Общее знакомство с PovRay. Описание сцен, теоретико-множественные операции.

Povray – это система рейтрейсинга, то есть не realtime-рендеринг. На входе у него – программа (скрипт), описывающая пространство, на выходе – отрендеренное изображение.

Система координат: x слева направо, y снизу в верх, z – от нас. То есть – левосторонняя.

Структура программы (скрипта):

#include "заголовочные файлы"
Описание объектов

Заголовочные файлы включают в себя описание заранее заданных объектов – цветов, материалов, текстур и так далее.

Описание каждого объекта происходит следующим образом:
Название {
Параметры
}

Параметры могут быть вещественными (число с/без десятичной точкой), вектором (несколько чисел, объединенных в треугольные скобки), быть связанными с конкретным объектом (в зависимости от его определения) и быть также объектами.

В каждой сцене должны быть определены по крайней мере три типа объектов:
А) Освещение
Б) Камера (положение наблюдателя)
В) Геометрия

Пример простейшей программы:

```
#include "colors.inc"
background { color Cyan }
camera {
    location <0, 2, -3>
    look_at  <0, 1,  2>
  }
sphere {
    <0, 1, 2>, 2
    texture {
      pigment { color Yellow }
    }
  }
light_source { <2, 4, -3> color White}
```

Разберем построчно. В первой строке подключаются стандартные имена цветов (чтобы не задавать их с помощью RGB-значений). Во второй мы задаем объект, определяющий какого цвета будет у нас фон. Дальше идет описание наблюдателя, расположенного в точке, заданной location и смотрящего в направлении точки look_at. Далее идет описание объекта сферы с центром в точке (0,1,2) и радиусом 2. У сферы в качестве параметра задан объект «текстура», представляющая из себя равномерный цвет (желтый). Последним параметром идет точечный источник света белого цвета, расположенный в точке (2,4,-3).

1. Простейшие геометрические примитивы.

У каждого геометрического примитива могут быть дополнительные параметры: материал/текстура, геометрические операции такие, как поворот, вращение, перенос. Геометрические примитивы могут быть:

- Параллелепипед.

```
  box {
    <-1, 0, -1>,  // Две крайних точки
    <1, 0.5, 3>  // параллелепипеда
    {Дополнительные параметры}
  }
```

- Конус

```
cone {
    <0, 1, 0>, 0.3    // первый центр и радиус
    <1, 2, 3>, 1.0    // второй центр и радиус
    open              // у конуса нет крышек – это необязательный параметр
    texture { T_Stone25 scale 4 } // пример текстуры конуса
  }
```

- Цилиндр

```
cylinder {
    <0, 1, 0>,      // Центр первой крышки
    <1, 2, 3>,      // Центр второй крышки
    0.5             // Радиус
    open            // необязательный – убрать крышки
    texture { T_Stone25 scale 4 }
  }
```

- Плоскость

```
plane { <0, 1, 0>, -1
    pigment {
      checker color Red, color Blue
    }
  }
```

Параметры: вектор нормали к плоскости и расстояние от плоскости до начала координат. Если отрицательное – в обратную сторону.

- Тор

```
torus {
    4, 1              // внешний и внутренний радиус
    rotate -90*x      // геометрическое преобразование (не обязательный)
    pigment { Green } // цвет
  }
```

Тор по умолчанию располагается в начале координат, то есть чтобы расположить его в другом месте, нужно геометрическое преобразование в параметрах.

Теоретико-множественные операции:

- Объединение

union {
объект 1
объект 2
…
}

Пример:

-Обьедеинение

```
union{
    sphere { <0, 0, 0>, 1
```

```
      pigment { Blue }
      translate -0.5*x
    }
    sphere { <0, 0, 0>, 1
      pigment { Red }
      translate 0.5*x
    }
  }
```

- Пересечение

```
intersection {
    sphere { <0, 0, 0>, 1
      translate -0.5*x
    }
    sphere { <0, 0, 0>, 1
      translate 0.5*x
    }
    pigment { Red }
  }
```

Обратите внимание, что цвет (красный) будет для объекта пересечение.

- Разница

```
difference{
    sphere { <0, 0, 0>, 1
      translate -0.5*x
    }
    sphere { <0, 0, 0>, 1
      translate 0.5*x
    }
    pigment { Red }
    rotate 90*y
  }
```

Задание на работу: Пользуясь полученными знаниями, написать скрипт, который определяет объект некоторой сложности, используя теоретико-множественные операции.

