# Оглавление

Введение

1. Конструкторская часть

Трассировка лучей

Построение теней

- Сплошные тени

- Реалистичные тени

Математические и физические предпосылки алгоритма обратной трассировки

- Освещение

- Схема расчета интенсивности. Параметры, задающие свойства тел.

Вычисление нормалей

Вычисление отраженного луча

Вычисление преломленного луча

Вычисление точки пересечения с примитивом

Описание типов данных. Структура программы

Краткое описание структур и классов

2. Технологическая часть

Выбор языка программирования

Описание интерфейса программы

3. Исследовательская часть

Зависимость времени построения от глубины рекурсии

Зависимость времени построения от количества источников

Выводы по результатам исследований

Заключение

Список литературы

**Введение**

Эта программа посвящена генерации реалистичных изображений, в частности – реалистичному изображению стекла.

Существует несколько методов генерации реалистичных изображений, таких как прямая трассировка лучей (трассировка фотонов), обратная трассировка лучей, radiosity.

Методы трассировки лучей на сегодняшний день считаются наиболее мощными и универсальными методами создания реалистичных изображений. Известно много примеров реализации алгоритмов трассировки для качественного отображения самых сложных трехмерных сцен. Можно отметить, что универсальность методов трассировки в значительной степени обусловлена тем, что в их основе лежат простые и ясные понятия, отражающие наш опыт восприятия окружающего мира.

Окружающие нас объекты обладают по отношению к свету такими свойствами:

- излучают;

- отражают и поглощают;

- пропускают сквозь себя.

Каждое из этих свойств можно описать некоторым набором характеристик.

Излучение можно охарактеризовать интенсивностью и спектром.

Свойство отражения (поглощения) можно описать характеристиками диффузного рассеивания и зеркального отражения. Прозрачность можно описать ослаблением интенсивности и преломлением.

Из точек поверхности (объема) излучающих объектов исходят лучи света. Можно назвать такие лучи первичными – они освещают все остальное. От источников излучения исходит по различным направлениям бесчисленное множество первичных лучей. Некоторые лучи уходят в свободное пространство, а некоторые попадают на другие объекты. Если луч попадает в прозрачный объект, то, преломляясь, он идет дальше, при этом некоторая часть световой энергии поглощается.

В результате действия на объекты первичных лучей возникают вторичные лучи. Некоторые из них попадают на другие объекты. Так, многократно отражаясь и преломляясь, отдельные световые лучи приходят в точку наблюдения. Таким образом, изображение сцены формируется некоторым множеством световых лучей.

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

Непосредственная реализация данной лучевой модели формирования изображения представляется затруднительной. Можно попробовать построить алгоритм построения изображения указанным способом. В таком алгоритме необходимо предусмотреть перебор всех первичных лучей и определить те из них, которые попадают в объекты и в камеру. Затем выполнить перебор всех вторичных лучей, и также учесть только те, которые попадают в объекты и в камеру. И так далее. Такой алгоритм называется прямой трассировкой лучей. Главный недостаток этого метода – много лишних операций, связанных с расчетом лучей, которые затем не используются.

*Обратная трассировка лучей.*

Именно этому методу генерации реалистичных изображений посвящена эта работа.

Метод обратной трассировки лучей позволяет значительно сократить перебор световых лучей. Метод разработан в 80-х годах, основополагающими считаются работы Уиттеда и Кэя. Согласно этому методу отслеживание лучей производится не от источников света, а в обратном направлении – от точки наблюдения. Так учитываются только те лучи, которые вносят вклад в формирование изображения.

Плоскость проецирования разбита на множество пикселов. Выберем центральную проекцию с центром схода на некотором расстоянии от плоскости проецирования. Проведем прямую линию из центра схода 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.

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 одному отраженному лучу соответствует только один 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.

При практической реализации метода обратной трассировки вводят ограничения. Некоторые из них необходимы, чтобы можно было в принципе решить задачу синтеза изображения, а некоторые ограничения позволяют значительно повысить быстродействие трассировки.

*Ограничения при реализации трассировки.*

1. Среди всех типов объектов выделим некоторые, которые назовем источниками света. Источники света могут только излучать свет, но не могут его отражать или преломлять. Будем рассматривать только точечные источники света.
2. Свойства отражающих поверхностей описываются суммой двух составляющих – диффузной и зеркальной.
3. В свою очередь, зеркальность также описывается двумя составляющими. Первая (reflection) учитывает отражение от других объектов, не являющихся источниками света. Строится только один зеркально отраженный луч r для дальнейшей трассировки. Вторая компонента (specular) означает световые блики от источников света. Для этого направляются лучи на все источники света и определяются углы, образуемые этими лучами с зеркально отраженным лучом обратной трассировки (r). При зеркальном отражении цвет точки поверхности определяется собственным цветом того, что отражается.
4. При диффузном отражении учитываются только лучи от источников света. Лучи от зеркально отражающих поверхностей ИГНОРИРУЮТСЯ. Если луч, направленный на данный источник света, закрывается другим объектом, значит, данная точка объекта находится в тени. При диффузном отражении цвет освещенной точки поверхности определяется собственным цветом поверхности и цветом источников света.
5. Для прозрачных (transparent) объектов не учитывается зависимость коэффициента преломления от длины волны. (Иногда прозрачность вообще моделируют без преломления, то есть направление преломленного луча t совпадает с направлением падающего луча.)
6. Для учета освещенности объектов светом, рассеянным другими объектами, вводится фоновая составляющая (ambient).
7. Для завершения трассировки вводится ограничение количества итераций (глубины рекурсии).

*RADIOSITY (ИЗЛУЧАТЕЛЬНОСТЬ).*

Кроме вышеперечисленных методов генерации реалистичных изображений существует еще один метод. Он называется radiosity. Этот метод рассматривает распространение в пространстве световых волн. Это более сложный метод, более точно учитывающий законы распространения световой энергии и требующий больших затрат времени и аппаратных ресурсов. С помощью этого метода реализуются не только такие явления, как зеркальное отражение и преломление, но и более сложные явления, например, диффузное отражение и диффузное преломление.

Сцену можно представить как набор поверхностей, обменивающихся световой энергией. Большинство реальных поверхностей является диффузными отражателями, когда падающий луч отражается или рассеивается во всех направлениях полусферы, находящейся над отражающей поверхностью.

