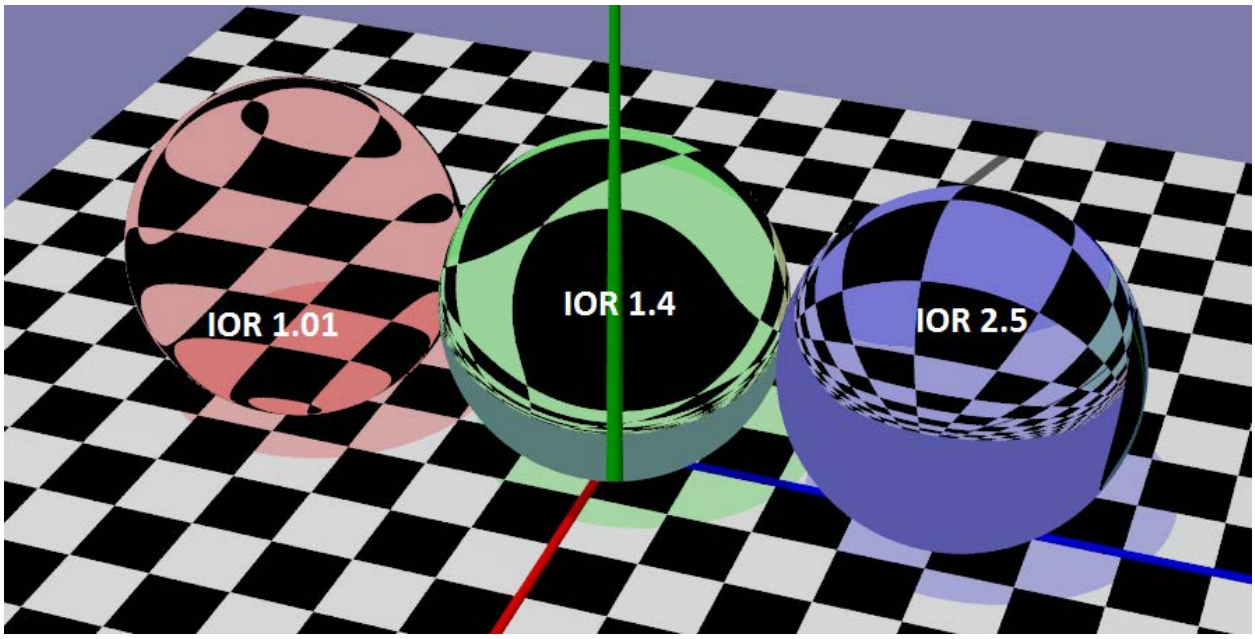
Гаврилов А. Г. Кафедра ИТиВС
V1.1 22.04.2018

Pov-Ray
Лабораторная работа 2

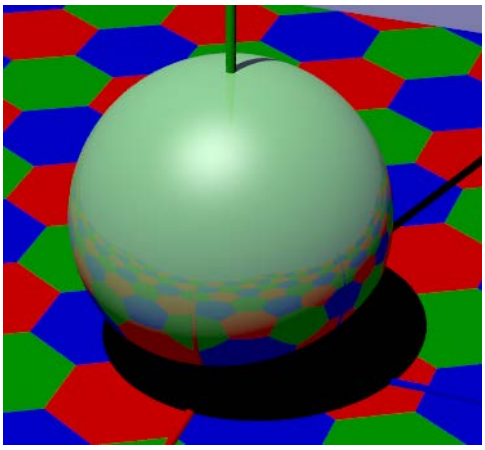
Прозрачность




```
//Зеленая сфера
sphere{
    y,1

    pigment {
          rgbf <0.7, 1, 0.7, 1>  //задание цвета с прозрачностью!
         }

    interior {
         ior 1.4   //коэффициент преломления. (1,33 - вода, 1,5 ~ стекло)
         }
}

//Красная
sphere{
    y,1

    pigment {
          rgbf <1, 0.7, 0.7, 1>
         }

    interior {
         ior 1.05  //почти без преломления
         }

     translate -2*z
}

//Синяя
sphere{
    y,1

    pigment {
          rgbf <0.7, 0.7, 1, 1>
```

