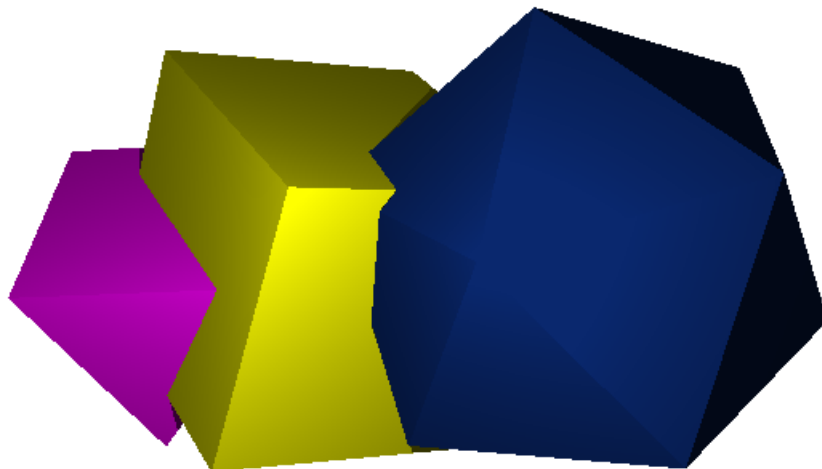


Рисунок. 4.1. Сцена, сгенерированная обратной трассировкой лучей.

На Рисунке 4.2. изображена та же самая сцена, сгенерированная обратной трассировкой, но с дополнительным вычислением отражений.

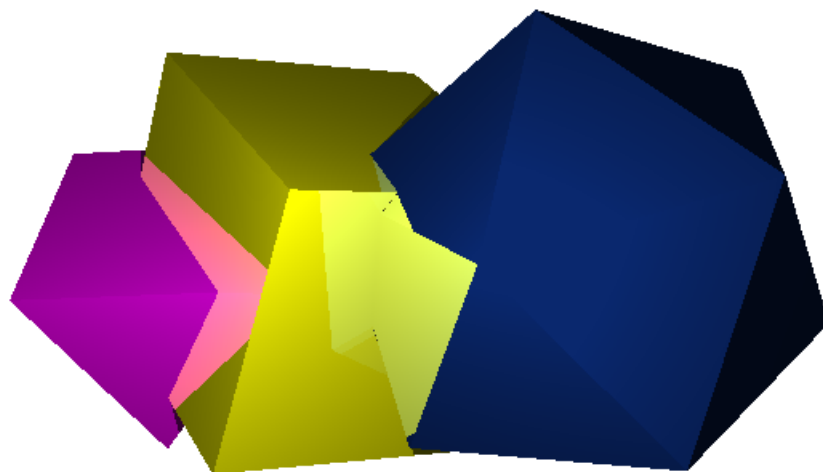


Рисунок. 4.2. Сцена, сгенерированная обратной трассировкой лучей с дополнительным вычислением отражений.

На Рисунке 4.3. показана сцена, сгенерированная с использованием Z-буфера и метода Фонга.

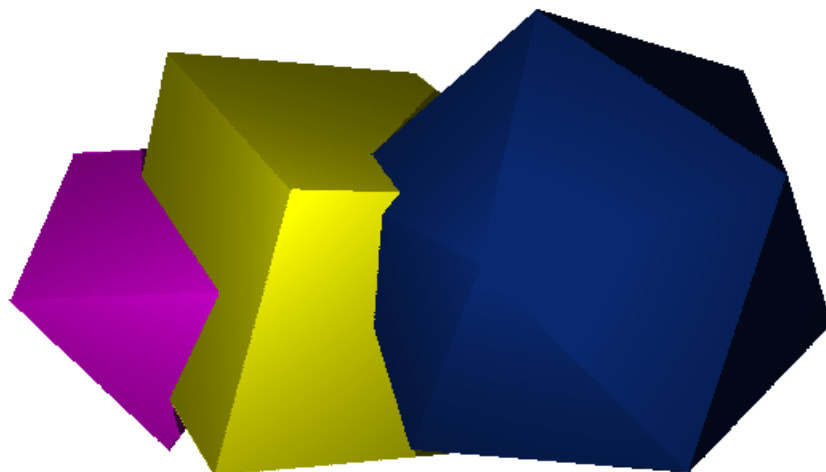


Рисунок. 4.3. Сцена, сгенерированная с использованием Z-буфера и метода Фонга.

На Рисунке 4.4. изображена сцена, сгенерированная с использованием Z-буфера и метода Гуро.