Особый здесь случай - отражение Ламберта (идеальная диффузия). Метод излучательности описывает баланс энергетического равновесия в замкнутой системе.

Предполагается, что поверхности идеально диффузны, т.е. после отражения падающий луч пропадает. Излучательность отдельной поверхности включает самоизлучение и отраженный или пропущенный свет.

*Выводы по методу обратной трассировки.*

Положительные черты:

1. Универсальность метода, его применимость для синтеза изображений достаточно сложных пространственных схем. Воплощает многие законы геометрической оптики. Просто реализуются разнообразные проекции.
2. Даже усеченные варианты данного метода позволяют получить достаточно реалистичные изображения. Например, если ограничиться только первичными лучами (из точки проецирования), то это дает удаление невидимых точек. Трассировка уже одного-двух вторичных лучей дает тени, зеркальность прозрачность.
3. Все преобразования координат линейны, поэтому достаточно просто работать с текстурами.

Недостатки:

1. Проблемы с моделированием диффузного отражения и преломления.
2. Для каждой точки изображения необходимо выполнять много вычислительных операций. Трассировка относится к числу самых медленных алгоритмов синтеза изображений.

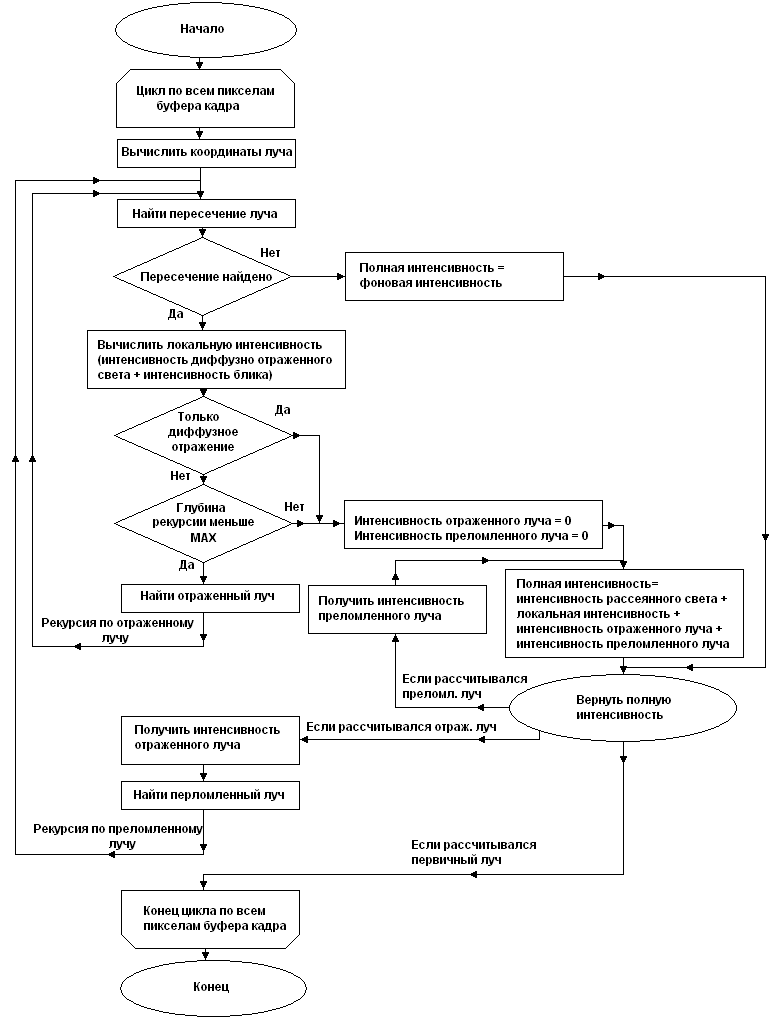
# 2. Конструкторская часть

## 

## Алгоритмы

### 

### Обратная трассировка лучей



**Рис. 2.1.1.** *Блок-схема рекуррентного алгоритма обратной трассировки лучей.*

В этой программе алгоритм обратной трассировки реализован рекуррентным образом. Функция расчета интенсивности первичного луча рекуррентно вызывает саму себя для нахождения интенсивностей отраженного и преломленного лучей.

Алгоритм:

Для расчета цвета каждого пиксела буфера кадра выполняются следующие действия:

1. Найти координаты пиксела в мировой системе координат.

2. Найти координаты первичного луча.

3. Запуск функции вычисления интенсивности первичного луча.

4. Найти пересечения луча со всеми примитивами сцены и выбрать ближайшее.

5. Если пересечение не найдено, значит, луч ушел в свободное пространство.

Для расчета цвета принимаем полную интенсивность равной фоновой интенсивности. Перейти на шаг 12. Если пересечение найдено, перейти на шаг 6.

6. Рассчитываем «локальную» интенсивность цвета объекта, которому принадлежит точка пересечения. Под «локальной» интенсивностью понимается интенсивность с учетом интенсивности диффузно отраженного света и интенсивность бликов.

7. Если материал отражает свет только диффузно, то считаем интенсивности отраженного и преломленного света нулевыми. Перейти на шаг 12. Иначе перейти на шаг 8.

8. Если достигнута максимальная глубина рекурсии, то принять интенсивности отраженного и преломленного света нулевыми. Перейти на шаг 12. Иначе перейти на шаг 9.

9. Вычислить вектор отраженного луча. Запуск рекурсии для нахождения интенсивности отраженного луча.

10. Вычислить вектор преломленного луча. Запуск рекурсии для нахождения интенсивности преломленного луча.

11. Вычисление полной интенсивности цвета. Полная интенсивность включает в себя интенсивность рассеянного света, локальную интенсивность и интенсивности отраженного и преломленного лучей.

12. Возврат в точку вызова функции вычисления интенсивности луча.

Если шел расчет первичного луча, то в буфер кадра помещается пиксел вычисленного цвета. Переходим к расчету следующего пиксела буфера кадра Если шел расчет отраженного (преломленного) луча, то вычисленная интенсивность будет принята как интенсивность отраженного (преломленного) луча на предыдущем шаге рекурсии.

Построение теней

**СПЛОШНЫЕ ТЕНИ**

Для построения сплошных теней в алгоритме трассировки на этапе вычисления «локальной» интенсивности цвета в точке объекта проверяется «видимость» каждого источника света из этой точки.