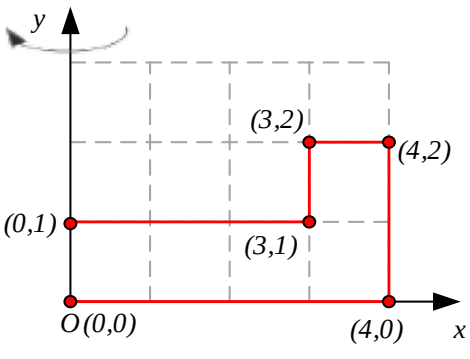
```
        }

    interior {
         ior 2.5   //алмаз
         }

    translate 2*z
}
```

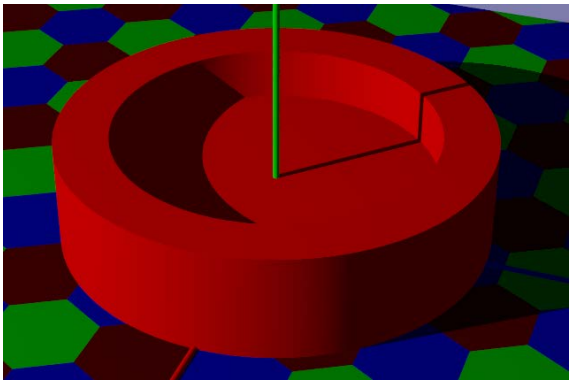
Отражение



```
//зеленый
sphere{
    y,1
    pigment {
          rgbf <0.4, 1, 0.4, 0>
         }

    finish {
        reflection {
            0.2   //степень отражения. 1 - полное отражения всех цветов. трехкомпонентный
            metallic 0 //»металличность». настолько отражение имеет цвет объекта
           }

        phong 0.4 //степень блика
        phong_size 10  //размер блика
       }
}
```

Поверхность вращения

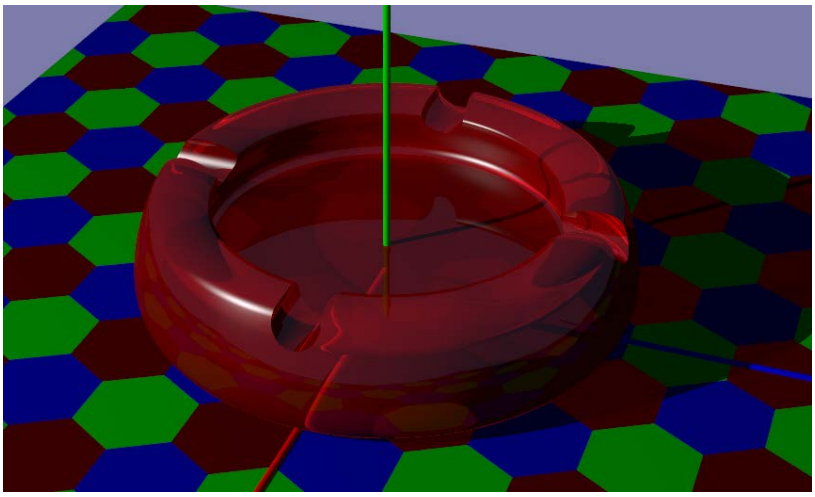




```
lathe {
  linear_spline  //тип кривой
```

```
  6 //кол-во точек
  // сами точкт в плоскости xy:
  <0, 1>, <3, 1>, <3,2>, <4, 2>, <4, 0>, <0, 0>
  pigment {color rgbf<1,0,0,0>}

  scale 0.5 //чуточку уменьшим
  }
```

Попробуем из прошлого примеров собрать что ни будь красивое. Заменим тип кривой в теле вращения на кубическую (вместо линейной). Как результат - она станет более гладкой. Двумя цилиндрами прорежем отверстия. Чуточку отражения, бликов и прозрачности – пепельница готова! А курить вредно(.

