Pov-Ray

Квик гуиде 1

Оглавление

1 Принципы 2
2 Освещение 4
3 Камера 4
4 Геометрия 4
4.1 Основные примитивы 4
4.1.1 Сфера 4
4.1.2 Бокс 5
4.1.3 Цилиндр 5
4.1.4 Конус 5
4.1.5 Тор 6
4.1.6 Плоскость 6
4.1.7 Призма 6
4.2 Перенос, поворот, масштаб 7
4.3 Сложные объекты 7
4.3.1 Объединение 7
4.3.2 Пересечение 7
4.3.3 Вычитание 8
4.3.4 Слияние 9
5 Цвет и текстура 9
5.1 Однотонная закраска 9
5.2 Сложные типы закраски 9
5.2.1 Градиентная закраска 9
5.2.2 Шахматная закраска 10
5.2.3 Масштабирование закраски 10

1 Принципы

Создание сцены заключается в описании ее составляющих на языке Pov-Ray.

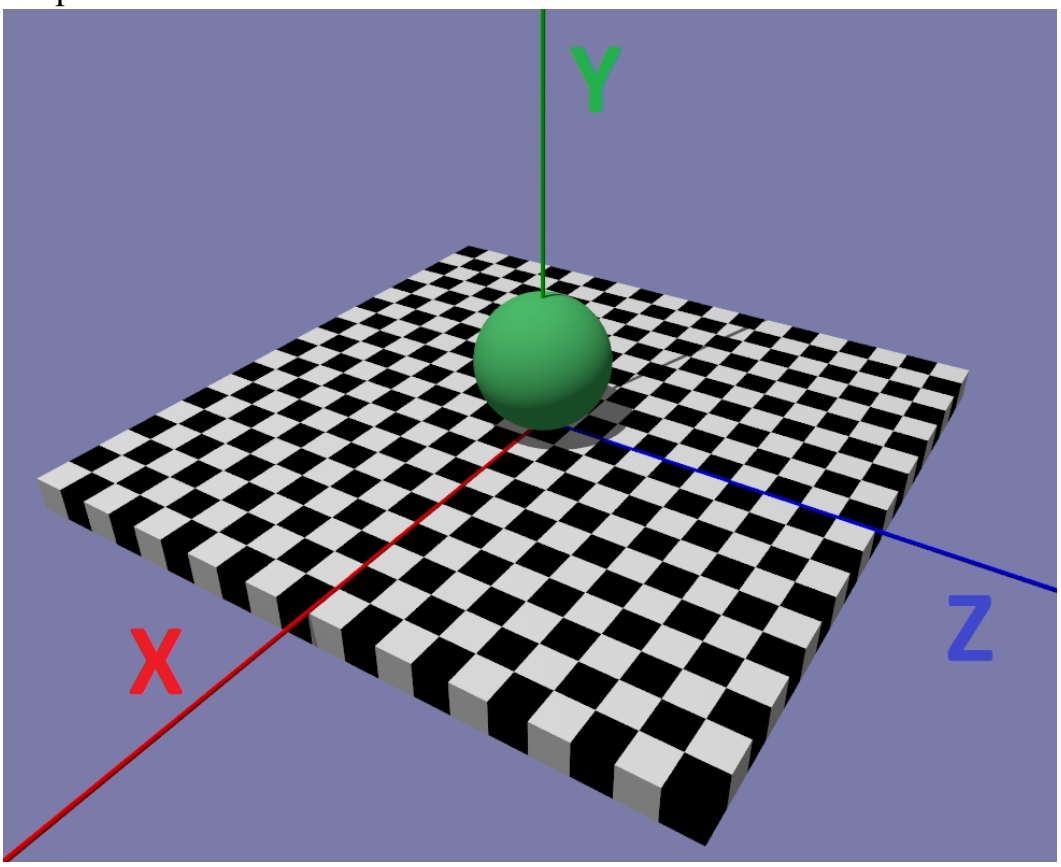
Описание заключается в вызове определенных команд для которых требуется задать определенный набор параметров.