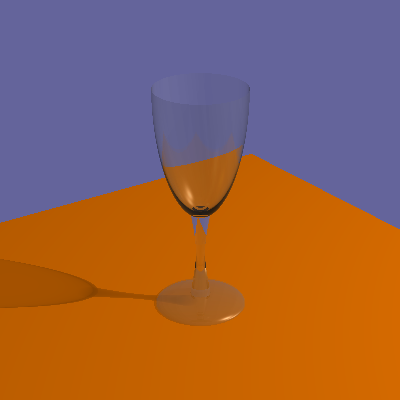
Принцип работы алгоритма.

1. Из проверяемой точки строится луч, направленный на источник света.
2. Производится поиск пересечений этого луча с примитивами сцены между проверяемой точкой и источником.
3. Если найдено хотя бы одно пересечение, то проверяемая точка находится в тени. При расчете ее цвета источник, для которого проводилась проверка, не учитывается.
4. Если пересечений не найдено, точка не в тени. При расчете ее цвета учитываем проверяемый источник.

Такой метод нахождения теней дает приемлемый результат до тех пор, пока на сцене нет прозрачных объектов. Однако сплошная черная тень от стакана выглядит не реалистично. Стекло частично пропускает свет, поэтому интенсивность заслоненного источника должна учитываться при подсчете интенсивности света в точке объекта, но она должна ослабляться при проходе света сквозь стекло.

**РЕАЛИСТИЧНЫЕ ТЕНИ**

В этой программе предлагается способ построения теней, хотя и не соответствующий физике процесса образования тени, но, тем не менее, дающий более реалистичные результаты, чем обыкновенные сплошные тени. В этом алгоритме ослабление интенсивности света, дошедшего до проверяемой точки, привязано к расстоянию, пройденному светом в материале (в стекле). При этом преломлением света полностью пренебрегаем.



Алгоритм работает следующим образом.

1. Из проверяемой точки строится луч, направленный на источник света.
2. Находятся ВСЕ пересечения этого луча с примитивами сцены между проверяемой точкой и источником и сохраняются в массив пересечений. Первым элементом в массив заносится 0 (сама точка, для которой истроится тень). Одновременно с этим формируется массив, в котором хранятся данные о том, какому объекту сцены принадлежит найденная точка пересечения.
3. Массив пересечений сортируется по возрастанию расстояния от проверяемой точки до точки пересечения. При этом абсолютно аналогичные операции проводятся и со вторым массивом.
4. Если в массив не была добавлена ни одна точка кроме начальной, то проверяемая точка не затенена. Берется полная интенсивность источника. В противном случае начинаем проход массива со второй точки по всем точкам.
5. Если объект, которому принадлежит текущая точка пересечения, абсолютно непрозрачный, то имеет место сплошная тень, цикл прерываем. Иначе проверяем, входит луч в объект или выходит из него. Если входит – переход в следующей точке.
6. Если выходит, то вычисляем расстояние от предыдущей точки до текущей. Это будет расстояние, пройденное светом внутри объекта. В зависимости этого расстояния ослабляем интенсивность источника. Переходим к следующей точке.

Надо заметить, что различные прозрачные объекты могут отбрасывать разную тень даже при одинаковом расстоянии, пройденном светом в объекте. Так цветное стекло должно отбрасывать более темную тень, чем прозрачное. Для учета этого факта для каждого объекта введен специальный параметр - это расстояние, пройденное светом внутри объекта до полного затухания. Чем больше этот параметр, тем светлее тень, отбрасываемая объектом.

## Математические и физические предпосылки алгоритма обратной трассировки лучей

### 

### Освещение

Согласно модели Уиттеда цвет некоторой точки объекта определяется суммарной интенсивностью

I(l)=Ka\*Ia(l)C(l) + Kd\*Id(l)\*C(l) +Ks\*Is(l) + Kr\*Ir(l) +Kt\*It(l),

где l – длина волны, C(l) – заданный исходный цвет точки объекта, Ka, Kd, Ks, Kr и Kt – коэффициенты, учитывающие свойства конкретного объекта параметрами фоновой интенсивности, диффузного рассеивания, зеркальности, отражения и прозрачности.

Ia – интенсивность фоновой подсветки,

Id – интенсивность, учитываемая для диффузного рассеивания,

Is – интенсивность, учитываемая для зеркальности,

Ir – интенсивность излучения, пришедшая по отраженному лучу,

It – интенсивность излучения, пришедшая по преломленному лучу.

Интенсивность фоновой подсветки (Ia) для некоторого объекта обычно константа. Запишем формулы для остальных интенсивностей.

Для диффузного отражения:

Id=∑I(l)cosӨ,

где I(l) – интенсивность излучения i-го источника света, Ө - угол между нормалью к поверхности объекта и направлением на i-й источник света.

Для зеркальности:

Id=∑I(l)cosα,

где p – показатель степени от единицы до нескольких сотен (согласно модели Фонга), α – угол между отраженным лучом (обратной трассировки) и направлением на i-й источник света.

Надо заметить, что в модели Уиттеда в чистом виде есть один существенный недостаток. Дело в том, что доля световой энергии отраженной от поверхности объекта зависит от угла падения и преломления. Например, если смотреть вдоль поверхности стекла, то доля отраженного света повысится, а преломленного понизится. Но в модели Уиттеда коэффициенты, отражающие доли интенсивности отраженного и преломленного света в суммарной интенсивности точки поверхности объекта задаются константами (не зависят от угла). В большинстве случаев это дает приемлемый результат. Но при моделировании стекла мы встречаемся с эффектом полного внутреннего отражения. Некоторые лучи света не могут выйти из толщи стекла за заданное фиксированное число итераций. В результате этого по краям стакана возникает четкая черная полоса. Если учесть зависимость интенсивности отраженного и преломленного света от углов падения и преломления, то эффект будет намного более реалистичным. Края черной области размоются.

Для вычисления коэффициента при интенсивности отраженного луча используется следующая формула:

r = 0.5(cos α - Ncos β)2 /(cos α + Ncos β)2 + 0.5(Ncos α – cos β)2 /(Ncos α + cos β)2,

где α – угол падения луча, β – угол преломления луча, N – относительный показатель преломления двух сред.

Коэффициент при интенсивности преломленного луча вычисляется по формуле:

t = 1 – r.

**Схема рассчета интенсивности. параметры, задающие свойства тел**

В связи с недостатками модели Уиттеда в этой программе предлагается следующая система вычисления интенсивности света в точке.

