Лабораторная работа № 3.

Изоповерхности

1. Общее понятие о функциональном представлении геометрических тел, изоповерхностях.

Изоповерхности - поверхности, проходящие через точки с одинаковым значением какой-либо величины и характеризующие распределение этой величины в пространстве. Изоповерхности родственны изолиниям, которые применяют в климатической географии, когда обозначают на картах высоты, среднегодовое количество осадков и т.д.

Изоповерхности в компьютерной графике применяются совместно с функциональным геометрическим представлением объектов (Functional representation, F-rep).

Функциональное представление (Functional representation, F-rep), основано на неравенстве

$$f(x, y, z) \geq 0 \qquad (1).$$

которое описывает подпространство в 3х мерном пространстве, представляющее собой твердое тело. Таким образом, изоповерхность при $f(x, y, z) = 0$ будет представлять собой границу твердого тела.

Представление твердого тела в виде нераверства (1) является неявным, т.е. заранее не известно в каких областях и точках тело существует, а в каких – нет, а значит для визуализации требуется произвести либо дискретизацию всего пространства, где может находиться функционально заданный объект (алгоритм «Марширующие кубы»-«Marching cubes»), и в конце дискретизации получить явное представление его границы, либо, для систем на основе трассировки луча, ввести алгоритм нахождения пересечения луча и границы функционального объекта.

Визуализация программы PovRay основана на алгоритме трассировки луча и использует следующий алгоритм для поиска границы функционально описанного объекта (см рис. 1):

Находятся границы пересечения трассируемого луча и области, указанной как область возможного нахождения функционального описанного объекта. Две точки – точка вхождения в область поиска и точка выхода из области поиска – являются основными для последующего рекурсивного деления отрезка, вычисления значений на границе.

В отличие от других фигур в PovRay, изоповерхности апроксимируются во время визуализации, что часто вызывает определенные неудобства в их использовании. Однако, такого рода поверхности имеют ряд полезных особенностей, в частности возможность моделировать реальные деформации, сдвиги в поверхностях и другое. С помощью изоповерхностей можно задать сложной формы объекты, что либо сложно, либо невозможно сделать с помощью методов CSG.

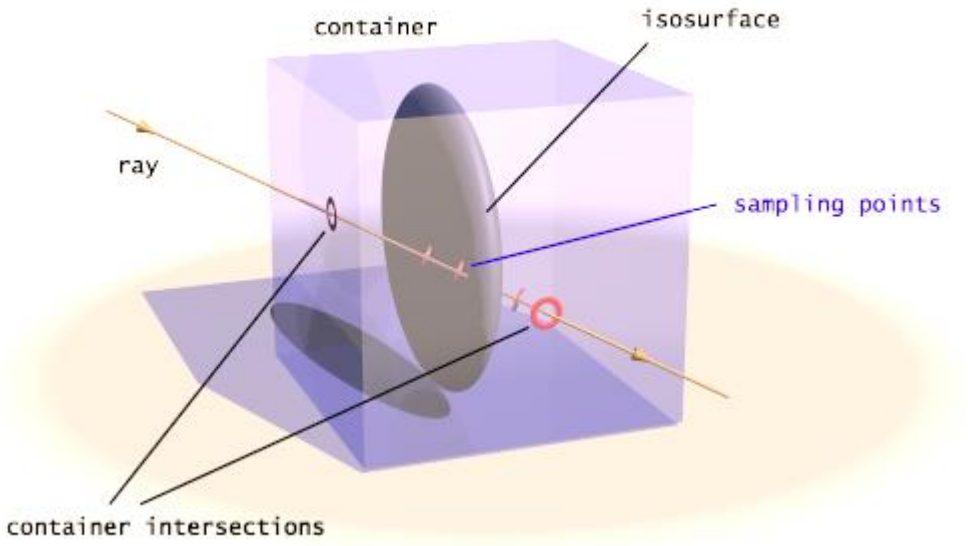


Рис 1. Иллюстрация к алгоритму пересечения функционального объекта и луча трассировки.

2. Использование изоповерхностей в программе PovRay.

Разберем простейший пример:

```
#include "colors.inc"
camera { location  <0, 0, -4> look_at <0, 0, 0>}
light_source {<-100,200,-100> colour White}


#declare  S = function {x*x + y*y + z*z - 1}

isosurface {
    function { S(x,y,z)
             }
     threshold     0.0
     accuracy      0.01
     contained_by{box{-10,10}}
     max_gradient 20
     pigment {Yellow}
         }
```

После подключения файла с предопределенными цветами, установки камеры и источника освещения, определяется функция S.