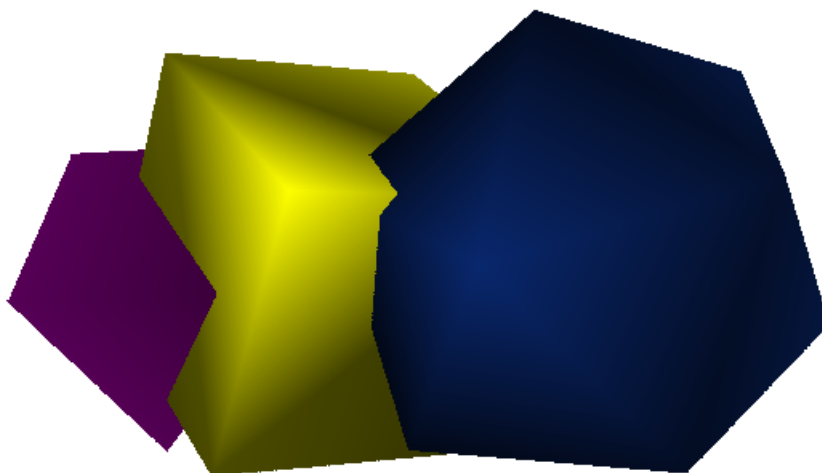


Рисунок. 4.4. Сцена, сгенерированная с использованием Z-буфера и метода Гуро.

4.3 Описание эксперимента

Суть эксперимента заключается в том, чтобы замерить скорость работы алгоритмов в двух различных случаях.

Первый случай - когда суммарное количество граней объектов на сцене не изменяется, увеличивается лишь суммарная площадь этих граней. В теории этот случай должен быть благоприятен для использования трассировки лучей.

Второй - обратный первому, когда увеличивается суммарное количество граней при неизменной площади. В теории этот случай должен быть благоприятен для использования Z-буфера с Фонгом или же с Гуро.

Все эксперименты проведены в однопоточном режиме.

Эксперимент проводился на следующей системе:

1. Intel(R) Core(TM) i7-3770K
2. 8.00 ГБ ОЗУ

4.3.1 Первый случай

В данном случае на сцену добавлен один объект - параллелепипед, содержащий 12 треугольников - граней. Во время эксперимента изменялись длины ребер, в следствии чего менялась площадь объекта.

Новых объектов не добавлялось, соответственно суммарное количество граней не изменялось.

Таблица. 4.1. Времена работы алгоритмов при различных суммарных площадях.

Площадь (px)	Z+gourand (c)	Z+phong (c)	RayTrace (c)
160000	0.21	0.51	1.261
320000	0.261	0.555	1.271
560000	0.306	0.718	1.356
880000	0.411	0.904	1.514
1280000	0.573	1.128	1.684
2480000	1.526	2.149	1.972
3280000	2.688	2.934	2.073
3680000	3.166	3.161	2.131
4080000	3.91	3.669	2.132

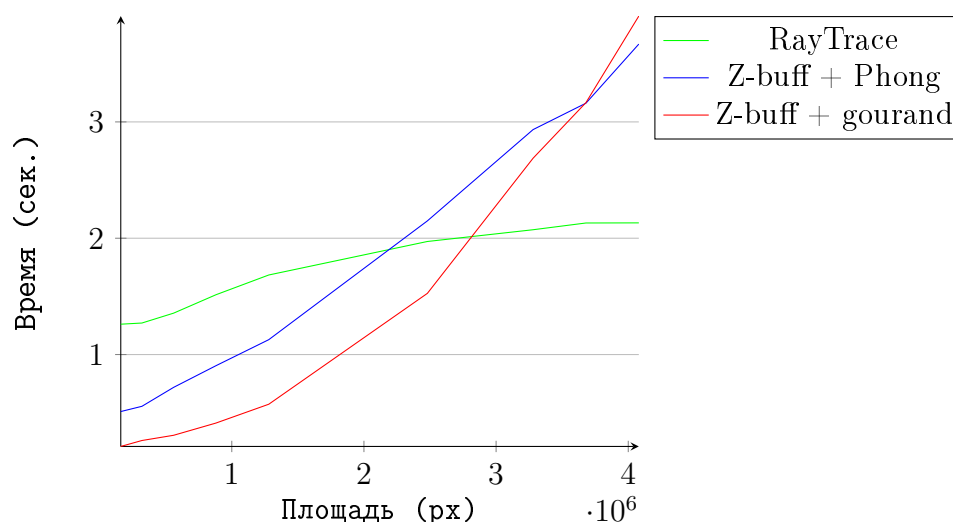


Рисунок. 4.5. График сравнения алгоритмов при различных суммарных площадях. (Таблица 4.1.)

Из Рисунка 4.5. видно, как при сильном увеличении суммарной площади без увеличения количества граней на сцене, трассировка лучей начинает выигрывать у Z-буфера с Гуро и Z-буфера с Фонгом. (Рис. 4.5.)