Как использовать переменные в программе.

Для задания переменных используются команды #declare или #local. Их отличие в области видимости: #declare определяет глобальную переменную, #local – локальную. Используйте глобальные переменные всегда, кроме макросов.

например:

```
#declare Rows = 5;
#declare Count = Count+1;
#local  Here = <1,2,3>;
#declare White = rgb <1,1,1>;
#declare Cyan = color blue 1.0 green 1.0;
#declare Font_Name = "ariel.ttf"
#declare Rod = cylinder {-5*x,5*x,1}
#declare Ring = torus {5,1}
#local  Checks = pigment { checker White, Cyan }
object{ Rod scale y*5 }     // not "cylinder { Rod }"
object {
```

```
  Ring
  pigment { Checks scale 0.5 }
  transform Skew
 }
```

Существуют **управляющие конструкции** if-else, while:

```
#declare Which=1;

 #if (Which)
   box { 0, 1 }
 #else
   sphere { 0, 1 }
 #end
```

```
#declare dx=0;
#while (dx<10)
  sphere {<dx,dx*dx,0>, 3 pigment { color Red } }
  #declare dx=dx+1;
#end
```

И макросы

```
#macro Parabola (Radius,MyColor)
        #local dx=0;
        #while (dx<10)
                sphere {<dx,dx*dx,0>, Radius pigment { color MyColor } }
                #local dx=dx+1;
        #end
#end
```

потом, в описании сцены надо сделать вызов макроса

```
Parabola (1,Red)
union{
Parabola (3,<0,0,1>)
translate <0,0,6>
}
```

Задание — смоделировать сыр (с «дырками»).
Для упрощения размещения сфер, которыми описываются дырки, рекомендуется сначала их размещать видимыми, для того, чтобы убедиться, они располагаются согласно ожиданиям, а затем внести их в качестве операндов вычитания из куска сыра в команду **difference.**

Лаба1 часть2. Использование сложных настроек освещенности.

Теория:

В настоящее время, программы в компьютерной графике, при создании изображения используют эффекты освещенности, которые условно можно отнести к 4-м группам:
1) Расчет первичной освещенности и тени
2) Расчет эффектов зеркальных отражений и преломлений
3) Расчет вторичных диффузных отражений (radiosity)
4) Расчет эффектов фокусировки света (caustic)

Расчет первичной освещенности состоит из получения освещенности точки на объекте с использованием 3-х составляющих освещенности: фоновой, диффузной, зеркальной(блик). Если объект находится в тени, то используется только фоновая составляющая, а если на него падает свет, то все три.

Расчет эффектов зеркальный отражений и преломлений влияет на то, насколько прозрачным получится объект, и/или насколько много может отразиться на его поверхности предметов (как в зеркале).

Вторичные диффузные отражения (radiosity) отвечают за очень мягкие тени, и за эффект похожий на самоизлучение рассеянного света поверхностью. Например, на полу в белой комнате лежит красный ковер, из окна на ковер падает яркий солнечный свет. И в результате вторичных диффузных отражений, стены и потолок этой комнаты приобретут красноватый оттенок. Использование такого эффекта значительно повышает фотореалистичность изображения, однако сильно замедляет вычисления, так как теперь, даже если объект(в нашем примере -палас) находится вне камеры(камера направлена в потолок), для него(паласа) требуется произвести расчет освещенности.

Расчет эффектов фокусировки света отвечает за фокусировку света, проходящего через объект(например, увеличительное стекло), или отраженного объектом, (например блики света от поверхности воды сфокусированные на слабоосвещенном предмете – потолке скального грота).