Детально рассмотрим назначение основных параметров объекта **isosurface:**

function { ... } – описание математической функции, которая впоследствии и будет определять основные свойства изоповерхности.

threshold – параметр, который определяет, насколько объемной будет изоповерхность.

$$f(x, y, z) \geq \text{threshold}$$

Поверхность рисуется тогда, когда её значение равно параметру threshold. По умолчанию используется threshold, равный 0.

Accuracy - для построения поверхностей в PovRay используется метод рекурсивного деления луча на отрезки, которое продолжается до тех пор, пока длина отрезка, в котором PovRay ищет точку пересечения луча и изоповерхности, не станет меньше значения параметра accuracy. По умолчанию, accuracy = 0,001. Меньшее значение позволяет строить более четкие поверхности, но при этом затрачивая больше ресурсов для рендеринга.

Contained_by – объект-контейнер, который ограничивает область, в которой PovRay изображает описанную изоповерхность. Этим контейнером может быть либо параллелепипед, либо сфера. Задаются объекты–контейнеры с помощью стандартного синтаксиса:

contained_by { sphere { CENTER, RADIUS } }
contained_by { box { CORNER1, CORNER2 } }

Если контейнер не указывается, то принимается значение по умолчанию
box {<-1,-1,-1>, <1,1,1>}

max_gradient - PovRay находит первую точку пересечения луча с изоповерхностью, описанной любой непрерывной функцией, если известно значение параметра max_gradient. Для получения корректного изображения, необходимо правильно задать величину параметра max_gradient. Если значение max_gradient будет указано слишком маленьким, то велика вероятность появления изъянов в изображении, например, в объекте могут получиться дырки или незаполненные полосы.

Для простых изоповерхностей, особенно тех, что имеют симметричную форму, очень просто рассчитать max_gradient из математических соображений.

Рассмотрим пример построения изоповерхности, задаваемой следующим образом:

```
isosurface {
 function { x*x + y*y + z*z - 1 }
     accuracy 0.0001
     contained_by{sphere{0,1.2}}
     pigment {rgb .9}
}
```

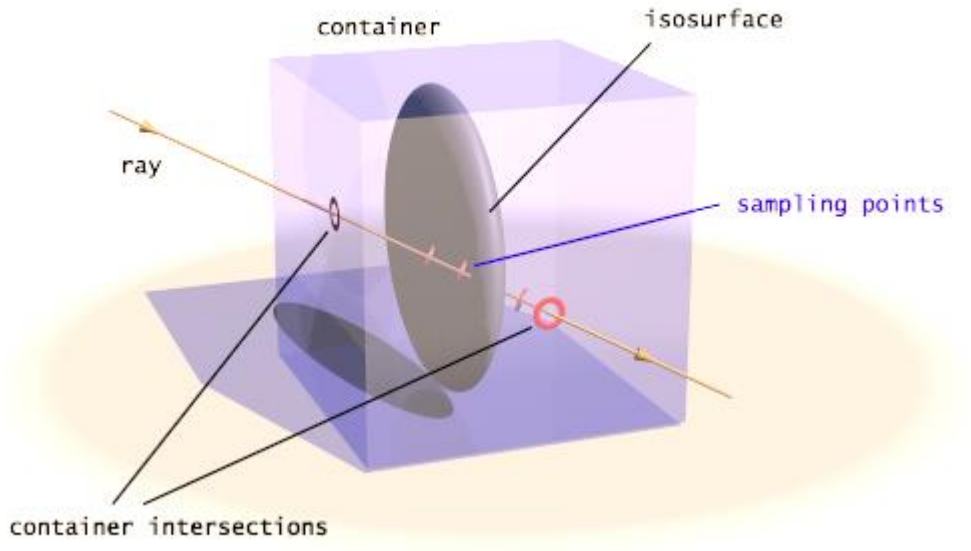
Поскольку параметр max_gradient не уточнён, принято значение по умолчанию max_gradient = 1.1, что значительно меньше реального, отсюда и результат изображения – часть сферы не прорисована, так как луч не проверил

на наличие точек изоповерхности те области, которые выходят за пределы максимального угда наклона луча, принятого по умолчанию равным 1.1.

Для данной изоповерхности расчёт max_gradient можно произвести из следующих соображений: сфера, задающая изоповерхность – объект, симметричный в любом направлении. Рассчитаем max_gradient в направлении вдоль оси x (значения y и z остаются неизменными). Тогда функция, задающая сферу будет иметь упрощенный вид: x*x – 1, её градиент gradient = 2*x. Функция ограничена сферой с центром в точке (0,0,0) и радиусом 1.2, следовательно градиент $\text{gradient} \subseteq [-2.4; 2.4]$, откуда находим max_gradient = 2.4.