Интенсивность света складывается из интенсивности фоновой подсветки сцены, интенсивности диффузно отраженного света источников, интенсивности бликов от источников («локальные» характеристики освещенности), интенсивности зеркально отраженного луча и интенсивности преломленного луча, если таковые имеются.

Интенсивность фоновой подсветки (IA) задается некоторой константой.

Интенсивность диффузно отраженного света (ID) вычисляется по классическому «закону косинуса».

ID = IL cos α,

где IL – интенсивность источника света, α – угол между нормалью к поверхности и направлением на источник.

В случае освещения сцены несколькими источниками Id вычисляется для каждого из них и затем суммируются.

ID***i*** =Σ IL***i*** cos α***i***.

Интенсивность блика от источника (IS) вычисляется в соответствии с моделью Фонга.

S = IL cosp β,

где IL – интенсивность источника света (0<=IL<=1), β – угол между отраженным лучом от источника света и направлением на точку, в которой расположена камера (центр проекции), p – некоторая степень от 1 до 200 –влияет на размытость блика. При маленьких значениях p блик более размытый.

Как и при вычислении ID в случае освещения сцены несколькими источниками IS вычисляется отдельно для каждого источника, а затем результаты суммируются.

S***i*** =Σ IL***i*** cosp β ***i***.

Интенсивности зеркально отраженного (IR) и преломленного (IT) света рассчитываются для отраженного и преломленного лучей на следующем шаге рекурсии. Если достигнут предел глубины рекурсии, то эти интенсивности берутся нулевыми. От интенсивности IR берется r процентов, а от IT – t = 1 – r (см. предыдущий раздел).

Кроме того, вводятся следующие коэффициенты: KD – коэффициент диффузного отражения поверхности, KS – коэффициент блика.

KD – этот коэффициент является характеристикой неровности отражающей поверхности. Чем больше неровность поверхности, тем меньше света отражается от неё зеркально и меньше света она пропускает, и соответственно больше света она отражает диффузно. 0 <= KD <= 1.

При KD = 0 - весь свет, падающий на поверхность, отражается и преломляется. KD = 1 - весь свет отражается диффузно. На этот коэффициент умножаются интенсивность диффузно отраженного света и интенсивность фоновой подсветки. Интенсивности зеркально отраженного и преломленного света умножаются на (1 – KD).

KS – этот коэффициент отвечает за яркость блика от источника. 0<=KS<=1. При KS = 0 - блик не виден, при KS = 1 – яркость блика максимальна.

Таким образом, окончательная формула для расчета интенсивности объекта в какой-либо точке будет выглядеть следующим образом:

I = IAKD + Σ(IL***i***KDcos α***i*** + IL***i***KScosp β ***i***) + (1 – KD )(IRr + IT(1 – r)).

При этом надо заметить, что итоговая интенсивность не должна получиться больше единицы. Если такое происходит, то эта точка изображения будет засвеченной. Ее интенсивность надо сбросить на единицу.

Для получения цветного изображения необходимо провести расчеты отдельно для красной, зеленой и синей компоненты света. Цвет пиксела изображения будет вычисляться путем умножения каждой компоненты интенсивности на число, определяющее максимальное количество градаций интенсивности изображения. Для 32-битного изображения оно равно 255 на каждый из цветов(R,G,B).

R = 255\*IR,

G = 255\*IG,

B = 255\*IB.

Здесь IR (не путать с интенсивностью зеркально отраженного света), IG, IB – интенсивности трех компонент света в точке, полученная по формуле, указанной выше.

Коэффициенты KD, KS, p – это индивидуальные характеристики объекта, отражающие его свойства. Кроме этого имеется еще один коэффициент – абсолютный показатель преломления n. n = c / v, где c – скорость света в вакууме, v – скорость света в среде (внутри объекта). Для абсолютно непрозрачных тел этот коэффициент равен ∞ (т.к. скорость света внутри тела нулевая). В программе для задания абсолютно непрозрачного тела необходимо поставить этот коэффициент >> 1 (порядка 10 000). При этом доля зеркально отраженного света r будет стремиться к единице, а преломленного, соответственно, к нулю.

Вычисление нормалей

В алгоритме трассировки нормали к объектам необходимы для вычисления отраженного и преломленного лучей, а также для определения освещенности согласно модели Фонга.

В этой программе присутствуют три вида примитивов, из которых строится сцена. Это полигон (треугольник), эллипсоид и параболоид. Последние два введены для более реалистичной имитации стакана (его можно было бы построить и из полигонов, но модель получилась бы более грубая).

Вычисление нормали к полигону (Треугольнику).

Вычисление нормали к треугольнику сводится к операции векторного умножения. Пусть задан треугольник ABC координатами трех своих вершин: XA, YA, ZA, XB, YB, ZB, XC, YC, ZC.

Вычислим координаты двух векторов, например AB и AC:

XAB = XB – XA,

YAB = XB – XA,

ZAB = XB – XA,

XAC = XC – XA,

YAC = XC – XA,

ZAC = XC – XA.

Координаты вектора нормали будут вычисляться по формулам:

Xn = YABZAC - YACZAB,

Yn = XABZAC - XACZAB,

Zn = XABYAC - XACYAB.

Нет необходимости вычислять координаты вектора нормали к треугольнику каждый раз в теле трассировки, так как в любой точке треугольника нормали одинаковые. Достаточно их посчитать один раз в инициализирующей части программы и сохранить. При повороте треугольника надо поворачивать и его нормаль.

Вычисление нормали к поверхности второго порядка.

Поверхность второго порядка задается в общем случае уравнением вида: Q(x,y,z) = a1x2 + a2y2 + a3z2 + b1yz + b2xz + b3xy + c1x +c2y +c3z + d =0.

Но мы будем использовать другую форму записи. Так уравнение эллипсоида будет выглядеть следующим образом:

(x-x0)2/A2 + (y-y0)2/B2 + (z-z0)2 /C2 = 1,

где x0, y0, z0 – координаты центра эллипсоида, A, B, C – длины полуосей эллипсоида. Уравнение параболоида:

(x-x0)2/A2 + (y-y0)2/B2 - (z-z0)2 /C2 = 1,

где x0, y0, z0 – координаты центра параболоида, A, B, C – длины полуосей параболоида. Ось параболоида расположена вдоль оси Oz мировой системы координат.