Radiosity считается достаточно долго и этой теме будет посвящен отдельный перевод, размещенный на сайте. Остальные параметры есть в следующем коде:

```
#include "colors.inc"

 plane {
   y, -9
   pigment {checker Green, Red}
  }
light_source { <-52, 14, -30> color White}
background {color Grey}
camera {

      location <-6, 9, -15>
      look_at <0, 0, 0>
     }


sphere {
   0,8
   pigment {
         rgbf <0.2,1,0.5,0.9>   // ЦВЕТ в RGBA, 4-й компонент - прозрачность
   }
   finish {
          ambient <.2,0.2,.2>   // фонов составлющ
          diffuse .6
          phong 0.9 phong_size 10              // настройки блика
          reflection{ 0.1,  0.8     // мин и макс энергии которое пойдет на отражение в
зависимости от угла падения
                    fresnel  // расчет отражен на основе коэффициента преломлен (ior)
параметр можно удалить
                    }
          conserve_energy  // использовать в паре с прозрачность
                      //   закон сохранен энергии
                     // параметр можно удалить


        }
   interior {ior 1.3    // коэффициент преломления
           caustics 1 // будет ли фокусировка света
           // dispersion 1.1 // коэф преломлен в зависимости от длины волны
          // dispersion_samples 12  // число полос в спектре
         }


   }
```

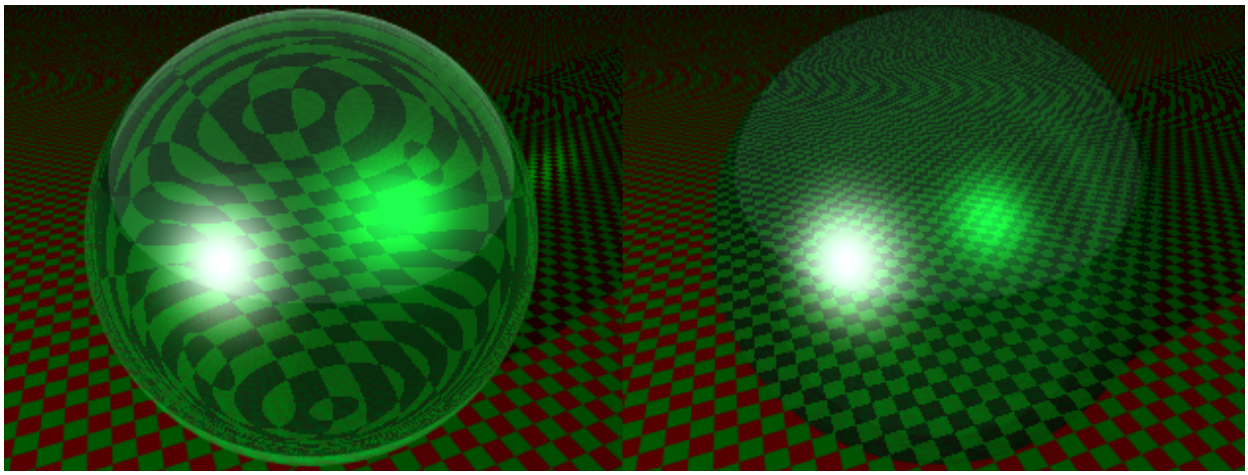

Рис 1. Изображение сферы с указанием коэффициента преломления света(ior и секция interrior) – слева, и без задания коэффициента преломления света — справа.
Можно видеть что сфера слева фокусирует свет в области своей тени.
Сферу слева можно сравнить с очень прозрачной просвечивающей тканью.
В обоих случаях, красный свет, отраженный от красных клеток пола не пропускается сферой

Разберем что здесь представлено:

Во первых хочется отметить, что povray бывает ругается на русские комментарии, в особенности ему не нравится буква «я», так что возможно вам придется часть комментариев убрать в ручную.

```
#include "colors.inc"

 plane {
   y, -9
   pigment {checker Green, Blue}
  }
light_source { <-52, 14, -30> color White}
background {color Grey}
camera {
       location <-6, 8, -15>
       look_at <0, 0, 0>
      }
```

Подключаются цвета, определяется плоскость закрашенная в шахматном порядке, создается источник света, описывается фон, устанавливается камера – подробнее следует смотреть в первой лабе.

```
sphere {0,8
   pigment {  …    }
   finish {   …
          reflection { }
            }
```

interior { … }

}

Здесь определяется геометрический объект- сфера, у которого есть специфические параметры:
pigment – отвечает за внешнюю окраску предмета
finish – отвечает за расчет первичной освещенности поверхности внутри секции finish,
reflection отвечает за настройки отражений других объектов;
interior - отвечает за преломление света, проходящего через геометрический обьект.

Разберем подробно:

Секция pigment:

Если написать
pigment {rgbf <0.5,0.0,0.0,0.8> }
Без секций **finish** и **interior** то у нас будет сфера с просвечивающими предметами находящимися за ней. Здесь команда **rgbf** задает цвет поверхности объекта в режиме RGBA, где A будет отвечать за прозрачность этой сферы. Она не будет преломлять свет, так как нет секции **interior,** а настройки **finish** будут приняты по умолчанию.