Команды представляют собой структурный код и могут обладать вложенностью, т.е в качестве параметра команды может идти другая команда, для которой так же есть список параметров, итд.

Простейшим случаем задания параметра является передача в него определенных чисел. Например, трех-компонентные точки задаются тремя числами <X,Y,Z>, так же такую точку можно задать одним числом, например 5, означает точку с координатами <5, 5, 5>. Вместо чисел можно использовать математические операции и функции: сложение, умножение, синус, косинус, итд.

Базовая сцена Pov-Ray

Используйте ее, для начала работы в Pov-Ray. Ось Y направленна вверх!




```
#include "colors.inc"

/*
Можно комментировать как в плюсах
```

```
*/

background{   //цвет заднего фона
rgb<0.2,0.2,0.4>
}

camera {
    angle 80  //угол обзора камеры 80 градусов
    location <11,8,7>   //расположение камеры
    look_at  0 //камера смотрит в точку 0 0 0
}

light_source {
    <10,30,-3>   //источник света
    color White   //белого цвета
}

//координатные оси, сделанные из цилиндров
cylinder {
    0, 10*x, 0.03
    pigment {  Red }
}

cylinder {
    0, 10*y, 0.03
    pigment {  Green }
}

cylinder {
    0, 10*z, 0.03
    pigment {  Blue }
}


//пол, сделанный из призмы
prism{
  -0.5,0,4
  <5,5>,<-5,5>,<-5,-5>,<5,-5>
  pigment {
       //шахматная закраска,
       //2 клетки = 1 ед. отрезок
       checker  Black  White   scale .5
   }
}

//РИСОВАТЬ ТУТ !


//Пример сферы
//центр в точке <0,1,0>
//единичный радиус
sphere{
y,1
pigment {rgb<0.1,0.7,0.2>}
}

//PS y = <0,1,0>, одна из констант  в Pov-Ray
```

2 Освещение

Точечный источник белого света в точке (x, y, z)

```
light_source { <X, Y, Z> color rgb 1  }
```

3 Камера

Камера с углом обзора u, с положением в точке (lx, ly, lz) с направлением на точку (rx, ry, rz)

```
  camera {
    angle u // u=70 норм
    location <lx,ly,lz>
    look_at <rx,ry, rz>
  }
```

4 Геометрия

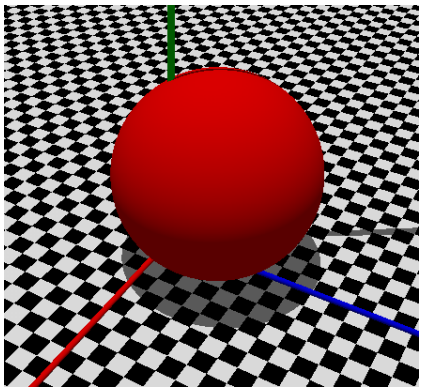
Общий принцип задания объектов

```
КОМАНДА {
      ПАРАМЕТРЫ ПРИМИТИВА
      МОДИФИКАТОРЫ
}
```

Параметры примитива задают его размеры, могут различается для каждых примитивов (бокс, сфера, итд). Модификаторы задают параметры общие для примитивов такие как цвет, закраска, смещение, поворот, итд.