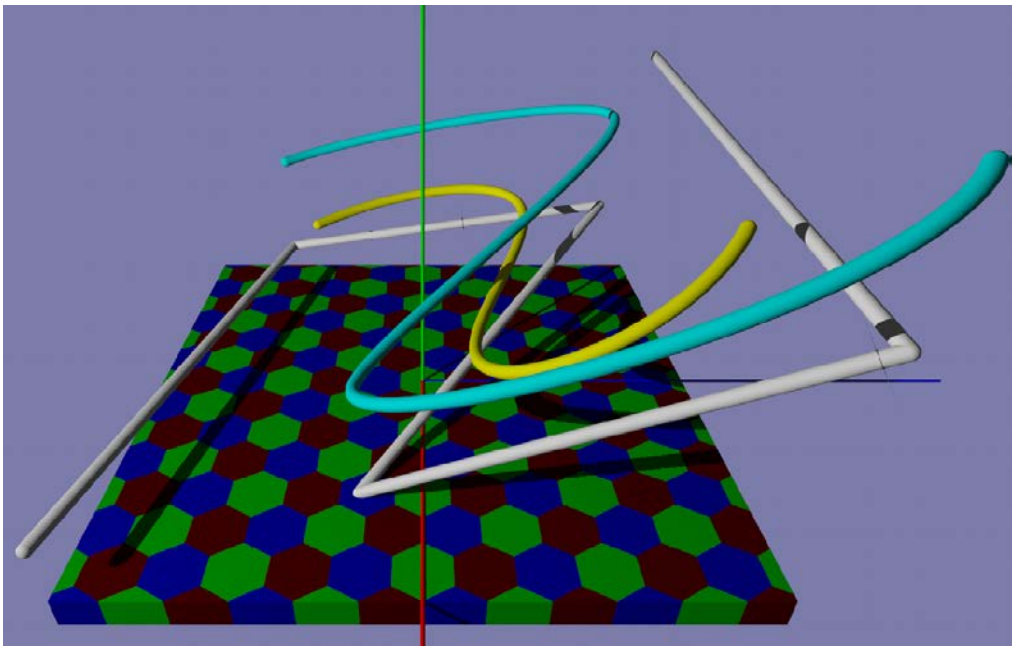


```
//пол, сделанный из призмы
prism{
  -0.5,0,4
  <5,5>,<-5,5>,<-5,-5>,<5,-5>
  pigment {
       //шестиугольная закраска (2 или 3 цвета)
       hexagon  0.5*Green  0.1*Red 0.5*Blue   scale .5
   }
}

//перельница
difference
 {
    //Объекты:
    lathe {
        cubic_spline  //кубическая кривая
        8
         // первая и последняя точка не является видимой частью кривой.
         // она нужна для определения кривизны кубического интерполяционного
         // полинома вначале и в конеце
        <-1,1>, <0, 1>, <3, 1>, <3,2>, <4, 2>, <4, 0>, <0, 0>, <-1, 0>
        scale 0.5
        }
    //Вычитаемые цилиндры образуют отверстия на стенках
    cylinder {
         <5,1,0>, <-5,1,0>, 0.2
        }
    cylinder {
         <5,1,0>, <-5,1,0>, 0.2
         rotate 90*y
        }

   //настройки
   pigment {color rgbf<1,0,0,0.8>}
   interior {
         ior 1.5
         }
   finish {
        reflection {
            0.03
          }
        phong 1 //степень блика
        phong_size 30  //размер блика
```

```
        }
    }
```

«Сферическая линия»

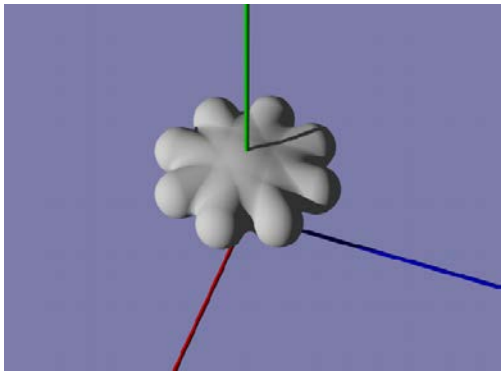
Этот объект можно представить, как след сферы, который она оставляет, летя в пространстве по определенной траектории. Траектория определяется набором узловых точек и типом аппроксимирующей кривой. Так как в узловых точках можно контролировать толщину фигуры. В примере три линии построены по одному набору точек, но с разным типом аппроксимации. Для наглядности линии были раздвинуты по оси *y*.



```
sphere_sweep{

    linear_spline, 6  //Тип интерполяции и кол-во точек
    <5,1,-5>, .1       //тут линейная интерполяция
    <-4,1,-3>, .1      //далее идут координаты точек и радиус фигуры в этой точке
    <-4,2,4>,.1
    <3 ,0,-1>,.1
    <5 ,4,5>,.1
    <-5 ,5, 5>,.1

    pigment {White}

}

sphere_sweep{

    b_spline, 6    //аппроксимация  Би-сплайнами
    <5,1,-5>, .1
    <-4,1,-3>, .1
    <-4,2,4>,.1
    <3 ,0,-1>,.1
    <5 ,4,5>,.1
    <-5 ,5, 5>,.1

    pigment {Yellow}
    translate 1*y

}

sphere_sweep{

    cubic_spline, 6    //кубическая аппроксимация
    <5,1,-5>, .1
```