Для вычисления координат вектора нормали необходимо вычислить частные производные по x, y, z.

Координаты вектора нормали эллипсоида:

Xn = 2(x-x0)/A2,

Yn = 2(y-y0)/B2,

Zn = 2(z-z0)/С2.

Направление вектора не изменится, если все его координаты разделить на 2:

Xn = (x-x0)/A2,

Yn = (y-y0)/B2,

Zn = (z-z0)/С2.

Координаты вектора нормали параболоида вычисляются аналогично:

Xn = (x-x0)/A2,

Yn = (y-y0)/B2,

Zn = – (z-z0)/С2.

Нормаль для поверхности второго порядка придется вычислять непосредственно в теле трассировки, так как в разных точках фигуры нормали разные.

### Вычисление отраженного луча

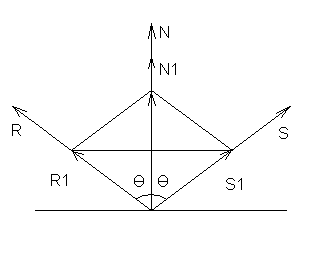
Пусть задан вектор падающего луча S, а также известен вектор нормали N. Требуется найти вектор отраженного луча R.

Рассмотрим единичные векторы R1, S1и N1. Поскольку векторы нормали, падающего луча и отраженного 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 2cosθ. Так как вектор N` по направлению совпадает с N1, то

N` = N`2cosθ.

Отсюда найдем единичный вектор отраженного луча:

R1 = N1 2cosθ – S1 = N/|N| 2cosθ – S/|S|.



Найдем cosθ. Это можно сделать, используя скалярное произведение векторов N и S:

cosθ = N\*S/(|N|\*|S|).

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

R = N 2NS/|N|2 – 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.

### Вычисление преломленного луча

Пусть даны два единичных вектора: S1 – вектор падающего луча, и N1 – вектор нормали к границе раздела двух сред. Также должны быть известны два коэффициента преломления для данных сред – n1 и n2 (или их отношение).

Требуется найти единичный вектор преломленного луча T1. Для решения выполним некоторые геометрические построения.

Искомый вектор T1 равен сумме двух векторов:

T1 = NT + B.

Найдем вначале вектор NT. Он противоположен по направлению вектору нормали, а его длина равна |T1| cos α2 = cos α2 (поскольку T1 – единичный). Таким образом, NT = -N1 cos α2. Необходимо определить cos α2. Запишем закон преломления n1 sin α1 = n2 sin α2 в виде:

sin α2 = n sin α1,

где n = n1 / n2.

Воспользуемся тождеством cos2α + sin2α = 1. Значение cos α1 можно выразить через скалярное произведение единичных векторов S1 и N1, то есть cos α1 = S1N1. Тогда мы можем записать такое выражение для вектора NT:

NT = -N1√1+n2((S1N1)2 – 1).

Осталось найти выражение для вектора B. Он располагается на одной прямой с вектором A, причем A = S1 – NS. Учитывая, что NS равен N1 cosα1, то A = S1 – N1 cosα1. Так как cosα1 = S1N1, то A = S1 – N1 (S1N1).

Поскольку длина вектора A равна sin α1, а длина вектора B равна sin α2,

|B|/|A| = sin α2/ sin α1 = n2/n1 = n,

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

B = –nA =n(N1(S1N1) – S1).

Теперь мы можем записать искомое выражение для единичного вектора луча преломления T1:

### 

### Вычисление точки пересечения с примитивами

В алгоритме трассировки для построения изображения необходимо вычислять точки пересечения лучей с примитивами сцены. Луч задается параметрическим уравнением прямой. Любая точка луча удовлетворяет уравнению

R = A + Vt,

где R – радиус вектор произвольной точки, принадлежащей лучу, A – радиус- вектор начальной точки луча, V – направляющий вектор 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:

x = x1 + at,

y = y1 + bt,

z = z1 + ct.

Здесь x1, y1, z1 – координаты начальной точки луча в прямоугольной декартовой мировой системе координат, a,b,c – координаты направляющего вектора луча.

Вычисление точки пересечения луча с поверхностью второго порядка.

Для нахождения точки пересечения луча, заданного уравнениями (2) с поверхностью второго порядка, заданной уравнениями (2.2.2.3) или (2.2.2.4):

(x–x0)2/A2 + (y–y0)2/B2 + (z–z0)2 /C2 = 1 (эллипсоид)

(x–x0)2/A2 + (y–y0)2/B2 – (z–z0)2 /C2 = 1 (параболоид),

нужно подставить в уравнение поверхности второго порядка вместо x, y и z соответствующие уравнения луча. В результате этого после раскрытия всех скобок и приведения подобных мы получим квадратное уравнение относительно параметра t. Если дискриминант квадратного уравнения меньше нуля, то луч и поверхность второго порядка общих 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.

Для нахождения координат точек пересечения луча и поверхности достаточно подставить найденные значения параметра t в уравнения луча (2).

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. Тут надо заметить, что в результате решения квадратного уравнения одно или оба значения параметра t могут получиться отрицательными. Это означает, что точка пересечения лежит «сзади» относительно точки начала луча, на половине прямой, находящейся «по нашу сторону» относительно картинной плоскости. Такие точки при поиске пересечения отбрасываются.

Кроме того, в программе для каждой фигуры введены верхняя и нижняя секущие плоскости. Отображается только часть фигуры, лежащая между ними.

Для этого после нахождения точки пересечения анализируется ее z-координата.

Вычисление точки пересечения луча с полигоном (Треугольником).

Для вычисления точки пересечения луча, заданного уравнениями (2) необходимо сначала определить точку пересечения этого луча с плоскостью, содержащей этот треугольник.

Уравнение плоскости выглядит следующим образом:

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

Здесь коэффициенты A, B, C совпадают с координатами нормали к этой плоскости. Координаты нормали плоскости совпадают с координатами нормали треугольника, которые мы посчитали на этапе загрузки сцены.

Для нахождения свободного члена D необходимо подставить координаты любой точки треугольника, например, одной из вершин.

D = –Ax –By – Cz.

По ходу выполнения программы значение D меняться не будет, поэтому его целесообразно посчитать при инициализации сцены и хранить, как и координаты нормали. Пересчитывать его необходимо только при изменении положения треугольника.

Теперь для нахождения точки пересечения подставим уравнения луча (2) в уравнение плоскости.