Это связано с тем, что алгоритм Z-буфера переводит в растр все грани каждого объекта, соответственно при увеличении площади увеличивается и трудоемкость.

В то же время обратная трассировка лучей лишь находит точку пересечения с гранью и никак не зависит от её площади.

Из этого эксперимента можно сделать вывод, что обратную трассировку лучей выгоднее использовать для синтеза сцены, на которой малое количество граней и их суммарная площадь очень велика.

По данным Таблицы 4.1. видно, что трассировка начала работать быстрее только при суммарной площади равной 3280000 px. При таком значении трассировка быстрее алгоритмов с Z-буфером примерно в 1.45 раз. Также можно заметить, что при площади равной 3680000 px скорость работы алгоритмов Z-буфера с закраской по Фонгу и по Гуро сравнялись, а при площади равной 4080000 px алгоритм Z-буфера с закраской по методу Гуро начал работать на 6% медленнее Z-буфера с Фонгом.

Вывод по первому случаю: в данном случае лучшее время работы показала обратная трассировка лучей, худшее - Z-буфер с методом закраски по Гуро.

4.3.2 Второй случай

В данном случае площадь зафиксированна в значении 60000 рх. На этот раз добавляются новые объекты, тем самым увеличивая суммарное количество граней.

Таблица. 4.2. Времена работы алгоритмов при увеличении количества граней.

Кол-во граней	Z+gourand (с)	Z+phong (с)	RayTrace (с)
24	0.348	0.245	1.675
48	0.242	0.485	3.095
60	0.247	0.593	3.793
72	0.25	0.702	4.609
84	0.259	0.751	5.155

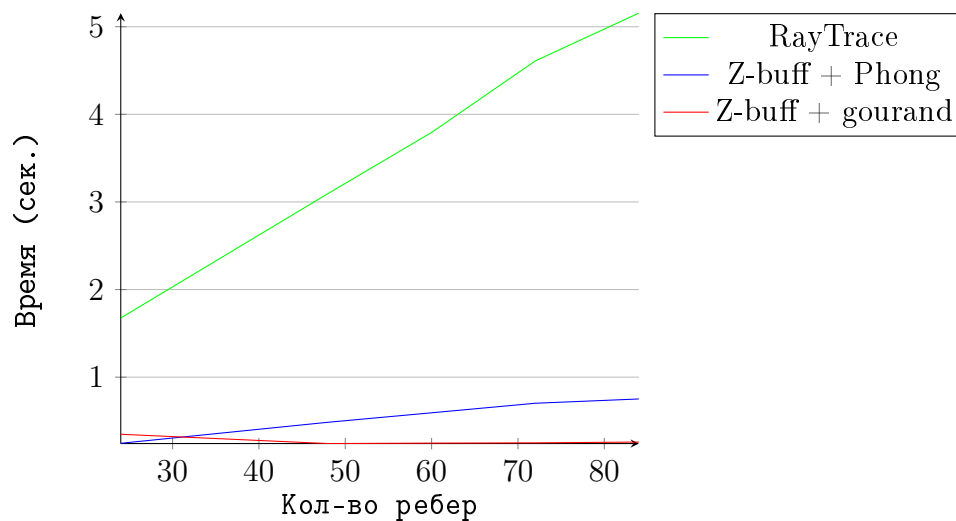


Рисунок. 4.6. График сравнения алгоритмов при увеличении количества граней.

Из Рисунка 4.6. видно, что при большом количестве граней трассировка лучей начинает проигрывать алгоритмам с Z-буфером. Согласно Таблице 4.2., при 24 гранях трассировка работает в 6 раз медленнее Z-буфера с Фонгом и в 4 раза медленнее Z-буфера с Гуро. При 84 гранях разница составляет уже 6,8 раз и 19 раз соответственно. При 84 гранях Z-буфер с Фонгом работает в 2.9 раза быстрее чем Z-буфер с Гуро.

Вывод по второму случаю: в данном случае лучшее время работы показал метод Z-буфера с закраской по Гуро, худшее - обратная трассировка лучшей.

Вывод

В данном разделе проведен эксперимент, в котором рассматривается два случая.

В первом случае, при увеличении суммарной площади граней всех объектов на сцене, показано, что обратная трассировка лучей демонстрирует наилучший результат, по сравнению с алгоритмами использующими Z-буфер, при превышении суммарной площади значения, равного 3280000 рх.

Также замечено, что при превышении суммарной площади значения, равного 3680000 рх Z-буфер с методом Гуро начал уступать по времени Z-буферу с методом Фонга. При 4080000 рх разница достигла 6%.