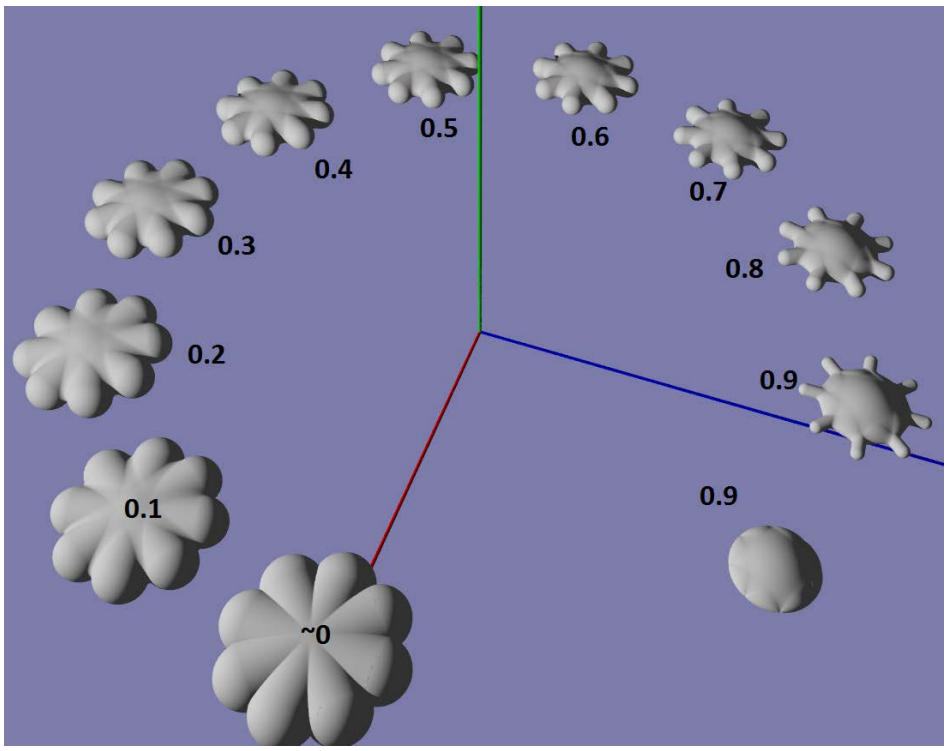
```
    <-4,1,-3>, .1
    <-4,2,4>,.1
    <3 ,0,-1>,.1
    <5 ,4,5>,.1
    <-5 ,5, 5>,.1

    pigment {Cyan}
    translate 2*y

}
```

«Капля»



```
blob
 {      //Цилиндры или сферы
        cylinder{
            <-1,1,0>, <1,1,0>, 0.5, 1  //помимо параметров фигуры нужно указать еще силу
                                                                    // "обкапливания"
        }                                                          // тут везде единица

        cylinder{
            <-1,1,0>, <1,1,0>, 0.5, 1
            rotate 45*y
        }

        cylinder{
            <-1,1,0>, <1,1,0>, 0.5, 1
            rotate 90*y
        }
        cylinder{
            <-1,1,0>, <1,1,0>, 0.5, 1
            rotate -45*y
        }

    threshold 0.3   //порог, т.е мин расстояние между точками, которые "стекаются"
    pigment {White}

}
```