Примечание: Если вы задаете цвет вида **rgb <1 , 1, 1 >** или **Red**, **White** то так вы делаете полностью непрозрачный цвет, и вне зависимости от того описываете, ли вы или нет секцию **interior**, объект не будет пропускать свет через себя. Чтобы описать прозрачный объект, надо использовать четырехкомпонентный вектор **rgbf** и 4-м параметром, отличным от 0, или использовать цвет в котором определена эта четвертая компонента, например цвет **Clear.**

Секция **finish**

```
finish {
        ambient <.2,0.2,.2>   // фонов составлющ
        diffuse .6
        phong 0.9 phong_size 10          // настройки блика
```

Эта секция отвечает за фоновую (ambient) и диффузную (diffuse)составляющие, а также за блик, который может быть рассчитан по двум разным формулам (вторая альтернатива закомментирована). Параметр 0.9 отвечает за интенсивность блика(только света отраженного от источников освещения, не путать с отражением объектов)

Группа команд

```
        reflection{ 0.1,  0.8     // мин и макс энергии которое пойдет на отражение
                fresnel  // расчет отражен на основе коэффициента преломлен (ior)
                }
```

Отвечает за отражения других предметов(не источников освещения!) на поверхности объекта. Два числа/вектора определяют минимум и максимум интенсивности отражения в зависимости от угла падения взгляда. Поясню: если луч падает перпендикулярно поверхности, то отражение максимально, если луч падает практически параллельно поверхности, то будет использован цвет материала.

Наличие параметра **fresnel** позволяет использовать коэффициент преломления – **ior** (определяемый в секции **interior)** в качестве определяющего фактора преломлять или

отражать падающий луч, то есть использовать формулу Френеля (см раздел физики-оптика)

Команда **conserve_energy** используется в паре с преломлением и она предписывает использовать закон сохранения энергии: если «преломилось» 80 процентов энергии, то 20 процентов будет отражено. Это позволяет создавать более реалистичные сцены.

Секция **interior.**

Секция **interior** описывает как лучи будут проходить через объект. Если эта секция не определена, то, пропуская лучи сквозь себя, объект не преломляет их (как например прозрачная ткань).

Задавая паремтры

```
interior {ior 1.8    // коэффициент преломления
          caustics 1    // будет ли фокусировка света

         // dispersion 1.1 // коэф преломления в зависимости от длины волны
         // dispersion_samples 12  // число полос в спектре
           }
```

мы заставляем povray считать преломления по физическим законам, где ior определяет коэффициент преломления объекта относительно воздуха (например у воды он равен 1.3, у стекла равен 1.6, у алмаза - 2.4)
команда **caustics** с параметром отличным от нуля заставляет считать эффекты фокусировки света, создаваемые этим предметом.
Команды **dispersion_samples** и **dispersion** позволяют определить изменение коэффициента преломления в зависимости от длины волны проходящего луча света. Это применятся, когда необходимо смоделировать явление призматического разложения солнечного света в спектр

Рис 2. Призматическия дисперсия света.

Лаба1 часть2. Использование сложных настроек освещенности.

Теория:

В настоящее время, программы в компьютерной графике, при создании изображения используют эффекты освещенности, которые условно можно отнести к 4-м группам:
1) Расчет первичной освещенности и тени
2) Расчет эффектов зеркальных отражений и преломлений
3) Расчет вторичных диффузных отражений (radiosity)
4) Расчет эффектов фокусировки света (caustic)

Расчет первичной освещенности состоит из получения освещенности точки на объекте с использованием 3-х составляющих освещенности: фоновой, диффузной, зеркальной(блик). Если объект находится в тени, то используется только фоновая составляющая, а если на него падает свет, то все три.

Расчет эффектов зеркальный отражений и преломлений влияет на то, насколько прозрачным получится объект, и/или насколько много может отразиться на его поверхности предметов (как в зеркале).