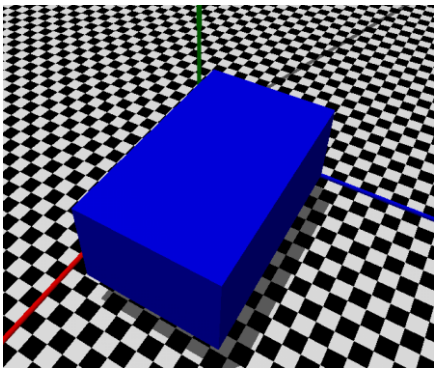
4.1 Основные примитивы

4.1.1 Сфера



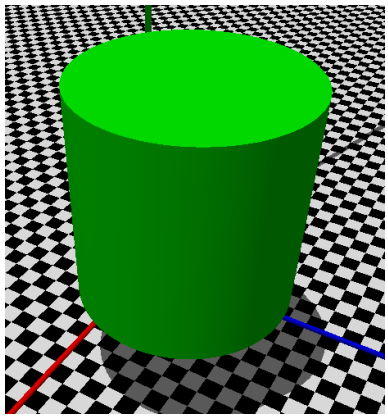
```
sphere {
    <X, Y, Z>, R //координаты центра и радиус
}
```

4.1.2 Бокс



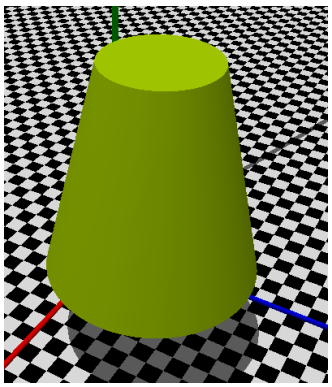
```
box{
     <X,Y,Z>
     <X1,Y1,Z1>  //координаты противоположных углов
}
```

4.1.3 Цилиндр



```
cylinder{
        <X,Y,Z>, <X1, Y1, Z1>, R
        open //делает цилиндр открытым, без верха, не обязательно
             //координаты центра нижней грани
             //координаты центра верхней грани
           //радиус
}
```

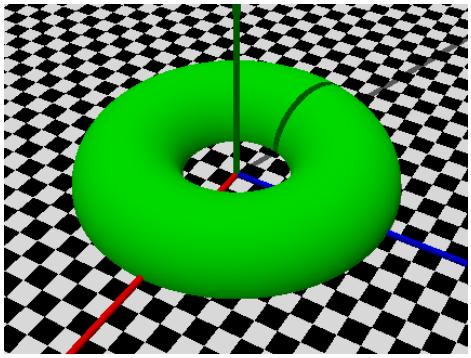
4.1.4 Конус



```
cone {
    <X, Y, Z>, R     // Центр и радиус основания
```

```
    <X1, Y1, Z1>, R1     // Центр и радиус вершины
    open               // Убирает «крышки», не обязательно
}
```

4.1.5 Тор



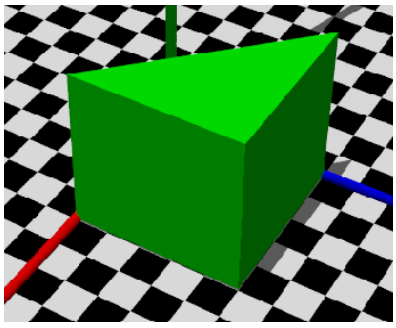
У тора нет настройки его положения, он создается всегда в начале координат. Для управления им нужно использовать перемещения и повороты (п. 4.2)

```
torus {
    R1, R2      // Радиус окружности, радиус «толщины бублика»
}
```

4.1.6 Плоскость

```
plane {
     <X, Y, Z>, D  //нормальный вектор, расстояние до начала
                   //координат
}
```

4.1.7 Призма



```
prism{
     Y1, Y2, N //Y1 уровень нижней грани, Y2 уровень верхней грани
     POINT_1    //N – кол-во точек
     ТОЧКА_2   //точки задаются в плоскости xz
     …                    // в виде <x1,z1>, <x2,z2> …
     ТОЧКА_N
}
```

На рисунке приведена призма, постоянная по трем точкам.