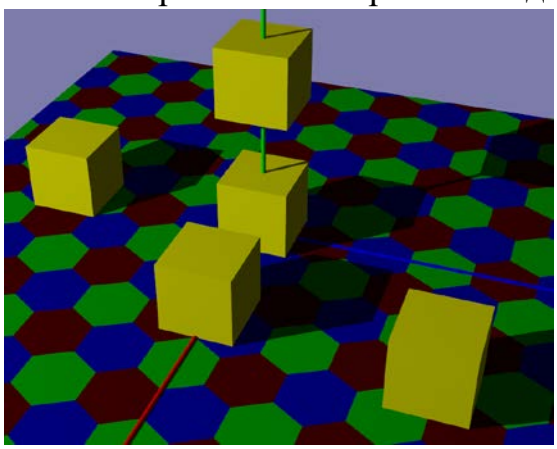
Так бы изменялась фигура при изменении порога:



Некоторые фичи языка Pow-Ray

1. Макросы

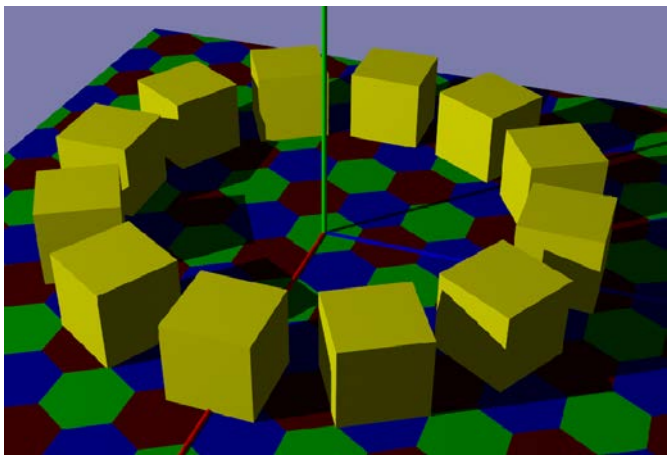
Позволяют избежать многократного копирования одинакового кода.



В примере кубик описан один раз, но благодаря макросу рисуется 5 раз:

```
//объявление макроса
//в скобках можно указывать произвольное кол-во параметров
#macro Cube (pos_x, pos_y, pos_z)
    box{
        -0.5, 0.5
        pigment {Yellow}
        translate <0, 0.5, 0>
        translate <pos_x,pos_y,pos_z>
    }
#end

//использование макроса
Cube(0,0,0)
Cube(2.4,0,3)
Cube(0,2,0)
Cube(2,0,0)
Cube(0,0,-3)
```

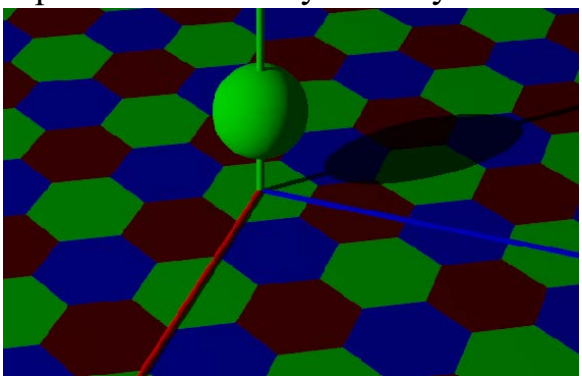
2. Циклы



```
//счетчик, начальное значение, конечное, шаг
#for (i,0, 359, 30)

    union{
        Cube(0,0,3)
        rotate i*y
    }

#end
```

3. Переменные

В Pov-Ray каждый объект, будь то сложная фигура, ее свойство или простой вектор или число могут быть упакованы в переменные.