Вторичные диффузные отражения (radiosity) отвечают за очень мягкие тени, и за эффект похожий на самоизлучение рассеянного света поверхностью. Например, на полу в белой комнате лежит красный ковер, из окна на ковер падает яркий солнечный свет. И в результате вторичных диффузных отражений, стены и потолок этой комнаты приобретут красноватый оттенок. Использование такого эффекта значительно повышает фотореалистичность изображения, однако сильно замедляет вычисления, так как теперь, даже если объект(в нашем примере -палас) находится вне камеры(камера направлена в потолок), для него(паласа) требуется произвести расчет освещенности.

Расчет эффектов фокусировки света отвечает за фокусировку света, проходящего через объект(например, увеличительное стекло), или отраженного объектом, (например блики света от поверхности воды сфокусированные на слабоосвещенном предмете – потолке скального грота).

Radiosity считается достаточно долго и этой теме будет посвящен отдельный перевод, размещенный на сайте. Остальные параметры есть в следующем коде:

Удаляйте комментарии на Русском языке, так как PovRay плохо относится к некоторым русским буквам.

```
#include "colors.inc"

 plane {
   y, -9
   pigment {checker Green, Blue}
  }
light_source { <-52, 14, -30> color White}
background {color Grey}
camera {

      location <-6, 8, -15>
      look_at <0, 0, 0>
```

```
    }


sphere {0,8
   pigment {
         rgbf <1,1,1,0.8>   // ЦВЕТ в RGBA, 4-й компонент - прозрачность
    }
   finish {
          ambient <.2,0.2,.2>   // фонов составлющ
          diffuse .06
          phong 0.9 phong_size 10              // два альтернативных вида блика
          //specular 0.9 roughness 0.02         // альтернатива
            metallic  0.5 // совместно с phong или specular
          reflection{ <0.0,0,0>,  <0.0,0.9,0.8>     // мин и макс в зависимости от угла
падения
                    fresnel  // расчет отражен на основе коэффициента преломлен (ior)
                     falloff 0.00001
                     exponent 0.6    // >1 -больше интенсивность, <1 - меньше интенсивность

                    }
          conserve_energy  // использовать в паре с прозрачность
                      //   закон сохранен энергии


        }
    interior {ior 1.8    // коэффициент преломления
           caustics 1    // будет ли фокусировка света
          // dispersion 1.1 // коэф преломлен в зависимости от спектра
          // dispersion_samples 12  // число полос в спектре

           fade_color Red  // производитс затухание   к цвету
           fade_distance 999999 // расстоние на котором луч произведет затухание на
половину
           fade_power 1  // сила затухания
             }


   }
```

Разберем то как это дело работает:
Во первых хочется отметить, что povray бывает ругается на русские комментарии, в особенности ему не нравится буква «я», так что возможно вам придется часть комментариев убрать в ручную.

Разберем что здесь представлено:

```
#include "colors.inc"

 plane {
   y, -9
   pigment {checker Green, Blue}
  }
light_source { <-52, 14, -30> color White}
background {color Grey}
camera {

      location <-6, 8, -15>
      look_at <0, 0, 0>
     }
```

Подключаются цвета, определяется плоскость закрашенная в шахматном порядке, создается источник света, описывается фон, устанавливается камера – подробнее следует смотреть в первой лабе.

```
sphere {0,8
   pigment {  …    }
   finish {   …
          reflection { }
           }
    interior {   …       }

   }
```

Здесь определяется геометрический объект- сфера, у которого есть специфические параметры:

pigment – отвечает за внешнюю окраску предмета
finish – отвечает за расчет первичной освещенности поверхности внутри секции finish,
reflection отвечает за настройки отражений других объектов;
interior - отвечает за преломление света, проходящего через геометрический обьект.

Разберем подробно:

Секция pigment:

Если написать

```
pigment {rgbf <0.5,0.0,0.0,0.8>         }
```

Без секций **finish** и **interior** то у нас будет сфера с просвечивающими предметами находящимися за ней. Здесь команда **rgbf** задает цвет поверхности объекта в режиме RGBA, где A будет отвечать за прозрачность этой сферы. Она не будет преломлять свет, так как нет секции **interior,** а настройки **finish** будут приняты по умолчанию.

Примечание: Если вы задаете цвет вида **rgb <1 , 1, 1 >** или **Red**, **White** то так вы делаете полностью непрозрачный цвет, и вне зависимости от того описываете, ли вы или нет секцию **interior**, объект не будет пропускать свет через себя. Чтобы описать прозрачный объект, надо использовать четырехкомпонентный вектор **rgbf** и 4-м параметром, отличным от 0, или использовать цвет в котором определена эта четвертая компонента, например цвет **Clear.**

В секции **pigment** можно воспользоваться шаблоном , таким как **checker** – шахматный порядок, или **gradient –** повторяемый список цветов вдоль вектора градиента и другие (перевод соответствующей секции справки будет выложен на сайте).