4.2 Перенос, поворот, масштаб

являются модификаторами, могут применятся многократно. Порядок следования играет роль: повернуть-переместить и переместить-повернуть разные вещи. При создании сложных сцен удобнее создавать объекты в начале координат, поворачивать/масштабировать как нужно, а затем перемещать куда нужно.

```
ОБЪЕКТ {
      …
     translate<X,Y,Z>  //перемещение объекта по осям
     rotate<X,Y,Z>  //поворот объекта вокруг осей
     scale<X,Y,Z>  //масштаб объекта по осям
}
```

Примечание:

translate<5,0,0> равносилен translate 5*x;

rotate<0,10,0> равносилен rotate 10*y;

x = <1,0,0>, y=<0,1,0>, z=<0,0,1> удобно использовать бля более компактной записи векторов, коллинеарных ортогональным.

4.3 Сложные объекты

Сложные объекты – объекты, сформированные их других, как простых, так и сложных

4.3.1 Объединение

Служит для объединения других объектов, для последующего применения к ним общих операций. Само по себе объединение не изменяет геометрию объектов.

```
union{
     ОБЪЕКТ_1
     ОБЪЕКТ_2
     …
     ОБЪЕКТ_N
}
```

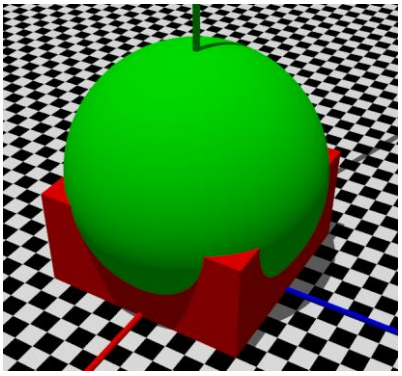
4.3.2 Пересечение

Применение булевой операции «пересечение» к объектам.

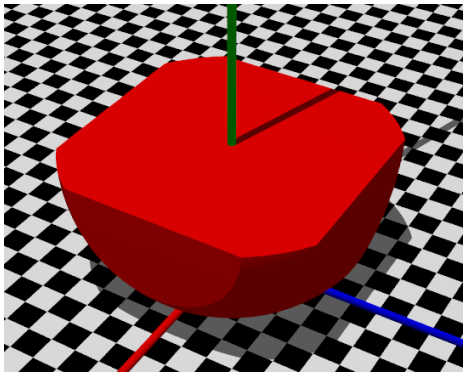
```
intersection {
     ОБЪЕКТ_1
     ОБЪЕКТ_2
}
```

Пример:

Исходные объекты



их пересечение



$$РЕЗУЛЬТАТ = БОКС \cap СФЕРА$$

```
    intersection {
        box
        {
            -1, 1
        }

        sphere
        {
           y, 1.2
        }
        pigment { Red } //цвет
    }
```

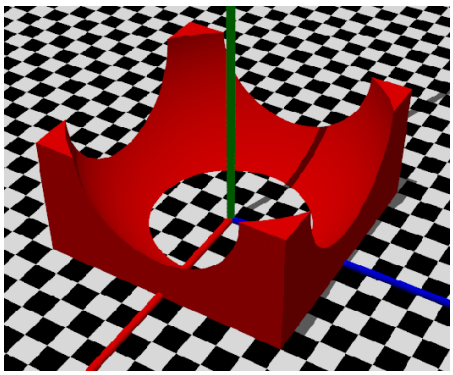
4.3.3 Вычитание

Примирение булевой операции «вычитание»

```
difference {
      ОБЪЕКТ_1
      ОБЪЕКТ_2
}
```

Слияние

$$РЕЗУЛЬТАТ = БОКС - СФЕРА$$



4.3.4 Слияние

Визуально выглядит как объединение, но удаляет «внутренности» получившегося объекта.

```
merge{
      ОБЪЕКТ_1
      ОБЪЕКТ_2
      …
      ОБЪЕКТ_N
}
```

5 Цвет и текстура

```
pigment{
 ТИП_ПИГМЕНТА
 МОДИФИКАТОРЫ_ПИГМЕНТА
}
```

Тип пигмента определяет способ закраски и список модификаторов.