A (x1 + at) + B (y1 + bt) + C (z1 + ct) + D = 0

t = – (Ax1 + By1 + Cz1 + D) / (Aa + Bb + Cc)

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

Для нахождения координат точки пересечения надо подставить найденное значение параметра t в уравнения луча (2). Назовем точку пересечения D. Мы получим координаты xD, yD, zD.

Теперь необходимо определить, попала ли точка D внутрь треугольника. Найдем координаты векторов AB, BC, CA (A, B, C – вершины треугольника) и координаты векторов AD, BD, CD. Затем найдем три векторных произведения:

nA = AB x AD,

nB = BC x BD,

nC = CA x CD.

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

Операцию проверки принадлежности точки D треугольнику ABC можно ускорить. Если ортогонально спроецировать треугольник ABC и точку D на одну из плоскостей xOy, yOz или xOz, то попадание проекции точки в проекцию треугольника будет означить попадание самой точки в треугольник (конечно же, если уже известно, что точка D лежит в плоскости, содержащей треугольник ABC). При этом число операций заметно сокращается. Так для поиска координат всех векторов нужно искать по две координаты на каждый вектор, а при поиске векторных произведений нужно искать только одну координату (остальные равны нулю).

Для проверки сонаправленности векторов, полученных при вычислении векторного произведения нужно проверить знаки этой единственной координаты для всех трех векторов. Если все знаки больше нуля, или меньше нуля, то вектора сонаправлены. Равенство нулю одного из векторных произведений соответствует случаю, когда точка D попадает на прямую, содержащую одну из сторон треугольника.

Кроме того, перед вычислениями векторов и векторных произведений можно провести простой габаритный тест. Если проекция точки D лежит правее, левее, выше или ниже каждой из проекций вершин треугольника, то она не может лежать внутри.

Остается добавить, что для проецирования лучше выбирать ту из плоскостей, площадь проекции треугольника на которую больше. При таком условии исключается случай проецирования треугольника в отрезок (при условии, что проверяемый треугольник не вырожден в отрезок). Кроме того, при увеличении площади проекции уменьшается вероятность ошибки. Для определения такой плоскости проецирования достаточно проверить три координаты нормали треугольника. Если z-координата нормали больше (по абсолютному значению) x и y, то проецировать надо на плоскость xOy. Если y больше чем x и z, то проецируем на xOz. В оставшемся случае – на yOz.

### **Описание типов данных. Структура программы**

Список модулей:

TTex.h - описание структуры TTex

TSurfTex.h - описание структур TPlaneTex и TEllipsoidTex

TPoint.h - описание структур TPoint2d и TPoint3d

TRGBColor.h - описание страктуры TRGBColor

TLamp.h - описание класса TLamp

TCam.h - описание класса TCam

TPrimitive.h - описание класса TPrimitive

TFrstSurface.h - описание класса TFrstSurface

TScndSurface.h - описание класса TScndSurface

TTriangle.h - описание класса TTriangle

TEllipsoid.h - описание класса TEllipsoid

TParaboloid.h - описание класса TParaboloid

TScene.h - описание класса TScene

TTracer.h - описание класса TTracer

Модули реализующие, интерфейс программы:

\_AboutUnit.h - модуль формы «О программе»

\_ZoomUnit.h - модуль формы «Лупа»

\_Options.h - модуль формы «Опции»

\_ExtraGlassOptions.h - модуль формы «Свойства стекла»

\_ExtraTableOptions.h - модуль формы «Свойства стола»

\_ExtraCamOptions.h - модуль формы «Свойства камеры»

\_MainUnit.h - модуль главной формы программы

# 

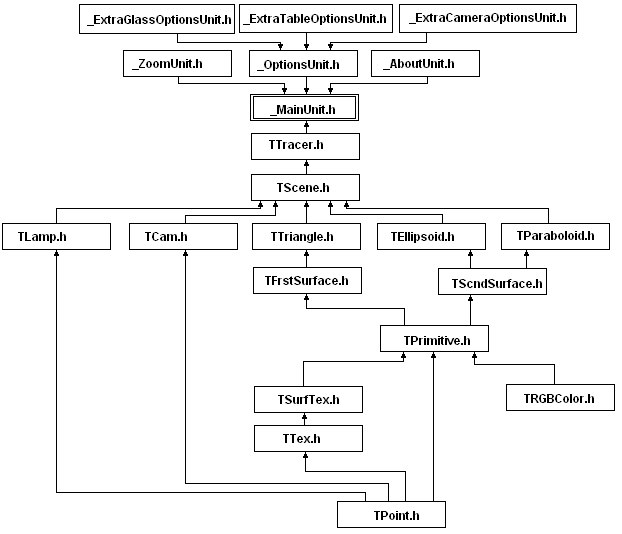
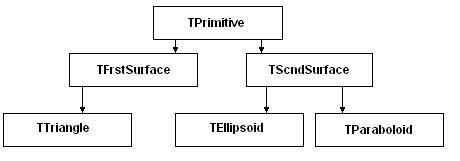


Рис. 2.3.1. Схема связей между модулями программы.



**Рис.2.3.2.** *Схема наследования примитивов*

Краткое описание структур и классов программы

struct TPoint3d – структура, описывающая точку в мировой системе координат

struct TPoint2d – структура, описывающая точку на плоскости (в текстуре) с целочисленными координатами

struct TRGBColor – структура, описывающая цвет по трем составляющим (RGB)

struct TTex – структура, описывающая текстуру – содержит адрес массива пикселей и его размеры

struct TPlaneTex – структура, описывающая привязку текстуры к плоскости.

Содержит три точки, к которым привязывается текстура

class TLamp – класс, описывающий источник освещения.

Содержит объект TPoint3d coord с координатами источника и три переменные типа float (Ir, Ig, Ib) для хранения интенсивности трех компонент света.

class TCam – класс, описывающий камеру.

Содержит два угла (a, b), указывающих направление зрения камеры, точку, на которую направлена камера (viewP) и расстояние от камеры до этой точки (r).

class TPrimitive – абстрактный класс примитива. От него наследуются поверхности первого и второго порядка.

class TFrstSurface – абстрактный класс поверхности первого порядка. От него наследуется класс треугольника.

class TScndSurface – абстрактный класс поверхности второго порядка. От него наследуются классы эллипсоида и параболоида.

class TTriangle – класс треугольника. Содержит три вершины треугольника и его нормаль

class TParaboloid – класс параболоида.

class TEllipsoid – класс эллипсоида.

class TScene – класс сцены. Содержит информацию о всех примитивах, источниках и камере.

class TTracer – класс, отвечающий за построения изображения. Содержит буфер (buffer) разметом 400x400 пикселей, в котором формируется изображение сцены. Перед генерацией необходимо вызвать функцию

selectScene передав ей в качестве параметра указатель на сцену, которуюнеобходимо сгенерировать. Для генерации вызвать функцию render.