Если указать **pigment {**

```
        gradient y      //this is the PATTERN_TYPE
              color_map {
                [0.1  color rgbf <1  ,0.0,0.0,0.8>]
                [0.3  color rgbf <0.9,0.9,0.0,0.8>]
                [0.4  color rgbf <0.0, 0 ,0.0,1>]
                [0.9  color rgbf <0  ,0.0,1.0,1>]
                [1.2  color rgbf <0.0,  0,0,1>]
                                }
                scale 5
   }
```

То эти команды закрасят объект сферу в пять разноцветных полос меняющихся вдоль Y

Секция **finish**

```
finish {
        ambient <.2,0.2,.2>   // фонов составлющ
        diffuse .06
        phong 0.9 phong_size 10             // два альтернативных вида блика
        //specular 0.9 roughness 0.02        // альтернатива
          metallic  0.5 // совместно с phong или specular
```

Эта секция отвечает за фоновую (ambient) и диффузную (diffuse)составляющие, а также за блик, который может быть рассчитан по двум разным формулам (вторая альтернатива закомментирована). Параметр 0.9 отвечает за интенсивность блика, команды **phong_size** или **roughness** отвечают за размер блика, в своих формулах. Допустимо суммировать оба блика. Параметр **metallic** вносит поправки в формулу расчет блика (в любую из альтернатив), так что блик становится похожим на блик от металлического предмета.

Группа команд

```
        reflection{ <0.0,0,0>,  <0.0,0.9,0.8>     // min и max
                fresnel  // расчет отражен на основе коэффициента преломлен (ior)
                 falloff 0.00001
                 exponent 90    // >1 -больше интенсивность, <1 - меньше интенсивность

                }
```

Отвечает за отражения других предметов(не источников освещения!) на поверхности объекта. Два числа/вектора определяют минимум и максимум интенсивности отражения в зависимости от угла падения взгляда. Поясню: если луч падает перпендикулярно поверхности, то отражение максимально, если луч падает практически параллельно поверхности, то будет использован цвет материала.

Наличие параметра **fresnel** позволяет использовать коэффициент преломления – **ior** (определяемый в секции **interior)** в качестве определяющего фактора преломлять или отражать падающий луч.
Параметр **falloff** определяет силу затухания отражения.

Параметр **exponent** подсвечивает или затемняет предметы отражающиеся на поверхности рассматриваемого объекта. Если **exponent > 1** то отражения становятся более светлыми, если **exponent < 1** то отражения будут более темными.

Команда **conserve_energy** используется в паре с преломлением и она предписывает использовать закон сохранения энергии: если преломилось 80 процентов энергии, то 20 процентов будет отражено. Это позволяет создавать более реалистичные сцены.

Секция **interior.**

Секция **interior** описывает как лучи будут проходить через объект. Если эта секция не определена, то, пропуская лучи сквозь себя, объект не преломляет их (как например прозрачная ткань).

Задавая паремтры

```
interior {ior 1.8    // коэффициент преломления
          caustics 1    // будет ли фокусировка света
         // dispersion 1.1 // коэф преломления в зависимости от длины волны
         // dispersion_samples 12  // число полос в спектре

          fade_color Red  // производитс затухание   к цвету
          fade_distance 99 // расстоние на котором луч произведет затухание на половину
                    }
```

мы заставляем povray считать преломления по физическим законам, где ior определяет коэффициент преломления объекта относительно воздуха (например у воды он равен 1.3, у стекла равен 1.6, у алмаза - 2.4)
команда **caustics** с параметром отличным от нуля заставляет считать эффекты фокусировки света, создаваемые этим предметом.
Команды **dispersion_samples** и **dispersion** позволяют определить изменение коэффициента преломления в зависимости от длины волны проходящего луча света. Это применятся, когда необходимо смоделировать явление призматического разложения солнечного света в спектр

Команда **fade_color** определяет цвет к которому будет стремиться измениться луч с проходом расстояния (как в тумане, чем дальше объекты, тем более они будут приобретать цвет тумана- белого, или если туман подсвечивается красным – красного)
Параметр **fade_distance** определяет расстоние на котором луч произведет затухание на половину.
Остальные параметры, отвечающие за преломления и отражения предметов рекомендуется изучать самостоятельно.