5.1 Однотонная закраска

Способы задания цвета:

```
pigment{
 rgb<R, G, B> //значения цветовых компонент [0..1]
}
```

```
pigment{
 color red R green G blue B> //R G B - значения цветовых
компонент
}
```

```
#include "colors.inc"
//подключить в начале сцены. В нем описаны константы цветов
…
pigment{
 НАЗВАНИЕ_ЦВЕТА // Red, Green, Blue, Yellow, Cyan, Magenta, Clear,
                //Black. В файле colors.inc описано намного больше
                // цветов. Регистр соблюдать обязательно

}
```

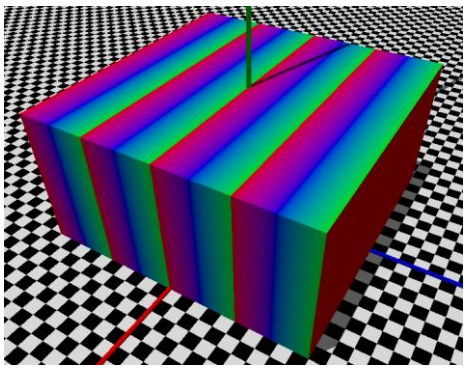
5.2 Сложные типы закраски

5.2.1 Градиентная закраска

```
pigment {
      gradient N       //Направление градиента: вектор
      color_map {
        [i1 ЦВЕТ1]
        [i2 ЦВЕТ2]
         …
```

```
        [iN ЦВЕТN]
        //i1, i2, … ,iN числа [0..1] записаные в порядке
        //возрастания
      }
    }
```

Пример:



```
box{
  -2,  2
  pigment {
      gradient z  //направление по оси z
      color_map {
        [0   Red]
        [.5 Blue]
        [1   Green]
      }
  }
}
```

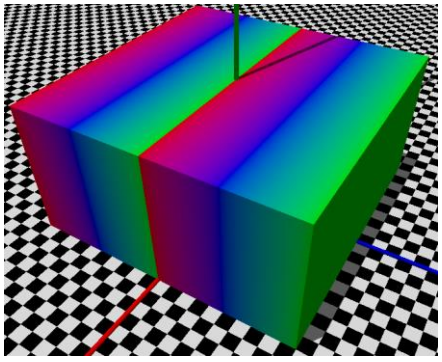
5.2.2 Шахматная закраска

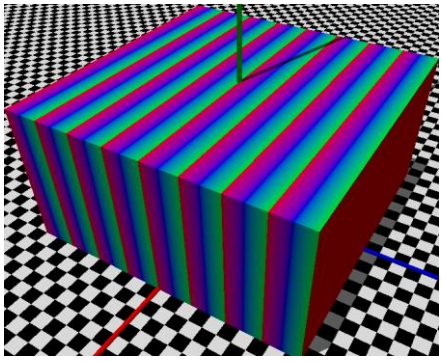
```
pigment {
       checker ЦВЕТ1  ЦВЕТ2
    }
```

5.2.3 Масштабирование закраски

происходит посредством применения модификатора scale

```
pigment {
       …
       scale S //по умолчанию S=1
    }
```





scale 2 scale 0.5

Задание на лабораторную работу:

25: Нарисовать плоскость и разместить на ней 5 различных разноцветных примитивов (описанных в 4.1)

30: Сделать домик (бокс – тело дома, призма - крыша, цилиндр – труба, все стоит на плоскости), выделить примитивы разными цветами.

35: Сделать кусок сыра с дырками. (используется операция вычитания), кусок сыра – призма.

40: 35 + сыр должен лежать на тарелке с бортиками.

45: Сделать домик с комнатой внутри, с применением булевых операций проделать дверной проем с окнами, разместить в домике дополнительный источник света. Примерный состав домика: боксы – стены и пол, призма – крыша, цилиндр – труба

Добавление дополнительных элементов в задание может повысить его стоимость. (например - забор вокруг домика, или положить сыр помимо тарелки на стол), низкое качество выполнения – понизить.