Все классы – потомки TPrimitive предоставляют следующие функции:

float getT(TPoint3d p0, TPoint3d viewDir) – возвращает расстояние от точки начала(p0) луча viewDir до ближайшей точки пересечения с примитивом void getTArr(float\* arr, int& n, TPoint3d p0, TPoint3d viewDir) – заполняет массив arr расстояниями от точки начала(p0) луча viewDir до ближайшей всех точек пересечения с примитивом void getNormal(TPoint3d& n, const TPoint3d& p) – возвращает координаты вектора нормали к примитиву в точке p void getColor(TRGBColor& c, const TPoint3d& p) - возвращает цвет примитива точке p (с учетом текстуры).

# *2. Технологическая часть*

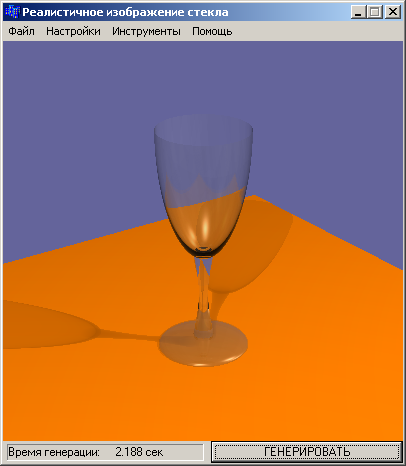
Выбор языка программирования

При разработке программы был использован язык программирования высокого уровня C++ в составе среды визуального программирования CBuilder6.

Данный язык был выбран благодаря тому, что он предоставляет максимально удобные средства по работе с оперативной памятью, позволяет реализовывать алгоритмы более эффективно, по сравнению с другими высокоуровневыми языками. Программы, написанные на C++, работают быстрее и занимают меньше места на диске.

Кроме того, среда визуального программирования CBuilder6 предоставляет большое количество стандартных визуальных компонентов для создания интерфейса, и ряд библиотек с различными часто используемыми полезными функциями.

## Описание интерфейса программы



Главное меню программы:

- **Файл**

**- Сохранить картинку –** записать в файл содержимое окна вывода изображения

- **Выход** – выход из программы

- **Настройки**

**- Опции –** открытие формы «Опции»

**- Инструменты**

**- Лупа –** открытие формы «Лупа»

**- Генерировать** – запуск генерации изображения

**- Помощь**

**- О программе** – сведения о разработчике и версия программы

Для генерации изображения нажмите кнопку ГЕНЕРИРОВАТЬ или выберите аналогичный пункт в главном меню.

В нижней строке во время генерации изображения выдается сообщение ИДЕТ ГЕНЕРАЦИЯ ИЗОБРАЖЕНИЯ. По завершении генерации здесь будет отображено время, затраченное на генерацию.



На этой вкладке находятся средства по настройке освещения сцены.

**Координаты источника** – координаты в мировой системе координат источника света, выбранного в выпадающем списке.

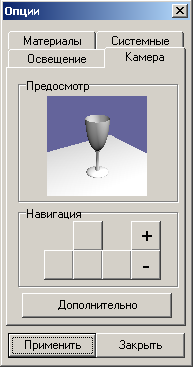
**Интенсивность источника** – значения трех компонент интенсивности источника света, выбранного в выпадающем списке.

**Фоновая интенсивность** – значения трех компонент фоновой интенсивности.

**Кнопка “+”** (рядом с выпадающим списком) – добавление нового источника света.

**Кнопка “**–**”** (рядом с выпадающим списком) – удаление источника света, выбранного в выпадающем списке.

ФОРМА «ОПЦИИ». ВКЛАДКА «КАМЕРА».



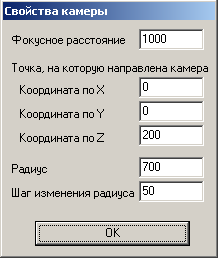
На этой вкладке находятся средства по настройке опций камеры.

**Предосмотр** – здесь можно увидеть примерный вид изображения до его генерации.

**Навигация** – настройки положения камеры.

**Дополнительно** – при нажатии на эту кнопку появляется форма **Свойства камеры** с дополнительными параметрами камеры.

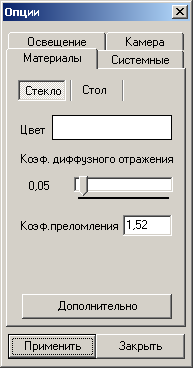
ФОРМА «СВОЙСТВА КАМЕРЫ».



**Радиус** –расстояние от камеры до точки, на которую она направлена.

**Шаг изменения радиуса** – приращение радиуса камеры при однократном нажатии кнопки “–” на вкладке “Камера” формы “Опции” (или уменьшение при однократном нажатии кнопки “+”).

ФОРМА «ОПЦИИ». ВКЛАДКА «МАТЕРИАЛЫ».

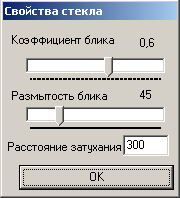


При нажатии на кнопку «Стекло» отображаются свойства материала стакана.

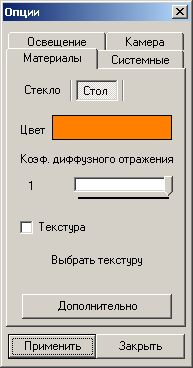
**Цвет** – цвет материала стакана.

**Коэф. диффузного отражения** – коэффициент KD материала стакана (см. раздел 2.2.1). **Коэф. преломления** – абсолютный коэффициент преломления материала стакана. **Дополнительно** – при нажатии на эту кнопку появляется форма **Свойства стекла** с дополнительными параметрами материала стакана.

ФОРМА «СВОЙСТВА СТЕКЛА».



**Коэффициент блика** – коэффициент KS материала стакана (см. раздел 2.2.1). **Размытость блика** – показатель степени p материала стакана. **Расстояние затухания** – максимальное расстояние, которое может пройти свет в материале до полного затухания. При нажатии на кнопку «Стол» отображаются свойства материала стола.