Во втором случае, при увеличении количества граней с неизменной их суммарной площадью, показано, что алгоритм Z-буфера с закраской по Гуро демонстрирует наилучший результат, по сравнению с остальными рассматриваемыми алгоритмами. В среднем, Z-буфер с закраской по Гуро работает в 2 раза быстрее Z-буфера с закраской по Фонгу, а так же в 12 раз быстрее обратной трассировки лучей.

Исходя из данных полученных в этом разделе, можно сделать вывод, что:

1. алгоритм обратной трассировки лучей является наиболее быстрым, в случае наличия на сцене объектов с малым количеством граней, но большой площадью;
2. алгоритм Z-буфера с методом закраски Гуро является наиболее быстрым, в случае наличия на сцене объектов с большим количеством граней и малой площадью;
3. изображения, синтезируемые методом Z-буфера с закраской по Фонгу и обратной трассировки лучей, практически не отличаются визуально;
4. отражения в объектах позволяет строить только метод обратной трассировки лучей.

Заключение

В рамках данной работы проанализированы рассматриваемые алгоритмы компьютерной графики для построения реалистичных изображений. Был выбран оптимальный формат для хранения информации об объектах. Также были получены и описаны все необходимые, для решения поставленной задачи, математические соотношения.

В результате был написан программный продукт для генерации реалистичного изображения и проведет анализ рассматриваемых алгоритмов. Анализ проводился на основе проведенного эксперимента, в котором рассматривались два возможных случая.

Первый случай, при котором синтезируемая сцена содержит малое количество объектов, следовательно и малое количество граней, но суммарная площадь всех этих граней очень велика. В данном случае лучшее время работы показала обратная трассировка лучей, худшее - Z-буфер с методом закраски по Гуро.

Второй случай, при котором синтезируемая сцена содержит объекты с большим количеством граней, но суммарная площадь этих граней мала. В данном случае лучшее время работы показал метод Z-буфера с закраской по Гуро, худшее - обратная трассировка лучей.

В результате проведенного анализа, получено:

1. алгоритм обратной трассировки лучей является наиболее быстрым, в случае наличия на сцене объектов с малым количеством граней, но большой площадью;
2. алгоритм Z-буфера с методом закраски Гуро является наиболее быстрым, в случае наличия на сцене объектов с большим количеством граней и малой площадью;
3. изображения, синтезируемые методом Z-буфера с закраской по Фонгу и обратной трассировки лучей, практически не отличаются визуально;
4. отражения в объектах позволяет строить только метод обратной трассировки лучей.

Литература

1. Косников Ю.Н. Поверхностные модели в системах трехмерной компьютерной графики. Учебное пособие. – Пенза: Пензенский государственный университет, 2008. – 60 с.
2. Роджерс, Д.Ф. Алгоритмические основы машинной графики / Пер. с англ. С.А. Вичеса, Г.В. Олохтоновой, П.А. Монахова. – М.: Мир: 1989. – 512 с. – ISBN 5-03-000476-9;
3. Закраска методом Гуро. [Электронный ресурс]
Режим доступа: <http://www.mari-el.ru/mmlab/home/kg/Lecture11/3.html> Последняя дата обращения: 25.12.2019
4. Особенности моделирования света: Аппроксимации Фонга (Phong Shading). [Электронный ресурс]
Режим доступа: <https://www.ixbt.com/video/light-model-phong.html> Последняя дата обращения: 25.12.2019
5. Закон Ламберта. Модель отражения Фонга. Модель отражения Блинна-Фонга. [Электронный ресурс]
Режим доступа: https://compgraphics.info/3D/lighting/phong_reflection_model.php Последняя дата обращения: 25.12.2019
6. Вельтмандер П.В. Основные алгоритмы компьютерной графики. Учебное пособие в 3-х кн. – Новосибирск: НГУ, 1997. -197 с.
7. Метод трассировки лучей против растеризации: новое поколение качества графики? [Электронный ресурс]
Режим доступа: http://www.thg.ru/graphic/ray_tracing_rasterization/onepage.html Последняя дата обращения: 25.12.2019
8. Стандарт языка C++11 согласно ISO. [Электронный ресурс]
Режим доступа: <https://isocpp.org/std/the-standard> Последняя дата обращения: 25.12.2019
9. Qt Documentation. [Электронный ресурс]
Режим доступа: <https://doc.qt.io/> Последняя дата обращения: 25.12.2019
10. std::thread [Электронный ресурс]
Режим доступа: <https://ru.cppreference.com/w/cpp/thread/thread> Последняя дата обращения: 25.12.2019

11. `std::mutex` [Электронный ресурс]

Режим доступа: <https://ru.cppreference.com/w/cpp/thread/mutex> Последняя дата обращения: 25.12.2019

12. `<ctime>` (`time.h`) [Электронный ресурс]

Режим доступа: <http://www.cplusplus.com/reference/ctime/> Последняя дата обращения: 25.12.2019