Различные способы построения твердых тел в PovRay.

1. Очень много тел строятся на основе кривых или сплайнов. В Pov-Ray поддерживаются 4 типа сплайнов:
- ломаная линия (точки просто соединяются друг с другом)
- квадратичная кривая (кривая определена уравнением второго порядка)
- кубический сплайн (кривая определена уравнением третьего порядка)
- естественный сплайн (кривая определена другим уравнением третьего порядка)
Для некоторых тел еще применяется кривая Безье.

2. Тело вращения (lathe) получается вращением кривой вокруг вертикальной оси Y. Кривая задается сплайном того или иного типа.
Пример.

```
#include "colors.inc"
  background{White}
  camera {
    angle 10
    location <1, 9, -50>
    look_at <0, 2, 0>
  }
  light_source {
    <20, 20, -20> color White
  }
  lathe {
    linear_spline
    6,
    <0,0>, <1,1>, <3,2>, <2,3>, <2,4>, <0,4>
    pigment { Blue }
    finish {
      ambient .3
      phong .75
    }
  }
```

Все параметры сплайна понятны: сначала идет тип кривой, потом число точек и последовательно идут точки, составляющие кривую.

3. Подобным получается другое тело вращения (surface of revolution). Отличие от lathe в том, что во-первых, точки связаны всегда не одним из вышеперечисленных методов, а интерполяционной кривой, во-вторых, координаты Y у точек не должны «идти обратно» при движении по X. То есть, если у нас есть точки <0,0> и <2,2>, то точки <4,0> быть уже не может.
Пример:

```
#include "colors.inc"
  #include "golds.inc"
  camera {
    location <10, 15, -20>
    look_at <0, 5, 0>
    angle 45
  }
  background { color rgb<0.2, 0.4, 0.8>  }
  light_source { <100, 100, -100> color rgb 1 }
  plane {
    y, 0
    pigment { checker color Red, color Green scale 10 }
  }
  sor {
    8,
    <0.0,  -0.5>,
    <3.0,   0.0>,
    <1.0,   0.2>,
    <0.5,   0.4>,
```

```
    <0.5,   4.0>,
    <1.0,   5.0>,
    <3.0,  10.0>,
    <4.0,  11.0>
    open
    texture { T_Gold_1B }
 }
```

Призма получается «протягиванием» замкнутой кривой вдоль оси Y.

```
prism {
    cubic_spline // тип кривой, лежащей в основании
    0, // координата Y нижней «крышки» призмы
    1, // координата Y верхней «крышки» призмы
    6, // число точек кривой
    < 3, -5>,
    < 3,  5>,
    <-5,  0>,
    < 3, -5>,
    < 3,  5>,
    <-5,  0>
    pigment { Green }
  }
```

Обратите внимание, что в кубическом сплайне первая и последняя точка – направляющие, то есть через них кривая не проходит. Поэтому мы замыкаем вторую и четвертую точку для получения результата.

4. Тело можно получить, «протягивая» шар по определенной траектории, заданной кривой.

```
sphere_sweep {
    linear_spline // тип кривой
    4, // число точек, составляющих кривую
    <-5, -5, 0>, 1 //координата точки, радиус сферы в этой точке
    <-5,  5, 0>, 1 //...
    < 5, -5, 0>, 1
    < 5,  5, 0>, 1
  }
```

5.Blob - каплевидные объекты

```
 #include "colors.inc"
  background{White}
  camera {
    angle 15
    location <0,2,-10>
    look_at <0,0,0>
  }
  light_source { <10, 20, -10> color White }
  blob {
    threshold .65
    sphere { <.5,0,0>, .8, 1 pigment {Blue} }
    sphere { <-.5,0,0>,.8, 1 pigment {Pink} }
    finish { phong 1 }
  }
```

Задание на работу:

Пользуясь полученными знаниями, написать скрипт, который определяет объект некоторой сложности. Примеры:

Елочка, у которой контур построен как кривая со звездой (призмой) на макушке.

Ваза с цветком (ваза – тело вращения, стебель цветка – протянутая сфера)

Капля капает из водопроводного крана