**Цвет** – цвет материала стола.

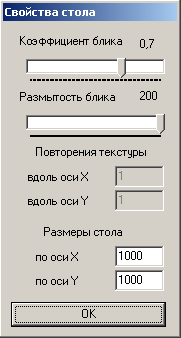
**Коэф. диффузного отражения** – коэффициент KD материала стола (см. раздел 2.2.1).

**Текстура** – если галочка установлена, то на столе будет отображаться текстура.

**Выбрать текстуру** – выбор файла изображения (\*.bmp), который будет использоваться как текстура стола.

**Дополнительно** – при нажатии на эту кнопку появляется форма **Свойства стола** с дополнительными параметрами материала стола.

ФОРМА «СВОЙСТВА СТОЛА».

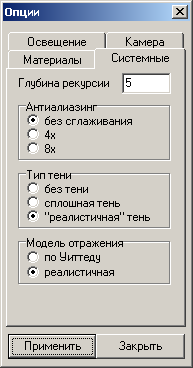


**Коэффициент блика** – коэффициент KS материала стола (см. раздел 2.2.1).

**Размытость блика** – показатель степени p материала стола.

**Повторения текстуры** – сколько раз текстура стола будет повторяться вдоль осей OX и OY.

ФОРМА «ОПЦИИ». ВКЛАДКА «СИСТЕМНЫЕ».



На этой вкладке можно настраивать алгоритмы, реализованные в программе.

**Глубина рекурсии –** этот параметрустанавливает глубину рекурсии в алгоритме трассировки. При бОльших значениях этого параметра качество сгенерированного изображения улучшается.

**ВНИМАНИЕ!**

Глубина рекурсии СИЛЬНО влияет на скорость генерации изображения. Не рекомендуется ставить значения этого параметра больше 10.

**Анитиалиазинг** – включение алгоритма сглаживания изображения.

**Тип тени** – выбор алгоритма построения теней.

Лупа позволяет рассмотреть сгенерированное изображение в увеличенном виде.

Можно менять масштаб увеличения лупы (1:1, 1:2, 1:4, 1:8, 1:10, 1:16) в выпадающем меню.

# 3. Исследовательская часть

Исследования проводились на компьютере со следующей конфигурацией:

CPU AMD Athlon 2100+

512 Mb DDR SDRAM

WindowsXP

## Зависимость времени генерации от глубины рекурсии

В этом тесте исследовалась зависимость времени генерации изображения от глубины рекурсии. Исследования проводились для сцены освещенной одним источником света.

T1 – время генерации без тени в секундах.

T2 – время генерации со сплошной тенью в секундах.

T3 – время генерации с реалистичной тенью в секундах.

N – глубина рекурсии.

### 



### **Зависимость времени генерации от количества источников**



# **Анализ результатов исследований**

Из первого исследования видно, что график зависимости времени генерации изображения от глубины рекурсии по виду совпадает с графиком функции 2n. Это хорошо соответствует теории, т.к. количество лучей растет с увеличением глубины рекурсии как 2n.

Надо заметить, что для сцен с маленьким количеством полигонов нет необходимости задавать большие значения максимальной глубины рекурсии, т.к. разница в качестве сгенерированного изображения будет несущественна.

Во втором исследовании показано, что зависимость времени генерации от количества источников света линейна. Из полученных значений можно вычислить время, необходимое для расчета одного источника. На машине, на которой проводились исследования, при глубине рекурсии 5 это время примерно равно 0,8 секунды.

Заключение

В этой программе были продемонстрированы результаты роботы алгоритма генерации реалистичных изображений – обратной трассировки лучей.

Данная реализация демонстрирует возможности алгоритма строить изображения близкие к фотореалистичным. Трассировка является одним из самых совершенных алгоритмов генерации реалистичных изображений. Качество получаемого изображения несравнимо лучше, чем качество изображения, полученного с помощью таких алгоритмов, как Z-буфер. Однако требования к вычислительным мощностям, необходимым для генерации одного кадра изображения намного выше, чем в том же Z-буфере. Это делает невозможным применение этого алгоритма для реализации приложений, где необходима генерация изображений в реальном времени (3D-игры, различного рода симуляторы). Существуют методы повышения быстродействия алгоритма. Для увеличения скорости генерации изображения применяют такие средства, как метод порталов и метод оболочек.

Метод порталов подразумевает деление виртуального пространства сцены на некоторые замкнутые области. Например, разделение дома на комнаты. Двери между комнатами закрыты, и мы не можем видеть объекты в другой комнате. Это значительно сокращает количество примитивов, с которыми необходимо искать пересечения лучей при трассировке.

В методе оболочек примитивы сцены разделены на некоторые логические объекты, каждый из которых заключен в оболочку простого вида (шар, цилиндр, параллелепипед). Перед нахождением пересечения луча с примитивом проверяется пересечение с оболочкой, содержащей этот примитив. Оболочки часто делают вложенными (образуется древовидная структура), что заметно ускоряет поиск и отброс тех примитивов, пересечений с которыми точно не будет.

Но даже со всеми усовершенствованиями на сегодняшний день реализация трассировки в реальном времени представляется затруднительной. Известны попытки создания 3D-ускорителей с аппаратной поддержкой алгоритма обратной трассировки, однако широкого распространения они не получили из-за относительно высокой цены и недостаточного быстродействия (удовлетворительная скорость построения изображения была только для изображения с разрешением 640x480).

Однако вычислительные мощности персональных компьютеров растут каждый день. В скором времени должны появиться машины, на которых алгоритм трассировки будет работать в реальном времени. Трассировка должна вытеснить другие алгоритмы ввиду явного превосходства в качестве изображения и универсальности метода.

Список литературы

1. Роджерс Д. Алгоритмические основы машинной графики: пер. с англ.— М.: Мир, 1989.— 512 с.: ил.
2. Порев В. Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2002. – 432 с.: ил.
3. Никулин Е. А. Компьютерная геометрия и алгоритмы машинной графики. СПб.: БХВ-Петербург, 2003. – 560с.: ил.
4. Авдеева С.М., Куров А.В. Алгоритмы трехмерной машинной графики: Учебное пособие. – М.: Изд-во МГТУ им. Н.Э. Баумана, 1996. – 60 с.: ил.