Open - наличие слова **open** в с писке параметров, задающих изоповерхность, говорит о том, что видимая часть объекта – контейнера должна быть удалена (см. пример в п. «примеры простейших изоповерхностей») .

Теперь, зная назначение параметров, задаваемых нашу изоповерхность, принцип её построения в системе PovRay становится совсем понятным:



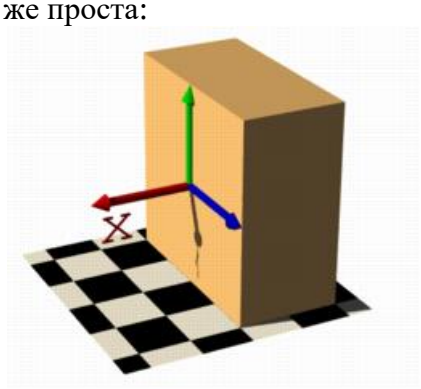
как было сказано выше, при построении изоповерхности используется метод рекурсивного деления: отрезок луча прошедшего внутрь объекта-контейнера рекурсивно делится, при этом на границах этих отрезков производятся вычисления значений функции, задающей изоповерхность. Деление происходит до тех пор, когда при условии, что максимальный угол наклона луча не превышает max_gradient по значениям функции можно будет понять, что в исследуемых точках – концах отрезков нет пересечения луча и изоповерхности , либо когда длина отрезка луча достигнет заданного параметра точности accuracy.

Примеры простейших изоповерхностей.

Для начала рассмотрим простую функцию f(x, y, z) = x. Значениями этой функции будет ось OX. Описание изоповерхности на встроенном языке PovRay будет выглядеть так:

```
isosurface {
   function { x }
   contained_by { box { -2, 2 } }
 }
```

Поверхность в результате так же проста:

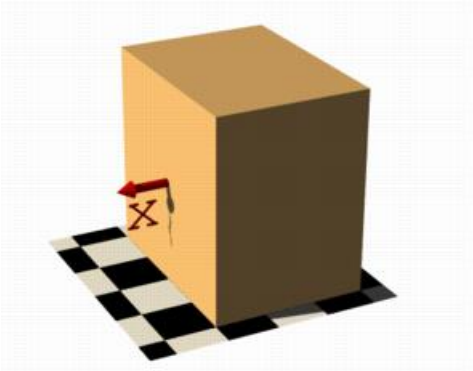


Причина, по которой мы получили коробку – использование параллелепипеда в качестве объекта – контейнера для изоповерхности, в списке параметров его нет, и, в данном примере по умолчанию используется box {<-1, -1, -1>, <1, 1, 1>}.

То есть фактически только одна сторона бокса получена описанной функцией, - плоскость, значение x вдоль которой равно нулю, так как значение параметра threshold (граница, предел) по умолчанию установлено равным 0.

Рассмотрим вариант описанной изоповерхности, когда threshold = 1. Тогда:

```
isosurface {
   function { x }
   contained_by { box { -2, 2 } }
   threshold 1
 }
```

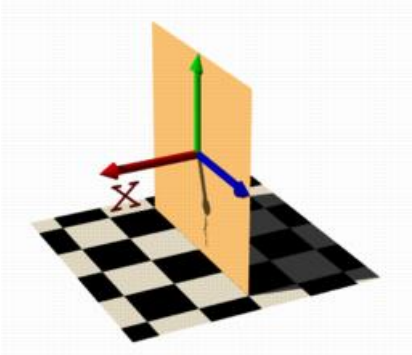


Добавив слово open в определении изоповерхности, можно удалить видимую часть объекта- контейнера, тогда из определения:

```
isosurface {
```

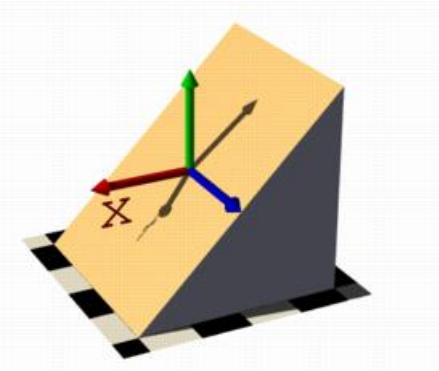
```
  function { x }
  contained_by { box { -2, 2 } }
  open
}
```

получим плоскость, перпендикулярную оси OX, как показано на рисунке:

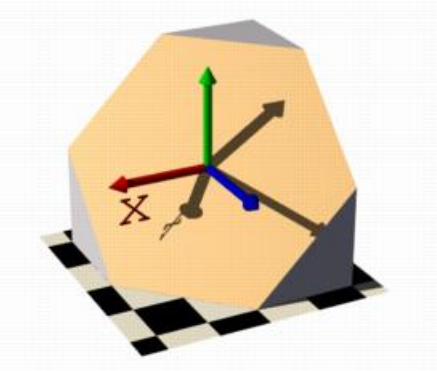


Далее рассмотрим изоповерхности, заданные более сложными функциями:

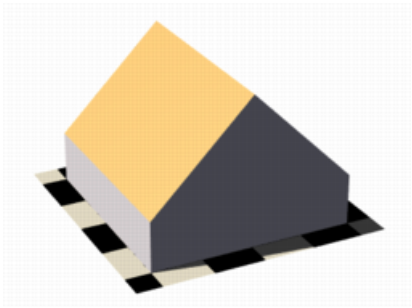
1. `function {x + y }`



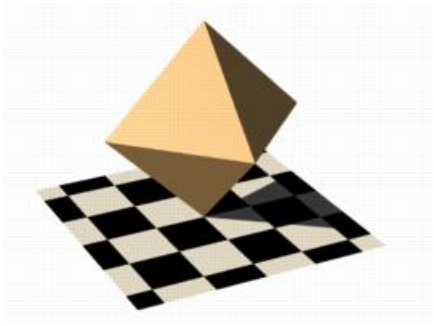
2. `function {x + y + z}`



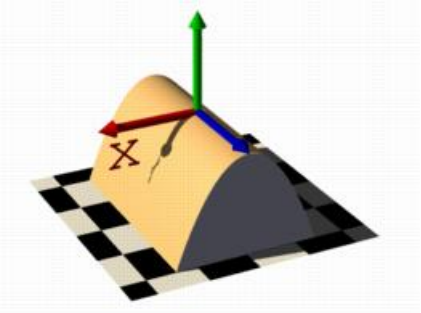
3. `function {abs(x)- 1 + y }`



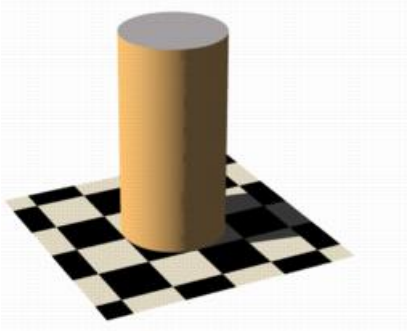
4. `function {abs(x) + abs(y) + abs(z) - 2 }`



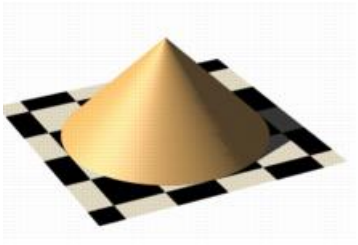
5. `function {pow(x, 2) + y }`



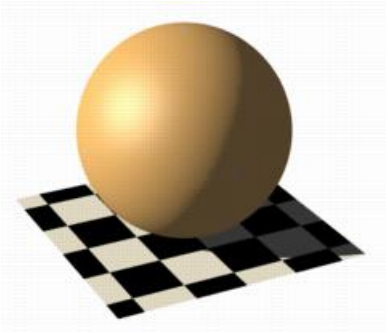
6. `function {sqrt(pow(x, 2) + pow(z, 2)) - 1 }`



7. `function {sqrt(pow(x, 2) + pow(z, 2)) + y }`



```
8. function {sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2)) - 2 }
```



Преобразования изоповерхностей.

К изоповерхностям применимы все преобразования, доступные для любого другого объекта в PovRay. Если вы хотите изменить положение или размер изоповерхности внутри объекта – контейнера, то следует изменять значения параметров, от которых зависит функция, описывающая изоповерхность.

Следует так же отметить, что изменение значений параметров для достижения желаемого результата необходимо производить в направлении, обратном тому эффекту, который мы хотим получить. Например, чтобы перенести сферу Sphere(x, y, z) с центром в точке (0,0,0) на две единицы в положительном направлении вдоль оси y, преобразование координат должно выглядеть так: Sphere(x, y - 2, z). Объясняется это просто: изначально координата центра сферы y = 0, наша цель – сделать её равной 2, то есть y = 2, отсюда получаем y-2 = 0.

Аналогичным образом делается масштабирование изоповерхности.

Итак, кратко об основных преобразованиях:

1. **Параллельный перенос** : $F(x,y,z) \to F(x - xtr, y - ytr, z - ztr)$,

   F – функция описывающая изоповерхность, xtr, ytr, ztr – величина переноса вдоль соответствующей оси.