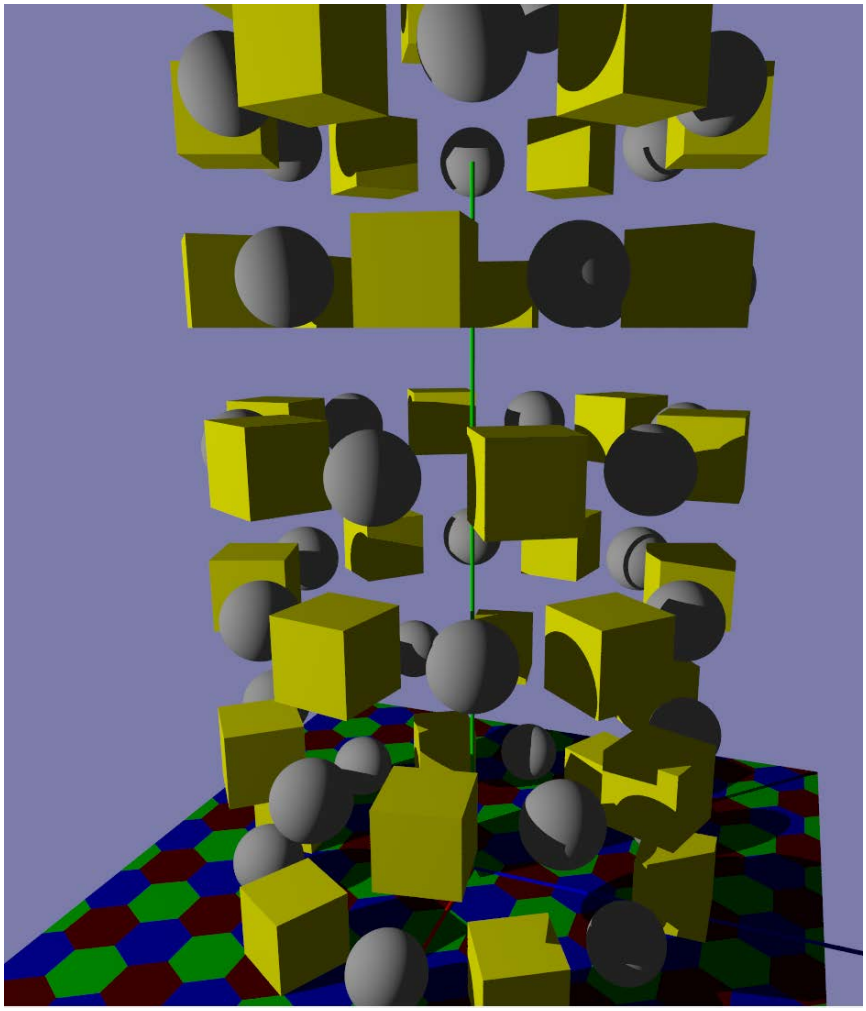
```
#declare Sph = sphere { 0,1 };
#declare Sph_pos = <0,2,0>;
#declare Sph_col = pigment {
                    Green
                   };
#declare Sph_size = 0.4;
// точка с запятой обязательно


object{
    Sph
    pigment {Sph_col}
    translate Sph_pos
    scale Sph_size
}
```

Для объявления локальных переменных (внутри макроса или цикла можно использовать **#local**)

4. Директивы препроцессора

Все что в Pow-Ray обозначается знаком # является директивой препроцессора и разворачивается в обычный Pow-Ray код непосредственно перед

отрисовкой сцены. В примере ниже показана возможность неограниченной вложенности директив друг в друга. Обратите внимание, что в цикле использован **#if #elseif #else**.



```
#macro Cube (pos_x, pos_y, pos_z)
    box{
        -0.5, 0.5
        pigment {Yellow}
        translate <0, 0.5, 0>
        translate <pos_x,pos_y,pos_z>
    }
#end

#macro Sphere (pos_x, pos_y, pos_z)
    sphere{
        0,0.5
        pigment {color 0.5*White}
        translate <0, 0.5, 0>
        translate <pos_x,pos_y,pos_z>
    }

#end


#for (j, 0, 30, 2)//цикл по ярусам
    union{
        #for (i,0, 359, 30) //цикл, рисующий один ярус
            union{

                #if (mod(i,60) = 0)  //чередование кубов/сфер
                    Cube(0,0,3)
                #else
                    Sphere(0,0,3)
                #end
```

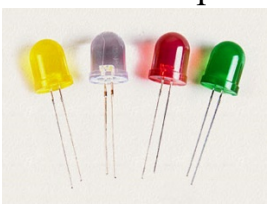
```
                rotate i*y
            }

        #end

        translate <0,j,0>
        rotate j*10*y //немного поворачиваем каждый следующий ярус вокруг y
       }
#end
```

Задания

Нарисовать че нить красивое, объект, сцена. Оценка зависит от степени продуманности объекта.

Примеры того, что можно нарисовать:

1. Панелька из разноцветных светодиодиков. Вот таких:



2. Цветочек в вазе.
3. Умывальник (кран), из которого течет вода.
4. Аквариум с рыбками
5. Чашка с водой и ложкой.
6. Лежащие на столе очки.
7. Ножницы
8. Гарнитура (наушники и микрофон) с проводом, лежащие на столе.
9. Итд…