2. **Масштабирование**: $F(x,y,z) \to F(\frac{x}{sc\_x}, \frac{y}{sc\_y}, \frac{z}{sc\_z})$,

   sc_x, sc_y, sc_z – величина масштабного преобразования изоповерхности вдоль соответствующей координаты.

   Для того, чтобы бесконечно масштабировать изоповерхность, необходимо заменить соответствующую координату на 0. В качестве примера можно привести преобразование сферы в цилиндр:

```
#include "functions.inc"
#include "colors.inc"
```

```
camera { location  <0, 0, -4> look_at <0, 0, 0>}
light_source {<-100,200,-100> colour rgb 1}
background  {Brown}

#declare R = 2;
#declare  S = function {x*x + y*y + z*z - 1}

isosurface {
  function { S(0,y,z) }
       max_gradient 3
       contained_by{sphere{0,R}}
       pigment {Yellow}
}
```

3. <u>Сдвиг относительно плоскости</u>

Осуществляется следующим преобразованием:

a) В плоскости XY: $F(x,y,z) \to F\left(x + y * \tan\left(radians(angle)\right), y, z\right)$;

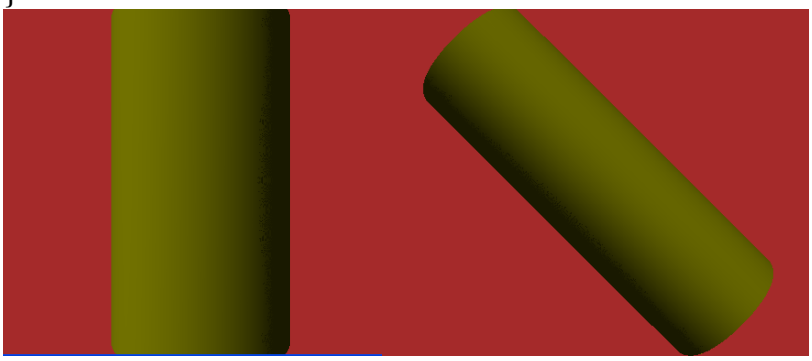
b) В плоскости YZ: $F(x,y,z) \to F\left(x, y + z * \tan\left(radians(angle)\right), z\right)$;

c) В плоскости XZ: $F(x,y,z) \to F\left(x + z * \tan\left(radians(angle)\right), y, z\right)$.

Пример, сдвиг цилиндра, направленного вдоль оси Y, на 45 градусов в плоскости XY:

```
#include "functions.inc"
#include "colors.inc"

camera { location  <0, 0, -4> look_at <0, 0, 0>}
light_source {<-100,200,-100> colour rgb 1}
background  { Brown}

#declare R = 2;
#declare  S = function {x*x + z*z  - 1}

isosurface {
  function { S(x+y*tan(radians(45)),y,z) }
       max_gradient 11
```

```
        contained_by{sphere{0,R}}
        pigment {Yellow}
}
```



4. **Поворот** осуществляется следующим образом:

   a) Вокруг оси **X**: $F(x,y,z) \to F(x,y',z')$, где
   $y' = \mathrm{z} * \sin(\mathrm{radians(angle)}) + \mathrm{y} * \cos(\mathrm{radians(angle)})$,
   $z' = \mathrm{z} * \cos(\mathrm{radians(angle)}) - \mathrm{y} * \sin(\mathrm{radians(angle)})$.

   b) Вокруг оси **Y**: $F(x,y,z) \to F(x',y,z')$, где
   $x' = \mathrm{x} * \cos(\mathrm{radians(angle)}) - \mathrm{z} * \sin(\mathrm{radians(angle)})$,
   $z' = \mathrm{x} * \sin(\mathrm{radians(angle)}) + \mathrm{z} * \cos(\mathrm{radians(angle)})$.

   c) Вокруг оси **Z**: $\mathrm{F(x,y,z)} \to \mathrm{F(x',y',z)}$, где
   $x' = \mathrm{x} * \cos(\mathrm{radians(angle)}) + \mathrm{y} * \sin(\mathrm{radians(angle)})$,
   $z' = -\mathrm{x} * \sin(\mathrm{radians(angle)}) + \mathrm{y} * \mathrm{scos(radians(angle))}$.

5. **Зеркальное отображение**

   1) Относительно плоскости XY: $F(x,y,z) \to F(y,-x,z)$;
   2) Относительно плоскости YZ: $F(x,y,z) \to F(x,z,-y)$;
   3) Относительно плоскости XZ: $F(x,y,z) \to F(-z,y,x)$.

6. **Кручение** изоповерхности на n оборотов вокруг оси X достигается следующим преобразованием: $F(x,y,z) \to F(x,y',z')$, где
   $y' = \mathrm{z} * \sin(\mathrm{x} * 2 * \mathrm{pi} * n) + \mathrm{y} * \cos(\mathrm{x} * 2 * \mathrm{pi} * \mathrm{n})$,
   $z' = \mathrm{z} * \cos(\mathrm{x} * 2 * \mathrm{pi} * \mathrm{n}) - \mathrm{y} * \sin(\mathrm{x} * 2 * \mathrm{pi} * \mathrm{n})$.
   Аналогично для других осей (см. преобразования для поворота).

Операции над изоповерхностями.

Для работы с изоповерхностями доступны аналоги всех тех операций над объектами, которые определены в CSG:

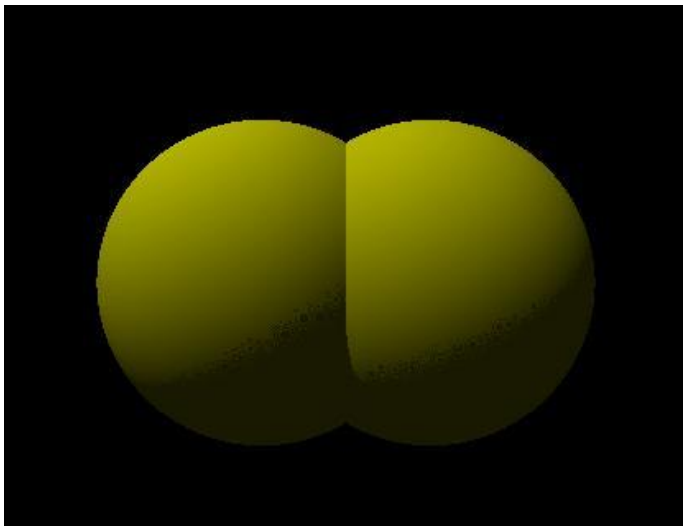
1. **Объединения** нескольких изоповерхностей можно добиться при использовании функции min (S1, S2, .., Sn), где S1, …, Sn – функции, описывающие изоповерхности.

Пример:

```
 #include "colors.inc"

camera { location  <0, 0, -4> look_at <0, 0, 0>}
light_source {<-100,200,-100> colour rgb 1}

#declare R = 2;
#declare S = function {x*x + y*y + z*z - 1}

isosurface {
  function { min(S(x+0.5,y,z), S(x-0.5,y,z)) }
      max_gradient 2
      contained_by{sphere{0,R}}
      pigment {Yellow}
}
```

В результате получаем объединение двух сфер, параллельно перенесённых вдоль оси x на -/+ 0.5 соответственно.

Данная операция может выполняться в отношение двух и более изоповерхностей.

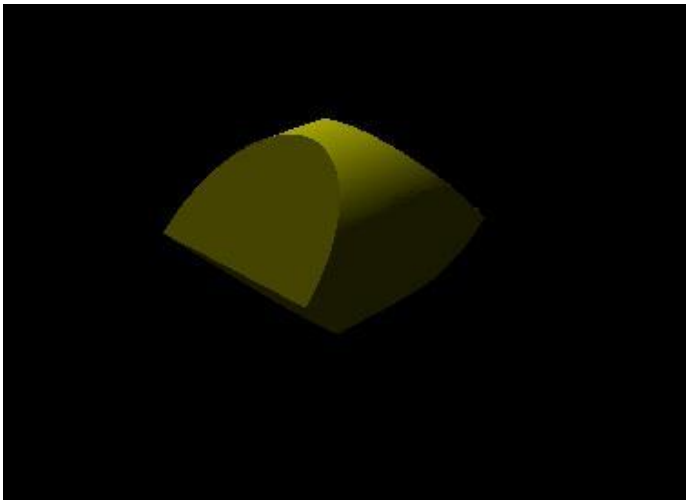


2. Аналогичным образом определяется операция **пересечения** нескольких изоповерхностей. При этом используется функция max (S1, S2, …, Sn), где S1,..Sn – функции, описывающие изоповерхности.

   Рассмотрим действие функции max (S1, S2), где S1, S2 – функции, задающие цилиндр и параллелепипед соответственно.

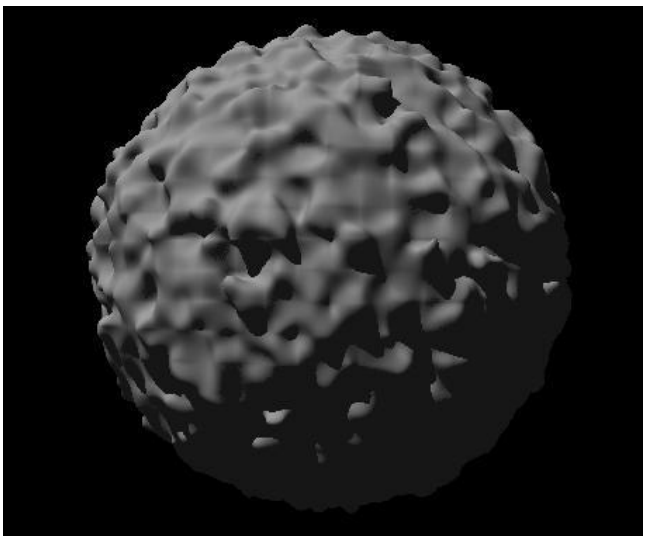
```
#include "colors.inc"
#include "functions.inc"

camera {
      location <5, 3.5, 5>
      look_at  <3, 2,  3>

      angle 50
      }
```

```
light_source {<-100,200,-100> colour rgb 1}

#declare S1 = function { sqrt(pow(y,2) + pow(z,2)) - 0.8 }
#declare S2 = function { abs(x)+abs(y)-1 }

isosurface {
  function { max(S1(x,y,z), S2(x,y,z)) }
       max_gradient 2
       pigment {Yellow}
}
```



3. **Суммирование** **или** **вычитание** изоповерхностей друг из друга можно задать с помощь обычного сложения или вычитания функций, задающих эти изоповерхности.
   Рассмотрим примеры:
   a) Сумирование двух изоповерхностей:

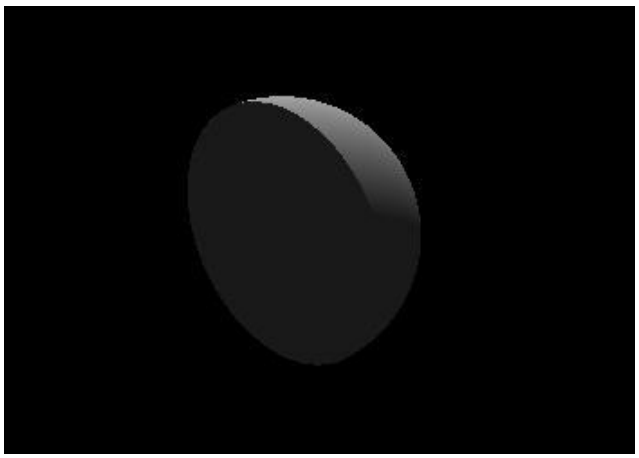
```
#include "functions.inc"

  camera { location  <0, 0, -4> look_at <0, 0, 0> angle 40}
  light_source {<-100,200,-100> colour rgb 1}

  #declare R = 1.1;
  #declare  S = function {x*x + y*y + z*z - 1}

   isosurface {
  function { S(x,y,z) +
  f_noise3d(x*10, y*10, z*10)*0.3 }
       max_gradient 7
       contained_by{sphere{0,R}}
       pigment {rgb .9}
  }
```



Как видно из рисунка, в результате сложения двух функций получаем что-то подобное «смеси» этих функций. В данном примере мы получили фигуру сферической формы, но с шероховатостью на поверхности, что объясняет наложение шероховатости функцией f_noise3d(x*10, y*10, z*10)*0.3 на сферу S = function {x*x + y*y + z*z - 1}.

b) Разность двух изоповерхностей:

```
#include "functions.inc"
#include "colors.inc"

camera {
      location <5, 3.5, 5>
      look_at  <3, 2,  3>
        angle 50
      }
light_source {<-100,200,-100> colour rgb 1}

#declare R = 1.1;
#declare  S = function {x*x + y*y + z*z - 1}

isosurface {
  function { S(x,y,z) - S(x-0.1, y,z)}

      max_gradient 2
      contained_by{sphere{0,R}}
      pigment {White}
}
```

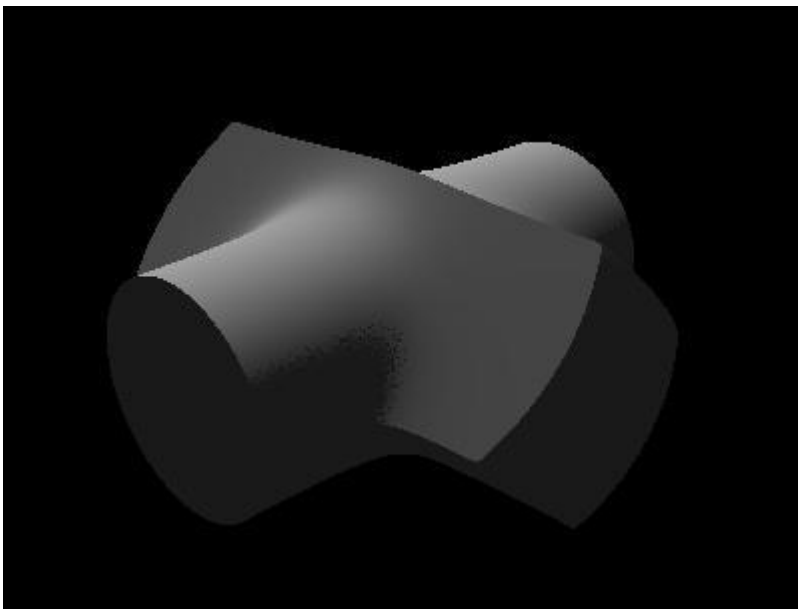


4. Гладкое сопряжение

С помощью умножения функций, задающих изоповерхности, можно добиться эффекта, подобного тому, что достигается при работе с blob – объектами – к примеру, плавного перехода одной поверхности в другую.

Рассмотрим пример, в котором параллелепипед плавно переходит в цилиндр:

```
#include "functions.inc"
#include "colors.inc"

camera {
      location <5, 3.5, 5>
      look_at  <3, 2,  3>
      angle 50
      }
light_source {<-100,200,-100> colour rgb 1}

#declare S1 = function { sqrt(pow(y,2) + pow(z,2)) - 0.8 } // цилиндр
#declare S2 = function { abs(x)+abs(y)-1 } // перевернутый параллелепипед


isosurface {
  function { S1(x,y,z)*S2(x, y,z)-0.03}

      max_gradient 6
      contained_by{sphere{0,2}}
      pigment {White}
}
```



Данный прием удобнее тем, что он применим при работе с любыми поверхностями, а не только со сферами и цилиндрами, как при работе с blob-объектами.

3. Примеры функционально описанных объектов:

Примечание: символ ^ означает возведение в степень.

-z+5+50*sin(sqr((x+2)^2+y^2))

-z+5+sin(sqr((x+40)^2+y^2))+sin(sqr(x^2+y^2))

Две затухающие волны

-z+5+5*exp(-0.2*sqr(x^2+y^2)+0.1)*sin(sqr(x^2+y^2)-3.14/2)+9*exp(-0.2*sqr((x-30)^2+y^2)+0.1)*sin(sqr((x-30)^2+y^2)-3.14/2)

конус

-z+5+sqr((x+2)^2+y^2)

конус с пимпой

mzero(-20+sqr((x+2)^2+y^2))*(-20+sqr((x+2)^2+y^2)-z)+mzero(20-sqr((x+2)^2+y^2))*(-z+50)

Ограничения

mzero(-x+2)*mzero(-y+7)

mzero(-y+7)*mzero(x+3)*mzero(-x+20)

Параллелипипид

mzero(-y+20)*mzero(x+3)*mzero(-x+20)*mzero(y-8)*mzero(z-10)*mzero(-z+20)

3хмерный синус

-z+sqr((25-(y-6*sin(x/6))^2)*mzero(25-(y-6*sin(x/6))^2))

3хмерный синус, вдавленный в параллипипид

-z-sqr((25-(y-sin(x))^2)*mzero(25-(y-sin(x))^2))

3хмерный синус, вдавленный в параллипипид

(-z+20-sqr((25-(y-sin(x))^2)*mzero(25-(y-sin(x))^2)))*mzero(-y+10)*mzero(x+10)*mzero(-x+40)*mzero(y+10)*mzero(z-10)

Сверло

(100-(3*cos(0.5*z)-x)^2-(3*sin(0.5*z)-y)^2)*mzero(z-sqr((x+2)^2+y^2))

-z+sqr((25-(y-6*sin(0.5*x))^2)*mzero(25-(y-6*sin(0.5*x))^2))

-z+(atn(y/x)*mzero(x)+(atn(y/x)+3.14/2)*mzero(-x))

Сердечко
(-((2*z^2+x^2+y^2-1)^3-(0.1*z^2+x^2)*y^3))

Глазик с веком
(-((z^2+x^2+y^2-10)^3-(0.9*z^2+y)*y^2))

Цветочек
(-((z^2+x^2+y^2-10)^3-(2*(z*x)^2)*y^5))

Другие примеры изоповерхностей можно посмотреть в файле C:\Program Files\POV-Ray for Windows v3.6\include**shapesq.inc,** или справке по ключевому слову isosurface.